Proiect Baze de Date

Muzeu mașini de colecție

Dumitrescu Marian-Daniel
CR 3.2A

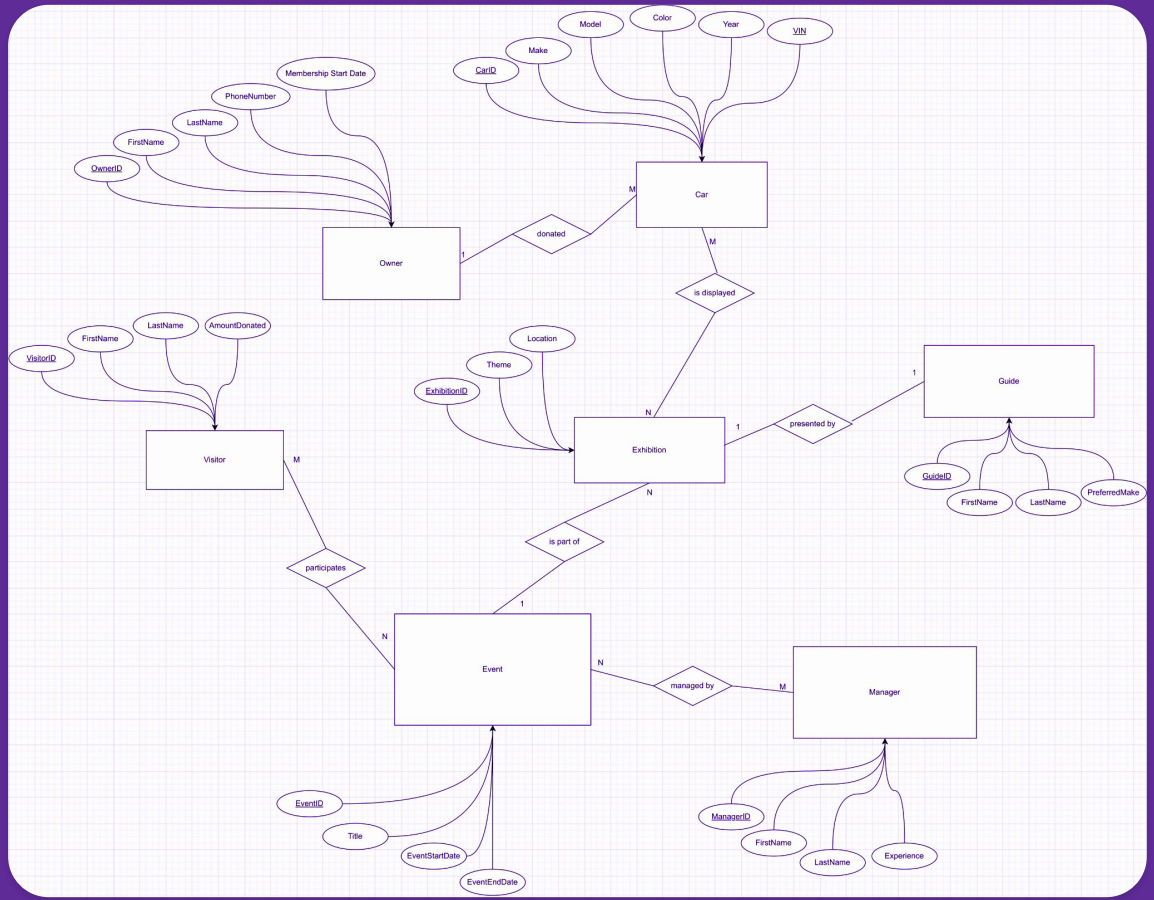
Link repository git

Descriere temă aleasă

Descriere bază de date

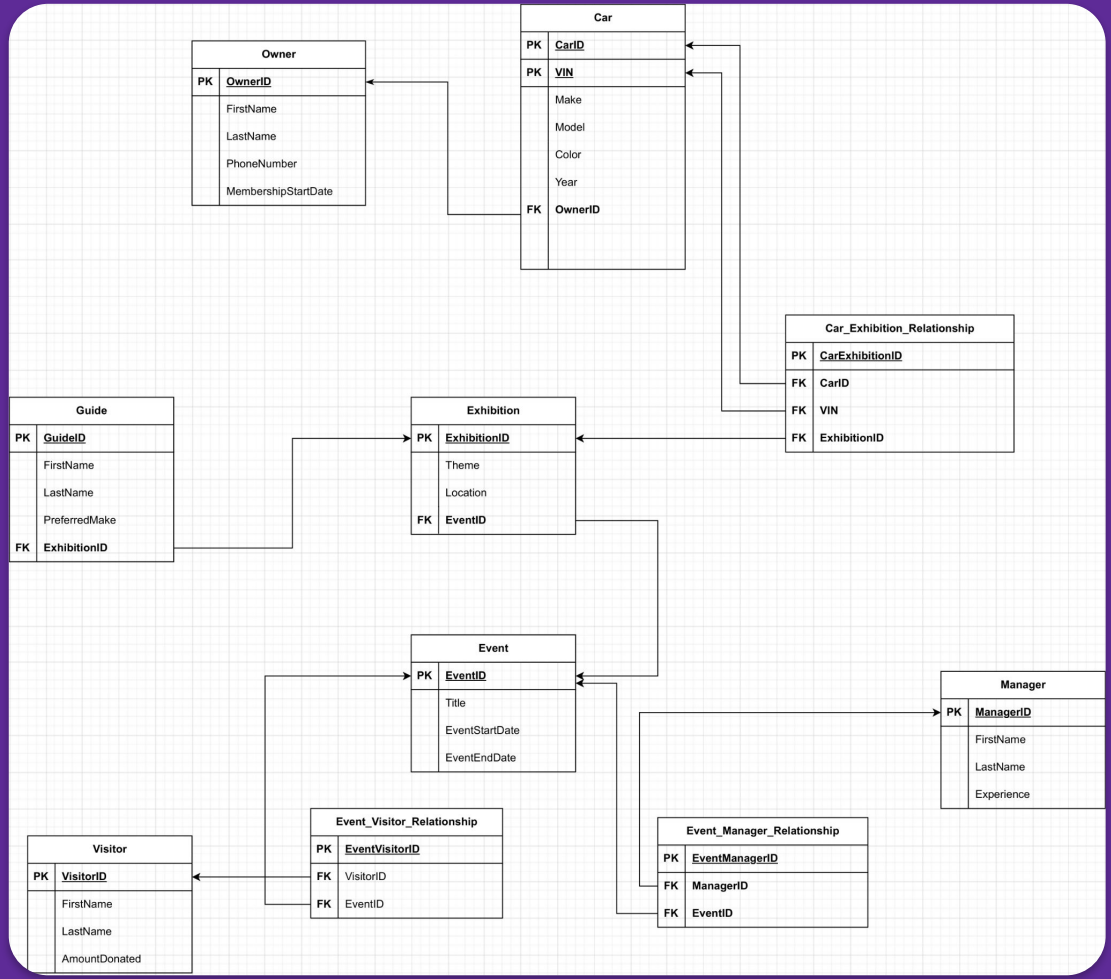
- Am proiectat și implementat o bază de date care descrie un muzeu la care se prezintă expoziții de mașini unor vizitatori care pot dona unei organizații caritabile.
- Fiecare mașină este donată pentru eveniment de către fiecare membru al muzeului (proprietar), iar mașinile vor fi expuse la o expoziție prezentată de către un ghid la un anume eveniment.
- Fiecare expoziție va avea o anumită temă, ca exemplu putem numi „Mașini sport istorice” o expoziție la care se vor afișa mașini cu renume în lumea motorsportului.

Diagrama ER



- Am folosit notația Chen

Schema Relațională



Crearea tabelelor

- Am creat tabelele descrise în schema relațională și am pus de asemenea constrângerile și relațiile dintre tabele prin foreign key.
- Query-urile pentru crearea tabelelor este aici.

Inserarea datelor în tabele

- Am continuat prin a insera în tabelele descrise.
- Query-urile pentru inserarea datelor în tabele este aici.

Exemple operații modificare date

- UPDATE Owner SET PhoneNumber = '0722000123' WHERE OwnerID = 1;
- UPDATE Car SET Color = 'Navajo Green' WHERE CarID = 2;
- UPDATE Event SET Title = 'Expozitie de masini de lux' WHERE EventID = 1;

Exemple operații ștergere date

- DELETE FROM Owner WHERE OwnerID = 5;
- DELETE FROM Car WHERE CarID = 3;
- DELETE FROM Car WHERE CarID = 3;

Exemple de operații de modificare sau ștergere folosind subinterogări:

- UPDATE Car SET OwnerID = (SELECT OwnerID FROM Owner
  WHERE LastName = 'Ionescu') WHERE CarID = 1;

- DELETE FROM Car WHERE OwnerID
  IN(SELECT OwnerID FROM Owner WHERE FirstName = 'Vasile');

- UPDATE Event SET Title = 'Zilele masinilor clasice' WHERE EventID
  IN (SELECT EventID FROM Event WHERE DATE_PART('MONTH', EventStartDate) = 9);

Normalizarea relațiilor

Normalizarea relațiilor prezente

- În cadrul tabelelor descrise, se respectă primele trei forme normalizate (**1NF**, **2NF** şi **3NF**).
- Putem considera mai multe situaţii pe această bază de date pentru a nu mai respecta:
  - **1NF** - În cazul în care vom avea o coloană într-un tabel care va conţine mai multe elemente, 1NF nu se va mai respecta.
    - ***Exemplu***: *OwnerPhoneNumber* păstrează mai multe numere de telefon, *etc*.
  - **2NF** - În cazul în care un atribut dependent funcţional de un anume tabel este prezent în alt tabel nu se va mai respecta 2NF.
    - ***Exemplu***: Adăugăm coloana *OwnerFirstName* în *Car*, *etc.*
  - **3NF** - În cazul în care un atribut nu este dependent tranzitiv de cheia primară.
    - ***Exemplu***: Atributul *OwnerPhoneNumber* se află în tabela *Car*, el trebuind să fie tranzitiv dependent prin cheia primară *OwnerID*, nu prin *(CarID, VIN)*, *etc*.

Interogări

Exemple interogări cu joncțiuni (Partea I)

- *Afișare membrii împreună cu mașina donată la expoziții*

```
SELECT O.FirstName, O.LastName, C.Make, C.Model
FROM Owner O
INNER JOIN Car C ON O.OwnerID = C.OwnerID;
```

- *Afișarea titlului evenimentelor și numele vizitatorilor care au participat la aceste evenimente*

```
SELECT E.Title, V.FirstName, V.LastName
FROM Event E
LEFT OUTER JOIN Event_Visitor_Relationship EVR ON E.EventID = EVR.EventID
LEFT OUTER JOIN Visitor V ON EVR.VisitorID = V.VisitorID;
```

Exemple interogări cu joncțiuni (Partea II)

- *Afișare manageri împreună cu evenimentele pe care le gestionează*

```
SELECT M.FirstName, M.LastName, E.Title
FROM Manager M
INNER JOIN Event_Manager_Relationship EMR ON M.ManagerID = EMR.ManagerID
INNER JOIN Event E ON EMR.EventID = E.EventID;
```

- *Afișarea ghizilor împreună cu expozițiile pe care le prezintă*

```
SELECT G.FirstName, G.LastName, E.Theme
FROM Guide G
LEFT OUTER JOIN Exhibition E ON G.ExhibitionID = E.ExhibitionID;
```

Exemple de interogări cu funcții de agregare și GROUP BY (Partea I)

- *Afișarea numărului de mașini pentru fiecare membru*

```
SELECT (O.FirstName || ' ' || O.LastName) AS MemberName, COUNT(*) AS NumberOfCars
FROM Owner O
JOIN Car C ON O.OwnerID = C.OwnerID
GROUP BY O.OwnerID
ORDER BY NumberOfCars DESC;
```

- *Afișarea sumei donațiilor pentru fiecare eveniment*

```
SELECT E.EventID, SUM(V.AmountDonated) AS TotalDonations
FROM Event E
JOIN Event_Visitor_Relationship EVR ON E.EventID = EVR.EventID
JOIN Visitor V ON EVR.VisitorID = V.VisitorID
GROUP BY E.EventID;
```

Exemple de interogări cu funcții de agregare și GROUP BY (Partea II)

- *Afișarea numărului de mașini expuse la fiecare expoziție*

```
SELECT E.ExhibitionID, COUNT(*) AS NumberOfCarsPresented
FROM Exhibition E
JOIN Car_Exhibition_Relationship CER ON E.ExhibitionID = CER.ExhibitionID
JOIN Car C ON CER.CarID = C.CarID AND CER.VIN = C.VIN
JOIN Owner O ON C.OwnerID = O.OwnerID
GROUP BY E.ExhibitionID;
```

- *Numărul de mașini fabricate de către fiecare producător*

```
SELECT C.Make, COUNT(*) AS NumberOfCars
FROM Car C
GROUP BY C.Make;
```

Exemple de subinterogări corelate

- Afişarea maşinilor expuse în evenimentele gestionate de un anumit manager

```
SELECT Car.CarID, Car.VIN, Car.Make, Car.Model, Car.Color
FROM Car
JOIN Car_Exhibition_Relationship ON Car.CarID = Car_Exhibition_Relationship.CarID
AND Car.VIN = Car_Exhibition_Relationship.VIN
JOIN Exhibition ON Car_Exhibition_Relationship.ExhibitionID = Exhibition.ExhibitionID
JOIN Event ON Exhibition.EventID = Event.EventID
JOIN Event_Manager_Relationship ON Event.EventID = Event_Manager_Relationship.EventID
JOIN Manager ON Event_Manager_Relationship.ManagerID = Manager.ManagerID
WHERE Manager.FirstName = 'Ionel' AND Manager.LastName = 'Ionescu';
```

- Afişarea vizitatorilor care au donat mai mult decât media donaţiilor la evenimentele cu o anumită tematică de expoziţie

```
SELECT DISTINCT Visitor.FirstName, Visitor.LastName, Visitor.AmountDonated
FROM Visitor
JOIN Event_Visitor_Relationship ON Visitor.VisitorID = Event_Visitor_Relationship.VisitorID
JOIN Event ON Event_Visitor_Relationship.EventID = Event.EventID
JOIN Exhibition ON Event.EventID = Exhibition.EventID
WHERE Visitor.AmountDonated > (
   SELECT AVG(Visitor.AmountDonated)
   FROM Visitor
   JOIN Event_Visitor_Relationship ON Visitor.VisitorID = Event_Visitor_Relationship.VisitorID
   JOIN Event ON Event_Visitor_Relationship.EventID = Event.EventID
   JOIN Exhibition ON Event.EventID = Exhibition.EventID
   WHERE Exhibition.Theme = 'Porsche'
);
```

Exemple de subinterogări necorelate

- Afişarea maşinilor care nu au fost expuse în expoziţii şi nu au fost prezentate de niciun ghid asociat unei expoziţii

```
SELECT Car.CarID, Car.VIN, Car.Make, Car.Model, Car.Color
FROM Car
LEFT JOIN Car_Exhibition_Relationship ON Car.CarID = Car_Exhibition_Relationship.CarID
AND Car.VIN = Car_Exhibition_Relationship.VIN
LEFT JOIN Guide ON Car_Exhibition_Relationship.ExhibitionID = Guide.ExhibitionID
WHERE Car_Exhibition_Relationship.CarID IS NULL
AND Car_Exhibition_Relationship.VIN IS NULL
AND Guide.GuideID IS NULL;
```

- Afişarea titlului unui eveniment şi numărul de vizitatori pentru evenimentul la care a participat o persoana cu prenumele 'Marian'

```
SELECT E.Title, COUNT(*) AS NumberOfVisitors
FROM Event E
WHERE E.EventID IN (
  SELECT EVR.EventID
  FROM Event_Visitor_Relationship EVR
  WHERE EVR.VisitorID IN (
    SELECT V.VisitorID
    FROM Visitor V
    WHERE V.FirstName = 'Marian'
  )
)
GROUP BY E.Title;
```

Exemple de interogări folosind funcții pe șiruri de caractere

- LOWER și UPPER

```
SELECT LOWER(FirstName) AS LowercaseFirstName, UPPER(LastName) AS UppercaseLastName
FROM Owner;
```

- CONCAT echivalent cu operatorul || infixat

```
SELECT CONCAT(FirstName, ' ', LastName) AS FullName --echivalent FirstName || ' ' || LastName
FROM Owner;
```

- LENGTH

```
SELECT LENGTH(FirstName) AS FirstNameLength
FROM Owner;
```

Exemple de interogări folosind funcții pe date calendaristice

- EXTRACT

```
SELECT EXTRACT(YEAR FROM MembershipStartDate) AS Year
FROM Owner;
```

- AGE

```
SELECT AGE(MembershipStartDate) AS MembershipDuration
FROM Owner;
```

- DATE_PART

```
SELECT DATE_PART('month', MembershipStartDate) AS Month
FROM Owner;
```

Exemple de interogări folosind CASE

- ```
  SELECT CarID,
  CASE
    WHEN Year < 2000 THEN 'Masina Clasica'
    WHEN Year BETWEEN 2000 AND 2010 THEN 'Masina Relativ Noua'
    ELSE 'Masina Noua'
  END AS Age
  FROM Car;
  ```

- ```
  SELECT *
  FROM Car
  ORDER BY CASE
    WHEN Make = 'Mercedes-Benz' THEN 1
    ELSE 2
  END;
  ```

Vederi

Exemple de vederi + Operații permise/nepermise (Partea I)

- *Vedere asupra evenimentelor, titlu + suma donațiilor la eveniment (*Doar interogări SELECT sunt permise, INSERT/UPDATE/DELETE nu se poate face*)*

```
CREATE VIEW EventDonations AS
SELECT E.EventID, E.Title, SUM(V.AmountDonated) AS TotalDonations
FROM Event E
JOIN Event_Visitor_Relationship EVR ON E.EventID = EVR.EventID
JOIN Visitor V ON EVR.VisitorID = V.VisitorID
GROUP BY E.EventID, E.Title;
```

- *Vedere asupra membrilor* (Permise toate interogările, SELECT/UPDATE/INSERT/DELETE)

```
CREATE VIEW OwnerView AS
SELECT OwnerID, FirstName, LastName FROM Owner;
```

Exemple de vederi + Operații permise/nepermise (Partea II)

- Vedere asupra mașinilor (Permise toate operațiile)

```
CREATE VIEW CarView AS
SELECT CarID, Make, Model, Year FROM Car;
```

Indecși

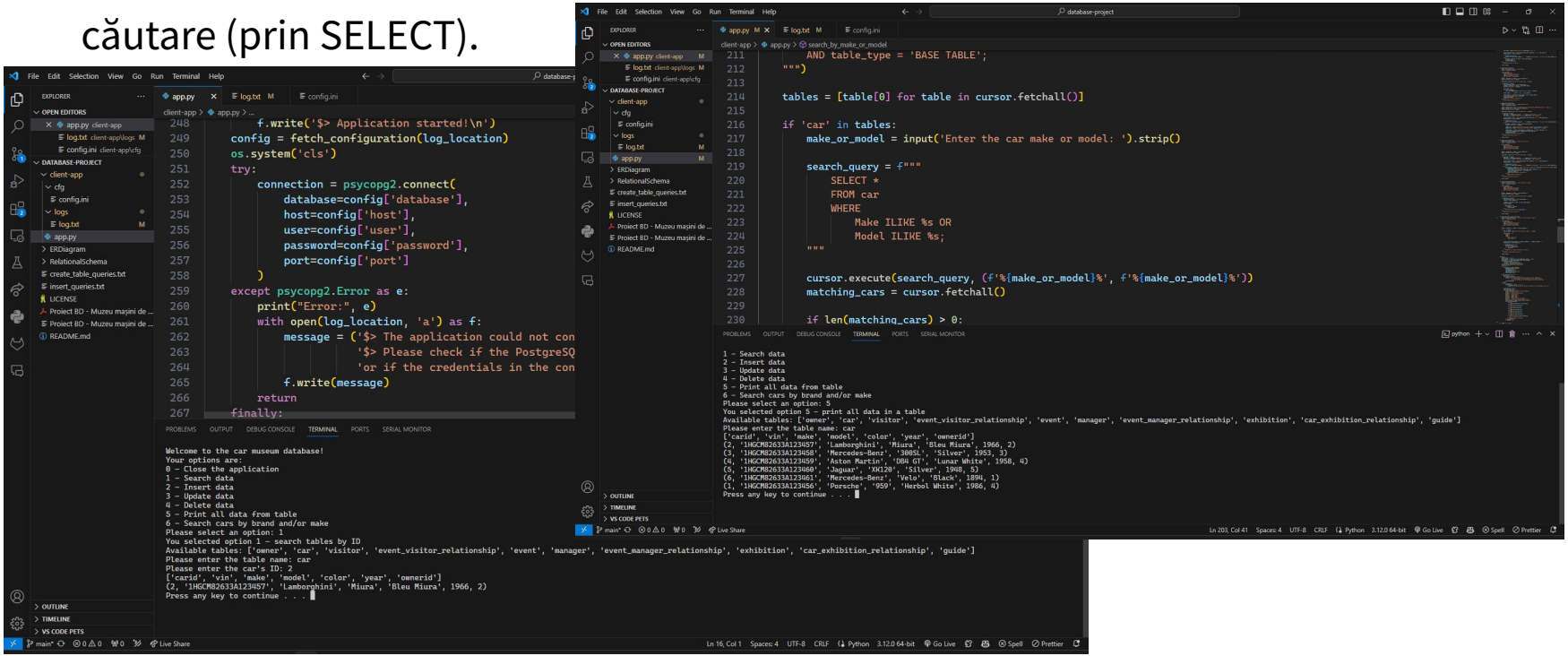
Exemplu creare index pentru optimizare interogări

- ```
  CREATE INDEX idx_car_make_model
  ON Car (Make, Model);
  -- optimizare pentru căutare cu 2 criterii
  -- SELECT * FROM Car
  -- WHERE Make = 'Mercedes-Benz' AND Model = '300SL';
  ```
  ```

Aplicație Client

Descriere aplicație client

- Pentru implementarea aplicației, am folosit modulul de conectare la baze de date PostgreSQL *psycopg2* pentru Python și am descris o aplicație simplă în consolă care ne permite operații de manipulare a datelor (INSERT, UPDATE, DELETE) și căutare (prin SELECT).



Modulul *psycopg2*

- Acest modul pentru limbajul de programare **Python** îmi permite interogarea serverului **PostgreSQL** cu query-uri trimise chiar din aplicația implementată, într-o manieră asemănătoare cu *API*-ul din **Java *JDBC***.
- Fiecare interogare se bazează pe un obiect de tip Connection care leagă aplicația de serverul PostgreSQL specificat, iar acesta poate returna mai multe obiecte de tip Cursor care pot executa query-uri SQL.

```
# Conectare server PostgreSQL
connection = psycopg2.connect(
        database=config['database'],
        host=config['host'],
        user=config['user'],
        password=config['password'],
        port=config['port']
    )
```

```
# Exemplu query SQL
cursor = connection.cursor()
cursor.execute("SELECT * FROM car;")
print(cursor.fetchall())
cursor.close()
```