Homework1

Phạm Ngọc Hải

March 27, 2024

1 Bài 1.

Name four major disadvantages of a file system to manage structured data like banking, or airline data.

Sử dụng hệ thống tệp tin truyền thống (file system) để quản lý dữ liệu có cấu trúc (structured data) như dữ liệu ngân hàng hoặc hàng không máy bay có thể gặp một số hạn chế:

1. **Thiếu tính toàn vẹn dữ liệu**:
   Hệ thống tệp tin thường thiếu các cơ chế tích hợp để đảm bảo tính toàn vẹn của dữ liệu. Thiếu kiểm soát đúng đắn, dữ liệu có thể bị hỏng hoặc không nhất quán, dẫn đến lỗi trong giao dịch hoặc sự quản lý không chính xác của thông tin quan trọng. Trong hệ thống ngân hàng hoặc hàng không, nơi độ chính xác là rất quan trọng, tính toàn vẹn dữ liệu là điều rất quan trọng để ngăn chặn mất mát tài chính hoặc các vấn đề an toàn.

2. **Khả năng mở rộng hạn chế**:
   Hệ thống tệp tin có thể không mở rộng tốt với lượng dữ liệu ngày càng tăng và sự phức tạp gia tăng. Khi lượng dữ liệu và số lượng người dùng truy cập tăng lên, hệ thống tệp tin có thể trở nên không hiệu quả và khó quản lý. Hạn chế này có thể cản trở khả năng của các hệ thống ngân hàng hoặc hàng không để xử lý một lượng lớn các giao dịch hoặc người dùng, dẫn đến những hạn chế về hiệu suất và chất lượng dịch vụ suy giảm.

3. **Vấn đề Đồng thời**:
   Hệ thống tệp tin thường gặp khó khăn trong việc xử lý truy cập đồng thời vào dữ liệu bởi nhiều người dùng hoặc quy trình. Thiếu các cơ chế kiểm soát đồng thời đúng đắn, dữ liệu có thể bị hỏng hoặc không nhất quán khi được truy cập đồng thời bởi nhiều người dùng. Trong các hệ thống ngân hàng hoặc hàng không nơi mà nhiều người dùng có thể cần truy cập và sửa đổi dữ liệu đồng thời, vấn đề đồng thời có thể dẫn đến lỗi dữ liệu, giao dịch thất bại hoặc vi phạm bảo mật.

4. **Bảo mật dữ liệu hạn chế**:
   Hệ thống tệp tin có thể thiếu các tính năng bảo mật mạnh mẽ để bảo vệ dữ liệu nhạy cảm khỏi truy cập, sửa đổi hoặc đánh cắp trái phép. Thiếu biện pháp bảo mật phù hợp như mã hóa, kiểm soát truy cập và nhật ký kiểm tra, các hệ thống ngân hàng hoặc hàng không có nguy cơ bị nhiều mối đe dọa bảo mật, bao gồm việc vi phạm dữ liệu, trộm danh tính và gian lận. Trong các ngành nơi tính bảo mật và quyền riêng tư dữ liệu là rất quan trọng, sự thiếu hụt các biện pháp bảo mật mạnh mẽ trong hệ thống tệp tin có thể tạo ra những rủi ro đáng kể đối với tổ chức và khách hàng của nó.

2 Bài 2.

Is SQL a procedural language or a declarative language? Give three reasons why something like SQL is a better choice than say C++ for writing database applications.

SQL (Structured Query Language) là một ngôn ngữ mô tả (declarative language), mặc dù nó có một số yếu tố thủ tục (procedural language).

Dưới đây là ba lý do tại sao SQL có thể là lựa chọn tốt hơn so với C++ để viết ứng dụng cơ sở dữ liệu:

- **Đơn giản và Dễ đọc**: SQL được thiết kế đặc biệt để truy vấn và điều chỉnh cơ sở dữ liệu, làm cho nó đơn giản và dễ đọc hơn cho các nhiệm vụ liên quan đến cơ sở dữ liệu so với các ngôn ngữ lập trình đa mục đích như C++. Cú pháp của nó là trực quan và ngắn gọn, cho phép nhà phát triển diễn đạt các hoạt động cơ sở dữ liệu phức tạp một cách rõ ràng và dễ hiểu. Sự đơn giản này có thể dẫn đến các chu kỳ phát triển nhanh hơn và bảo trì ứng dụng cơ sở dữ liệu dễ dàng hơn.
- **Tối ưu cho Các Hoạt động Cơ sở Dữ liệu**: SQL được tối ưu cho các hoạt động cơ sở dữ liệu, có nghĩa là nó cung cấp các cách hiệu quả để truy xuất, cập nhật và quản lý dữ liệu được lưu trữ trong cơ sở dữ liệu. Các hệ thống cơ sở dữ liệu SQL được tối ưu cao cho việc xử lý truy vấn, tạo chỉ mục và quản lý giao dịch, dẫn đến hiệu suất tốt hơn so với các hoạt động cơ sở dữ liệu tùy chỉnh được triển khai trong các ngôn ngữ như C++. Bằng cách tận dụng các khả năng tối ưu hóa tích hợp của SQL, nhà phát triển có thể tạo ra các ứng dụng cơ sở dữ liệu có khả năng mở rộng, đáng tin cậy và hiệu suất hơn.
- **Các Tính năng Bảo mật tích hợp**: SQL cung cấp các tính năng bảo mật tích hợp để quản lý kiểm soát truy cập, xác thực và bảo vệ dữ liệu trong các ứng dụng cơ sở dữ liệu. Các tính năng này bao gồm xác thực người dùng, kiểm soát truy cập dựa trên vai trò, mã hóa và che dấu dữ liệu, giúp các nhà phát triển triển khai các biện pháp bảo mật mạnh mẽ để bảo vệ thông tin nhạy cảm được lưu trữ trong cơ sở dữ liệu. So với C++, nơi nhà phát triển phải triển khai các tính năng bảo mật thủ công, SQL đơn giản hóa quá trình đảm bảo bảo mật dữ liệu và tuân thủ các yêu cầu quy định.

3 Bài 3.

In words, describe the left natural outer join operation over two relations, R(A,B) and S(B,C).

Phép nối bên trái tự nhiên (left natural outer join) giữa hai quan hệ R(A,B) và S(B,C) là một phép nối dữ liệu trong cơ sở dữ liệu mà **kết quả bao gồm tất cả các bản ghi từ quan hệ R và các bản ghi từ quan hệ S mà có giá trị cột B chung**. Kết quả của phép nối này bao gồm tất cả các cột từ quan hệ R và các cột từ quan hệ S (trừ cột B, vì đã có trong R).

Trong quan hệ kết quả, mỗi bản ghi từ R sẽ được nối với các bản ghi từ S mà có giá trị cột B tương ứng với cột B trong bản ghi từ R. Nếu không có bản ghi trong S phù hợp với bản ghi từ R, các cột từ quan hệ S sẽ được điền bằng giá trị NULL. Điều này có nghĩa là kết quả của phép nối bên trái tự nhiên sẽ bao gồm tất cả các bản ghi từ quan hệ R và các bản ghi từ quan hệ S mà có giá trị cột B chung, với các cột từ quan hệ S bổ sung và các cột còn lại sẽ giữ nguyên từ quan hệ R.

4 Bài 4.

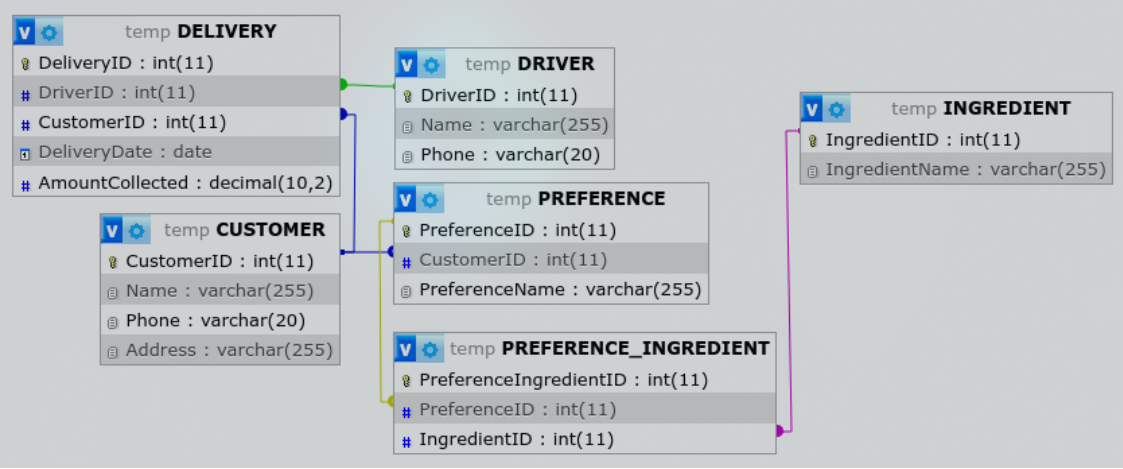
Design a relational schema for a pizza delivery place. The database is to keep track of CUSTOMERs, their PREFERENCEs, and for each preference the INGREDIENTs (onions, ham, bacon, etc.). Each customer may have several preferences. Each preference CONTAINs one or more ingredients. The database has to keep track of the DRIVERs delivering, dates, and money collected. Identify the primary keys, candidate keys, and foreign keys.

Dưới đây là code SQL xây dựng Schema theo yêu cầu bài toán.

```sql
-- Create CUSTOMER table
CREATE TABLE CUSTOMER (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(255),
    Phone VARCHAR(20),
    Address VARCHAR(255)
);

-- Create PREFERENCE table
CREATE TABLE PREFERENCE (
    PreferenceID INT PRIMARY KEY,
    CustomerID INT,
    PreferenceName VARCHAR(255),
    FOREIGN KEY (CustomerID) REFERENCES CUSTOMER(CustomerID)
);

-- Create INGREDIENT table
CREATE TABLE INGREDIENT (
    IngredientID INT PRIMARY KEY,
    IngredientName VARCHAR(255)
);

-- Create PREFERENCE_INGREDIENT table
CREATE TABLE PREFERENCE_INGREDIENT (
    PreferenceIngredientID INT PRIMARY KEY,
    PreferenceID INT,
    IngredientID INT,
    FOREIGN KEY (PreferenceID) REFERENCES PREFERENCE(PreferenceID),
    FOREIGN KEY (IngredientID) REFERENCES INGREDIENT(IngredientID)
);

-- Create DRIVER table
CREATE TABLE DRIVER (
    DriverID INT PRIMARY KEY,
    Name VARCHAR(255),
    Phone VARCHAR(20)
);

-- Create DELIVERY table
```

```sql
CREATE TABLE DELIVERY (
    DeliveryID INT PRIMARY KEY,
    DriverID INT,
    CustomerID INT,
    DeliveryDate DATE,
    AmountCollected DECIMAL(10, 2),
    FOREIGN KEY (DriverID) REFERENCES DRIVER(DriverID),
    FOREIGN KEY (CustomerID) REFERENCES CUSTOMER(CustomerID)
);
```

Sau đây là hình ảnh schema thiết kế



5 Bài 5.

Section 2.3.3 (5th edition) or Problem 6.4 (6th edition) describe the "division operation". What is the result of the operation (R / S) on the following two instances?

Trường hợp 1:
Instance 1: R

A	B
1	α
2	β
3	γ
1	β
2	γ
2	α

Instance 1: S

B
α

B
γ

For each tuple in S, we want to find all tuples in R where B matches. Then we'll take the common A values from those tuples. Let's calculate:

- For $B = \alpha$:
  - R tuples with $B = \alpha$: (1, α), (2, α)
  - A values: {1, 2}
- For $B = \gamma$:
  - R tuples with $B = \gamma$: (3, γ), (2, γ)
  - A values: {2, 3}

The result of the division operation (R / S) contains only those A values that appear in both sets: {2}.

So, (R / S) yields:

A
2

Trường hợp 2:
Instance 2: R

A	B	C
3	1	α
3	2	β
1	3	γ
2	1	α
3	1	β
1	2	γ
3	2	α

Instance 2: S

B
2

For each tuple in S, we want to find all tuples in R where B matches. Then we'll take the common A values from those tuples. Let's calculate:

- For $B = 2$:

- R tuples with $B = 2$: (1, 2, γ), (3, 2, β), (3, 2, α)
- A values: {1, 3}
- C values: $\{\alpha, \beta, \gamma\}$

The result of the division operation (R / S) contains only those A values that appear in both sets: {1, 3}.

So, (R / S) yields:

A	C
1	γ
3	β
3	α

6 Bài 6.

Consider the two SQL queries.
Are these equivalent? Examine the case of p and/or r being empty.

```
select p.a
from p
where p.b <>all (select r.b from r);

-- and

select p.a
from p
where p.b not in (select r.b from r);
```

Câu lệnh SQL này sử dụng toán tử so sánh để tìm các giá trị trong cột B của bảng P không xuất hiện trong bất kỳ giá trị nào của cột B trong bảng R. Tuy nhiên, có một sự khác biệt quan trọng giữa hai câu lệnh này, và nó phụ thuộc vào cách SQL xử lý các trường hợp khi bảng r hoặc bảng p trống.

1. **Query 1:**

   ```
   select p.a
   from p
   where p.b <> all (select r.b from r);
   ```

   Điều này chọn các giá trị từ cột A của bảng P mà không bằng bất kỳ giá trị nào trong cột B của bảng R. Trong trường hợp bảng R hoặc P rỗng, câu lệnh này sẽ trả về tập hợp rỗng, không có bất kỳ giá trị nào.

2. **Query 2:**

   ```
   select p.a
   from p
   where p.b not in (select r.b from r);
   ```

Điều này chọn các giá trị từ cột A của bảng P mà không có trong tập hợp các giá trị của cột B từ bảng R. Trong trường hợp bảng R hoặc P rỗng, câu lệnh này sẽ trả về tất cả các giá trị từ cột A của bảng P.

Vì vậy, khi bảng R hoặc P là rỗng, kết quả của hai câu lệnh sẽ khác nhau. Câu lệnh đầu tiên sẽ trả về một tập hợp rỗng, trong khi câu lệnh thứ hai sẽ trả về tất cả các giá trị từ cột A của bảng P.

7 Bài 7.

Given the following relations: R(A,B), S(B,C,D), T(D,E):
- Explain the result of the following RA expression in **plain English**.
$\pi_A(R) - \pi_{R2.A}(\sigma_{R2.A<R.A}(R * \rho_{R2}(R)))$
- Write a relational algebra expression to generate a relation RESULT(A,D,E), that is a natural join of R, S, and T, followed by a projection.
- Write a relational algebra expression to find those E values that do not appear in the above RESULT relation.

1. **Explanation of the RA Expression:**

   The given expression involves several relational algebra operations. Let's break it down step by step:

   - $\rho_{R2}(R)$: Renames relation R as R2.
   - $R * \rho_{R2}(R)$: Performs a Cartesian product between R and the renamed R2.
   - $\sigma_{R2.A<R.A}(R * \rho_{R2}(R))$: Selects tuples from the Cartesian product where the value of A in R2 is less than the value of A in R.
   - $\pi_{R2.A}(\sigma_{R2.A<R.A}(R * \rho_{R2}(R)))$: Projects only the A attribute from the selected tuples.
   - $\pi_A(R)$: Projects only the A attribute from relation R.
   - $\pi_A(R) - \pi_{R2.A}(\sigma_{R2.A<R.A}(R * \rho_{R2}(R)))$: Performs set difference between the result of the first projection and the result of the second projection.

   In plain English, the result of this expression is the set of values of attribute A from relation R, minus the set of values of attribute A from relation R, where there exists a tuple in R2 with a lesser value of attribute A.

2. **Relational Algebra Expression for RESULT(A,D,E):**

   The relational algebra expression to generate the relation RESULT(A,D,E) can be written as follows:

   $\pi_{R.A,S.D,T.E}((R \bowtie S) \bowtie T)$

   This expression performs a natural join between relations R and S, followed by a natural join with relation T. Then, it projects attributes A from R, D from S, and E from T to generate the RESULT relation.

3. **Relational Algebra Expression to Find E values not in RESULT:**

   The relational algebra expression to find those E values that do not appear in the RESULT relation can be written as follows:

   $\pi_{T.E}(T - (R \bowtie S \bowtie T))$

This expression first performs a natural join between relations R, S, and T, then subtracts this result from relation T, and finally projects attribute E to retrieve those E values that do not appear in the RESULT relation.

8 Bài 8.

Consider the division operation again on relations R(A,B) and S(B). You are to write R / S using SQL. As a starting point, here is the relational algebra expression for / using other operators:

- $R/S = \pi_A(R) - \pi_A((\pi_A(R) * S) - R)$

To implement the division operation R / S using SQL, we can follow the provided relational algebra expression. Here's how we can write it in SQL:

```
SELECT DISTINCT R.A
FROM R
WHERE R.A NOT IN (
    SELECT DISTINCT R1.A
    FROM R AS R1
    WHERE NOT EXISTS (
        SELECT *
        FROM S
        WHERE S.B = R1.B
        )
    );
```

Explanation: - We first select all distinct values of attribute A from relation R. - Then, we exclude those values of A from R for which there exist no corresponding tuples in S with matching values of B. - The NOT IN clause ensures that only those values of A from R are selected which are not associated with every value of B in S.