

QUẢN LÝ GIAO TÁC VÀ ĐIỀU KHIỂN ĐỒNG THỜI PHÂN TÁN

Outline

Outline.....	1
Content.....	2
1. Quản lý giao tác phân tán	2
1.1. Định nghĩa giao tác (transaction)	2
1.2. Tính chất của giao tác	2
1.2.1. Atomic – Tính nguyên tố.....	3
1.2.2. Consistency – Tính nhất quán.....	3
1.2.3. Isolation – Tính cô lập	4
1.2.4. Durability – Tính lâu dài, bền vững.....	4
1.3. Kiểm soát giao tác.....	5
2. Điều khiển đồng thời.....	6
2.1. Điều khiển đồng thời phân tán là gì	6
2.1.1. Nguyên lý điều khiển đồng thời	7
2.1.2. Điều khiển đồng thời phân tán.....	7
2.2. Các vấn đề xảy ra khi điều khiển đồng thời.....	7
2.2.1. Mất dữ liệu cập nhật (Lost update)	7
2.2.2. Đọc dữ liệu rác (Dirty read // Uncommitted data)	8
2.2.3. Thao tác đọc không thể lặp lại (Unrepeatable data)	8
2.2.3. Bóng ma (Phantom reads)	9
2.3. Một số tính chất khi thao tác trên đơn vị dữ liệu	10
2.4. Lịch tuần tự và lịch khả tuần tự.....	11
2.5. Sắp xếp các giao tác bằng nhãn thời gian	11
3. Phân biệt với quản lý giao tác và điều khiển đồng thời trong môi trường tập trung .	12
3.1. Giao tác tập trung.....	12
3.1.1. Giao tác phẳng (flat transaction)	12
3.1.2. Giao tác lồng nhau (nested transaction)	12
3.2. Giao tác phân tán	12

3.3. So Sánh Điều khiển đồng thời trong csdl tập trung và Điều khiển đồng thời trong csdl phân tán (SANG).....	12
--	----

Content

1. Quản lý giao tác phân tán

1.1. Định nghĩa giao tác (transaction)

Giao tác (transaction) là một tập hợp các **thao tác** có thứ tự (statement) truy xuất dữ liệu trên CSDL thành một đơn vị công việc logic (xem là 1 thao tác nguyên tố), chuyển CSDL từ trạng thái nhất này sang trạng thái nhất quán khác.

>> Hai thao tác cơ sở:

- Đọc dữ liệu từ CSDL: read(x)
- Ghi dữ liệu vào CSDL: write(x)

Minh họa:

CSDL nhất quán 1 à **Giao tác** à CSDL nhất quán 2.

Ví dụ:

```
(T1): Begin
    read(a);
    a:=a+100;
    read(a);
    a:=a+2;
    write(a);
end.
```

1.2. Tính chất của giao tác

Tính chất của giao tác: **ACID**

1. Atomic – Tính nguyên tố: Không thể chia nhỏ. Tất cả các nhiệm vụ của một giao tác được thực hiện hoặc không có nhiệm vụ nào trong số chúng. Không có giao tác từng phần.

2. Consistency – Tính nhất quán: Chuyển CSDL từ trạng thái nhất quán này sang trạng thái nhất quán khác. (Một giao tác được thực hiện độc lập với các giao tác khác xử lý đồng thời với nó để bảo đảm tính nhất quán cho CSDL)
3. Isolation – Tính cô lập: Các giao tác xử lý đồng thời phải độc lập với những thay đổi của giao tác khác.
4. Durability – Tính lâu dài, bền vững: Khi giao tác hoàn tất, tất cả thay đổi phải được ghi nhận bền vững lên CSDL.

1.2.1. Atomic – Tính nguyên tử

Tính nguyên tử của một giao tác là sự thực hiện trọn vẹn mà không một giao tác nào được chen vào.

Khi thực thi một giao tác thì hoặc là các hành động của giao tác đó được thực hiện hoặc là không một hành động nào được thực hiện cả.

Tính nguyên tử đòi hỏi rằng nếu việc thực thi giao tác bị cắt ngang bởi một loại sự cố nào đó thì DBMS sẽ chịu trách nhiệm xác định những công việc của giao tác để **khôi phục lại sau sự cố**.

>> Có 2 chiều hướng thực hiện:

- Nó sẽ được kết thúc bằng cách hoàn tất các hành động còn lại,
- Hoặc có thể kết thúc bằng cách hồi lại tất cả các hành động đã được thực hiện.

Ví dụ, nếu một giao dịch bắt đầu cập nhật 100 hàng, nhưng hệ thống không thành công sau 20 lần cập nhật, thì cơ sở dữ liệu sẽ khôi phục các thay đổi cho 20 hàng này.

1.2.2. Consistency – Tính nhất quán

Tính nhất quán của một giao tác chỉ đơn giản là tính đúng đắn của nó.

Nói cách khác, một giao tác là một chương trình đúng đắn, ánh xạ cơ sở dữ liệu từ trạng thái nhất quán này sang một trạng thái nhất quán khác.

Dữ liệu rác (dirty data) muốn nói đến những giá trị dữ liệu đã được cập nhật bởi một giao tác trước khi nó ủy thác ¹.

¹ Ủy thác (commitment): Sự hoàn thành một giao tác.

Ví dụ, trong một giao dịch ngân hàng ghi nợ tài khoản tiết kiệm và ghi có tài khoản séc, lỗi không được làm cho cơ sở dữ liệu chỉ ghi có một tài khoản, điều này sẽ dẫn đến dữ liệu không nhất quán.

1.2.3. Isolation – Tính cô lập

Tính chất này để ngăn ngừa sự hủy bỏ dây chuyền (cascading abort - Còn gọi là hiệu ứng domino).

>> Một giao tác đang thực thi không thể đưa ra các kết quả của nó cho những giao tác khác đang cùng hoạt động trước khi nó uý thác.

1.2.4. Durability – Tính lâu dài, bền vững

Để bảo đảm rằng mỗi khi giao tác uý thác, kết quả của nó sẽ được duy trì và không bị xoá ra khỏi CSDL.

DDBMS có trách nhiệm bảo đảm kết quả của giao tác và ghi vào CSDL.

Tính bền vững được sử dụng như là một điều kiện để khôi phục dữ liệu (database recovery), nghĩa là cách khôi phục CSDL về trạng thái nhất quán mà ở đó mọi hành động đã uý thác đều được phản ánh.

Ví dụ:

```
T: Read (A, t) ;  
   t:=t-50 ;  
   Write (A, t) ;  
   Read (B, t) ;  
   t:=t+50 ;  
   Write (B, t) ;
```

Atomic – Tính nguyên tố

– $A = 100, B = 200$ ($A + B = 300$)

– Tại thời điểm sau khi write (A, t)

- $A=50, B=200$ ($A+B=250$) - CSDL không nhất quán

– Tại thời điểm sau khi write (B, t)

- $A=50, B=250$ ($A+B=300$) - CSDL nhất quán

– Nếu T không bao giờ bắt đầu thực hiện hoặc T được đảm bảo phải hoàn tất thì trạng thái không nhất quán sẽ không xuất hiện

Consistency – Tính nhất quán

- Tổng $A + B$ là không đổi
- Nếu CSDL nhất quán trước khi T được thực hiện thì sau khi T hoàn tất CSDL vẫn còn nhất quán

Isolation – Tính cô lập

```

      T: Read(A, t);
        t:=t-50;
T' →  Write(A, t);
      Read(B, t);
        t:=t+50;
      Write(B, t);

```

- Giả sử có 1 giao tác T' thực hiện phép toán $A + B$ và chen vào giữa thời gian thực hiện của T
- T' kết thúc: $A + B = 50 + 200 = 250$
- T kết thúc: $A + B = 50 + 250 = 300$
- Hệ thống của các giao tác thực hiện đồng thời có trạng thái tương đương với trạng thái hệ thống của các giao tác thực hiện tuần tự theo 1 thứ tự nào đó.

Durability – Tính lâu dài, bền vững

- Khi T kết thúc thành công
- Dữ liệu sẽ không thể nào bị mất bất chấp có sự cố hệ thống xảy ra

1.3. Kiểm soát giao tác

Kiểm soát giao tác là quản lý các thay đổi được thực hiện bởi các câu lệnh DML và nhóm các câu lệnh DML thành các giao tác. Nói chung, việc kiểm soát giao tác mục đích là để công việc được hoàn thành theo các đơn vị logic và dữ liệu được giữ nhất quán.

Kiểm soát giao tác liên quan đến việc sử dụng các câu lệnh sau:

Bắt đầu giao tác	SET TRANSACTION + READ ONLY/ READ WRITE;
Kết thúc một giao tác thành công	COMMIT;
Kết thúc một giao tác không thành công, những thao tác làm ảnh hưởng CSDL trước đó được undo, CSDL được trả về tình trạng trước khi thực hiện giao tác.	ROLLBACK;

Lưu ý:

Trong khai báo giao tác, phải chỉ định rõ loại giao tác là read only hay read write.

Read only (transaction – level read consistency) giao tác chỉ thấy được những thay đổi dữ liệu được commit trước khi nó bắt đầu. Chỉ các câu lệnh sau được phép sử dụng trong loại giao tác này: select ... into, open, fetch, close, lock table, commit và rollback.

Read write (statement – level read consistency) các thao tác trong giao tác thấy được những thay đổi dữ liệu được commit trước khi thao tác bắt đầu. Bản thân giao tác thì vẫn thấy được những thay đổi do chính nó thực hiện. Cho phép đọc và thay đổi dữ liệu trong giao tác. Các câu lệnh: insert, update, delete được gọi thực hiện và giao tác đó sẽ thấy được các thay đổi này.

Ví dụ:

READ ONLY	READ WRITE
<pre>SET SERVEROUT ON; DECLARE Count INTEGER; BEGIN SET TRANSACTION READ ONLY; SELECT COUNT(*) INTO Count FROM ACCOUNT; DBMS_OUTPUT.PUT_LINE('Số tài khoản hiện có là: ' Count); COMMIT; END;</pre>	<pre>SET SERVEROUT ON; DECLARE Temp INTEGER; BEGIN SET TRANSACTION READ WRITE; Temp = 500; UPDATE ACCOUNT SET Balance = Temp Where AccID = '001'; COMMIT; END;</pre>

2. Điều khiển đồng thời

2.1. Điều khiển đồng thời phân tán là gì

2.1.1. Nguyên lý điều khiển đồng thời

Là quá trình điều khiển giúp cho nhiều giao tác diễn ra đồng thời mà không xảy ra tranh chấp.

2.1.2. Điều khiển đồng thời phân tán

Nhằm ngăn chặn việc sản sinh ra các thực hiện không khả tuần tự của các giao tác phân tán. Đảm bảo nhiều giao tác thực hiện đồng thời mà vẫn đảm bảo tính đúng đắn trên CSDL.

2.2. Các vấn đề xảy ra khi điều khiển đồng thời

2.2.1. Mất dữ liệu cập nhật (Lost update)

Tình trạng này xảy ra khi có nhiều hơn một giao tác cùng thực hiện cập nhật trên 1 đơn vị dữ liệu. Khi đó, tác dụng của giao tác cập nhật thực hiện sau sẽ đè lên tác dụng của thao tác cập nhật trước.

Ví dụ:

T1		T2	
BEGIN			BEGIN
Read (A)		A = 1	
		A = 1	Read (A)
A := A + 10	à	A = 11	
Write (A)	à	A = 11	
		A = 110	β A := A * 10
		A = 110	β Write (A)
		A = 110	COMMIT
COMMIT		A = 110	

Vấn đề: T1 đang thực hiện các cập nhật nhưng T2 lại thực hiện ghi đè lên dữ liệu được ghi bởi T1 à Thao tác cập nhật của T1 xem như bị mất, không được ghi nhận.

2.2.2. Đọc dữ liệu rác² (Dirty read // Uncommitted data)

Xảy ra khi một giao tác thực hiện đọc trên một đơn vị dữ liệu mà đơn vị dữ liệu này đang bị cập nhật bởi một giao tác khác nhưng việc cập nhật chưa được xác nhận.

Ví dụ:

T1		T2		
BEGIN				
Read (A)		A = 1		
A := A + 10	à	A = 11		
Write (A)	à	A = 11		
		A = 11		BEGIN
		A = 11		Read (A)
		A = 11		COMMIT
ROLLBACK	à	A = 1		

Vấn đề: T1 sửa đổi A nhưng chưa commit, T2 đọc dữ liệu được ghi bởi T1, nhưng sau đó T1 lại huỷ việc ghi.

2.2.3. Thao tác đọc không thể lặp lại (Unrepeatable data)

Tình trạng này xảy ra khi một giao tác T1 vừa thực hiện xong thao tác đọc trên một đơn vị dữ liệu (nhưng chưa commit) thì giao tác khác (T2) lại thay đổi (ghi) trên đơn vị dữ liệu này. Điều này làm cho lần đọc sau đó của T1 không còn nhìn thấy dữ liệu ban đầu nữa.

Ví dụ:

T1		T2		
BEGIN				
Read (A)	T1	A = 1		
		A = 1		BEGIN
		A = 1		Read (A)

² Dữ liệu rác (dirty data) muốn nói đến những giá trị dữ liệu đã được cập nhật bởi một giao tác trước khi nó ủy thác.

		A = 11	β	A := A + 10
		A = 11	β	Write (A) // Update / Delete
		A = 11		COMMIT
Read (A)	T2	A = 11		
COMMIT		A = 11		

Vấn đề: Transaction 1 thực hiện xong thao tác đọc trên A tại thời điểm T1 cho giá trị A = 1. Transaction 2 thay đổi giá trị của A thành 11. Điều này làm cho vào thời điểm T2, Transaction A không còn nhìn thấy dữ liệu ban đầu nữa do đã bị **cập nhật hoặc xóa** đi.

2.2.3. Bóng ma (Phantom reads)

Tình trạng mà một giao tác đang thao tác trên một tập dữ liệu nhưng giao tác khác lại chèn thêm các dòng dữ liệu vào tập dữ liệu mà giao tác kia quan tâm.

Ví dụ:

T1		T2		
BEGIN				
SELECT * FROM SV	T1	SV(EMPTY)		
		SV(EMPTY)		BEGIN
		SV(A)	β	INSERT INTO SV VALUES (A)
		SV(A)		COMMIT
SELECT * FROM SV	T2	SV(A)		
COMMIT		SV(A)		





Vấn đề: Transaction 1 thực hiện truy vấn bảng Sinh Viên tại thời điểm T1. Lúc này cho kết quả trống, do bảng Sinh Viên không có dữ liệu này. Transaction 2 lại thực hiện thêm thông tin A vào bảng Sinh Viên. Transaction 1 thực thi lại câu truy vấn tại thời điểm T2 cho kết quả có thông tin A vì tập dữ liệu đang sử dụng không còn chính xác do có dữ liệu đã bị **thêm** vào.

à Hai lần đọc dữ liệu của Transaction A cho kết quả khác nhau. (trong cùng một giao tác).

Sự khác nhau giữa Phantom read và Unrepeatable data là: dữ liệu đã đọc không bị thay đổi, nhưng thay vào đó, xuất hiện một hay nhiều dữ liệu khác đáp ứng các tiêu chí truy vấn hơn trước.





2.3. Một số tính chất khi thao tác trên đơn vị dữ liệu

- Hai thao tác tương thích
 - Hai thao tác O_i và O_j ($O_i \in T_i$, $O_j \in T_j$) gọi là tương thích Nếu và chỉ nếu:
 - Kết quả của việc thực hiện đồng thời O_i và O_j giống như kết quả của việc thực

	T_1	T_2		
O_{11}	Read $A \rightarrow a_1$ $a_1 + 1 \rightarrow a_1$ Print a_1	Read $A \rightarrow a_2$ $a_2 * 2 \rightarrow a_2$ Print a_2	O_{21}	 O_{11} và O_{21} là tương thích
O_{12}	Read $A \rightarrow a_1$ $a_1 + 5 \rightarrow a_1$ Write $a_1 \rightarrow A$	Read $A \rightarrow a_2$ $a_2 * 2 \rightarrow a_2$ Write $a_2 \rightarrow A$	O_{22}	 O_{12} và O_{22} không tương thích
O_{13}	Read $A \rightarrow a_1$ $a_1 + 2 \rightarrow a_1$ Write $a_1 \rightarrow A$	Read $A \rightarrow a_2$ $a_2 + 7 \rightarrow a_2$ Write $a_2 \rightarrow A$	O_{23}	 O_{13} và O_{23} không tương thích
O_{14}	Read $B \rightarrow b_1$ $b_1 + 1 \rightarrow b_1$ Write $b_1 \rightarrow B$			 O_{14} tương thích với các thao tác còn lại

hiện tuần tự O_i rồi đến O_j (hoặc O_j rồi đến O_i).

- Hai thao tác khả hoán vị
 - Hai thao tác O_i và O_j (O_i thuộc T_i , O_j thuộc T_j) là khả hoán vị Nếu: kết quả thực

	T_1	T_2		
O_{11}	Read $A \rightarrow a_1$ $a_1 + 1 \rightarrow a_1$ Print a_1	Read $A \rightarrow a_2$ $a_2 * 2 \rightarrow a_2$ Print a_2	O_{21}	 O_{11} và O_{21} là khả hoán vị
O_{12}	Read $A \rightarrow a_1$ $a_1 + 5 \rightarrow a_1$ Write $a_1 \rightarrow A$	Read $A \rightarrow a_2$ $a_2 * 2 \rightarrow a_2$ Write $a_2 \rightarrow A$	O_{22}	 O_{12} và O_{22} không khả hoán vị
O_{13}	Read $A \rightarrow a_1$ $a_1 + 2 \rightarrow a_1$ Write $a_1 \rightarrow A$	Read $A \rightarrow a_2$ $a_2 + 7 \rightarrow a_2$ Write $a_2 \rightarrow A$	O_{23}	 O_{13} và O_{23} là khả hoán vị
O_{14}	Read $B \rightarrow b_1$ $b_1 + 1 \rightarrow b_1$ Write $b_1 \rightarrow B$			 O_{14} khả hoán vị với các thao tác còn lại

hiện Oi ,Oj hay Oj, Oi là như nhau.

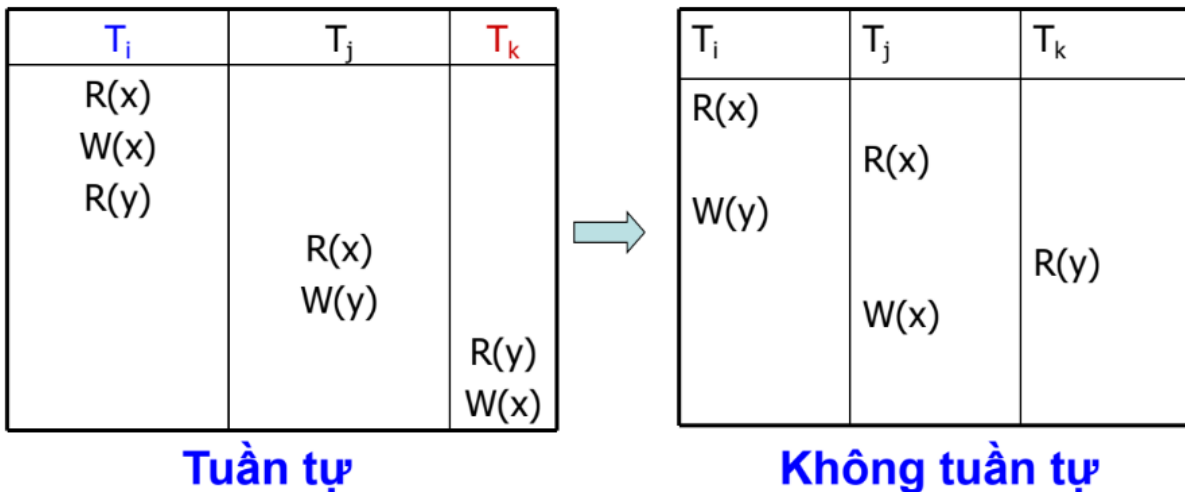
- **Nhận xét**

- Các thao tác truy xuất các đơn vị dữ liệu khác nhau là tương thích và khả hoán vị
- Các thao tác truy xuất trên cùng đơn vị dữ liệu:
 - Nếu có liên quan đến phép cộng, trừ thì khả hoán vị
 - Read – Read → khả hoán vị
 - Write – Write, Read – Write và Write – Read → không có tính khả hoán vị
- Chú ý: Hai thao tác không khả hoán vị thì gọi là xung đột

2.4. Lịch tuần tự và lịch khả tuần tự

- **Lịch tuần tự (serial schedule)**

- Một lịch S được lập từ n giao tác T1, T2,...,Tn xử lí đồng thời gọi là lịch tuần tự nếu các thao tác của từng giao tác được thực hiện liên tiếp nhau.



- **Lịch khả tuần tự (serializable schedule)**

- Một lịch S được lập từ n giao tác T1, T2,...,Tn xử lí đồng thời gọi là lịch khả tuần tự nếu nó cho cùng kết quả với một lịch tuần tự được lập từ n giao tác trên.

- **Như vậy:**

- Tính khả tuần tự của các giao tác là điều kiện đủ để tránh độn độ trong việc truy xuất đồng thời (Một lịch nếu khả tuần tự thì không độn độ, nếu không có khả tuần tự thì chưa chắc độn độ)
- Bộ lập lịch (schedule): Là một bộ phận của DBMS chịu trách nhiệm lập lịch khả tuần tự từ n giao tác xử lí đồng thời, sẽ tiến hành lập lịch các thao tác (thao tác nào sẽ được thực hiện trước, thao tác nào sẽ được thực hiện sau).

2.5. Sắp xếp các giao tác bằng nhãn thời gian

- **Khái niệm:**

- Nhãn thời gian (timestamp):
 - Là 1 con số được phát sinh bởi bộ lập lịch, được gán cho mỗi giao tác để chỉ định thời điểm bắt đầu thực hiện giao tác.
 - Nhãn thời gian có tính chất duy nhất và tăng dần.
- Nhãn thời gian của đơn vị dữ liệu:

- Là nhãn thời gian của giao tác cuối cùng truy cập thành công đến đơn vị dữ liệu đó, hay là nhãn thời gian cao nhất trong số các giao tác có truy cập thành công đến đơn vị dữ liệu đó

3. Phân biệt với quản lý giao tác và điều khiển đồng thời trong môi trường tập trung

3.1. Giao tác tập trung

3.1.1. Giao tác phẳng (flat transaction)

Giao tác phẳng (flat transaction) có một khởi điểm duy nhất (Begin transaction) và một điểm kết thúc duy nhất (End transaction).

Tất cả các ví dụ của chúng ta đã xem xét đều nằm trong nhóm này.

Phần lớn các nghiên cứu về quản lý giao tác trong cơ sở dữ liệu đều tập trung vào các giao tác phẳng.

3.1.2. Giao tác lồng nhau (nested transaction)

Đây là mô hình giao tác cho phép một giao tác chứa giao tác khác với điểm bắt đầu và ủy thác của riêng chúng. Những giao tác như thế được gọi là giao tác lồng (nested transaction).

Những giao tác được đặt vào trong giao tác khác thường được gọi là giao tác con (subtransaction)

3.2. Giao tác phân tán

Chỉ có một loại giao tác duy nhất là giao tác phẳng.

Vì vậy sẽ không có save transaction. (vì save transaction trong giao tác lồng)

3.3. So Sánh Điều khiển đồng thời trong csdl tập trung và Điều khiển đồng thời trong csdl phân tán (SANG)

CSDL Tập Trung	CSDL Phân Tán
Tính điều khiển đồng thời không quan trọng vì người dùng có thể sửa đổi dữ liệu trong cơ sở dữ liệu mà không cần quan tâm đến việc những người dùng khác sửa	Tính điều khiển đồng thời rất quan trọng vì nó kiểm soát tính đồng thời của dữ liệu và tính nhất quán của dữ liệu

đổi cùng một dữ liệu cùng một lúc	
Việc quản lý, sửa đổi, nâng cấp dữ liệu cũng dễ dàng vì nó chỉ có 1 file database	Việc quản lý, sửa đổi, nâng cấp dữ liệu để cho mọi vị trí đồng bộ dữ liệu với nhau sẽ tốn khá nhiều thời gian của người quản trị
Do cùng trên 1 cơ sở dữ liệu nên việc nhiều người dùng truy cập vào dữ liệu sẽ làm cho tốc độ xử lý của hệ thống chậm đi.	Việc nhiều người dùng truy cập vào dữ liệu sẽ không làm cho tốc độ truy cập chậm đi vì khi người dùng thực hiện thao tác gì đó nó sẽ tự động đồng bộ với những nơi khác làm giảm tải việc xử lý của server.
Vì có 1 cơ sở dữ liệu nên nếu database bị trục trặc thì người dùng sẽ không thể truy cập được	Vì có nhiều vị trí hiển thị cùng 1 kết quả cơ sở dữ liệu đó nên nếu một nơi bị trục trặc database thì người dùng có thể truy cập ở vị trí khác mà không lo mất dữ liệu, sai lệch kết quả.