Chương 5 GIỚI THIỆU VỀ GIAO TÁC

MỤC TIÊU

Chương này giới thiệu khái niệm giao tác và định nghĩa hình thức của giao tác. Chương này chia làm ba phần:

1. Phần thứ nhất: Khái niệm giao tác, định nghĩa hình thức của giao tác.
2. Phần thứ hai: Các tính chất của giao tác
3. Phần thứ ba: Phân loại giao tác

MỞ ĐẦU

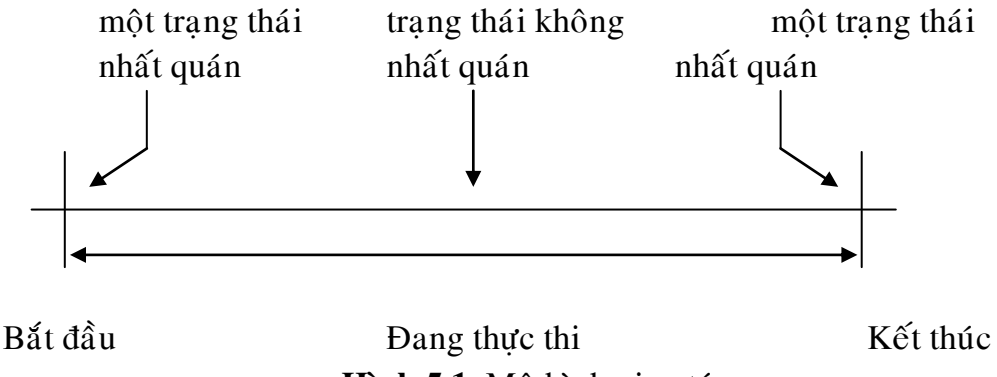
Cho đến lúc này, đơn vị truy xuất cơ bản là câu truy vấn. Trong các chương 4, chúng ta đã thảo luận về cách xử lý và tối ưu hoá các truy vấn. Tuy nhiên chúng ta chưa bao giờ xét đến các tình huống xảy ra, chẳng hạn khi hai câu truy vấn cùng cập nhật một mục dữ liệu, hoặc tình huống hệ thống bị sự cố phải ngừng hoạt động trong khi đang thực hiện câu truy vấn. Đối với những câu truy vấn chỉ truy xuất, không có tình huống nào ở trên gây rắc rối. Người ta có thể cho hai câu truy vấn đọc dữ liệu cùng một lúc. Tương tự, sau khi đã xử lý xong sự cố, các truy vấn chỉ đọc chỉ cần khởi động lại. Nhưng ngược lại có thể nhận ra rằng đối với những câu truy vấn cập nhật, những tình huống này có thể gây ra những tổn hại nghiêm trọng cho cơ sở dữ liệu. Như chúng ta không thể chỉ khởi động lại cho câu truy vấn cập nhật sau một sự cố hệ thống vì một số giá trị của các mục dữ liệu có thể đã được cập nhật trước khi có sự cố xảy ra và không cho phép cập nhật lại khi câu truy vấn được khởi động lại, nếu không thì cơ sở dữ liệu sẽ chứa những dữ liệu sai lệch.

Điểm mấu chốt ở đây là không có khái niệm "thực thi nhất quán" hoặc "tính toán đáng tin cậy" đi kèm với khái niệm truy vấn. Khái niệm *giao tác* (transaction) được sử dụng trong lãnh vực cơ sở dữ liệu như một đơn vị tính toán

nhất quán và tin cậy được. Vì thế các câu truy vấn sẽ được thực thi như các giao tác một khi các chiến lược thực thi được xác định và được dịch thành các thao tác cơ sở dữ liệu nguyên thủy.

Trong thảo luận ở trên chúng ta đã dùng thuật ngữ *"nhất quán"* (consistent) và *"đáng tin cậy"* (reliable) một cách hoàn toàn không hình thức. Thế nhưng do tầm quan trọng của chúng mà chúng ta cần phải định nghĩa chúng một cách chuẩn xác. Trước tiên phải chỉ ra rằng cần phân biệt giữa *nhất quán* cơ sở dữ liệu (database consistency) và *nhất quán giao tác* (transaction consistency).

Một cơ sở dữ liệu ở trong một *trạng thái nhất quán* (consistent state) nếu nó tuân theo tất cả các ràng buộc toàn vẹn (nhất quán) được định nghĩa trên nó. Dĩ nhiên chúng ta cần bảo đảm rằng cơ sở dữ liệu không bao giờ chuyển sang một trạng thái không nhất quán. Cơ sở dữ liệu có thể tạm thời không nhất quán trong khi thực hiện giao tác. Điều quan trọng là cơ sở dữ liệu phải trở về trạng thái nhất quán khi quan hệ giao tác chấm dứt.



Hình 5.1. Mô hình giao tác.

Ngược lại, tính nhất quán giao tác muốn nói đến hành động của các giao tác đồng thời. Chúng ta mong rằng cơ sở dữ liệu vẫn nhất quán ngay cả khi có một số yêu cầu của người sử dụng đồng thời truy xuất đến cơ sở dữ liệu (đọc hoặc cập nhật). Tính chất phức tạp nảy sinh khi xét đến các cơ sở dữ liệu có nhân bản. Một cơ sở dữ liệu được nhân bản ở trong một *trạng thái nhất quán lẫn nhau* (mutually consistent state) nếu tất cả các bản sao của mỗi mục dữ liệu ở trong đó đều có giá trị giống nhau. Điều này thường được gọi là *sự tương đương một bản* (one copy equivalence) vì tất cả các bản đều bị buộc phải nhận cùng một trạng thái vào cuối lúc thực thi giao tác. Một số khái niệm về tính nhất quán bản sao cho phép giá trị các bản sao có thể khác nhau. Những vấn đề này sẽ được thảo luận sau.

Độ *tin cậy* hay *khả tín* (reliability) muốn nói đến khả năng *tự thích ứng* (resistency) của một hệ thống đối với các loại sự cố và khả năng khôi phục lại từ những sự cố này. Một hệ thống khả tín sẽ tự thích ứng với các sự cố hệ thống và có thể tiếp tục cung cấp các dịch vụ ngay cả khi xảy ra sự cố.

Một hệ quản trị cơ sở dữ liệu khả hồi phục là hệ quản trị cơ sở dữ liệu có thể chuyển sang trạng thái nhất quán (bằng cách quay trở lại trạng thái nhất quán trước đó hoặc chuyển sang một trạng thái nhất quán mới) sau khi gặp một sự cố.

Quản lý giao tác (transaction management) là ***giải quyết các bài toán duy trì được cơ sở dữ liệu ở trong tình trạng nhất quán ngay cả khi có nhiều truy xuất đồng thời và khi có sự cố***.

Mục đích của chương này là định nghĩa những thuật ngữ cơ bản và đưa ra một bộ khung trên cơ sở đó để thảo luận các vấn đề này. Đây cũng là phần giới thiệu ngắn gọn về bài toán cần giải quyết và các vấn đề có liên quan. Vì thế chúng ta sẽ trình bày các khái niệm ở một mức trừu tượng khá cao và không trình bày những kỹ thuật quản lý. Trong phần tiếp theo chúng ta sẽ định nghĩa một cách hình thức và một cách trực quan về khái niệm giao tác.

5.1 ĐỊNH NGHĨA GIAO TÁC:

Giao tác được xem như một dãy các thao tác đọc và ghi trên cơ sở dữ liệu cùng với các bước tính toán cần thiết.

Thí dụ 5.1

Xét câu truy vấn SQL làm tăng ngân sách của dự án CAD/CAM lên 10%

```
UPDATE PROJ
SET     BUTGET = BUTGET * 1.1
WHERE   PNAME = "CAD/CAM"
```

Câu truy vấn này có thể được đặc tả, qua ký pháp SQL gắn kết, như một giao tác bằng cách cho nó một tên (thí dụ BUDGET_UPDATE) và khai báo như sau:

```
Begin transaction BUDGET_UPDATE
begin
    UPDATE PROJ
        SET  BUTGET = BUTGET * 1.1
        WHERE  PNAME = "CAD/CAM"
end
```

Các câu lệnh **Begin transaction** và **end** ấn định ranh giới một giao tác. Chú ý rằng việc sử dụng các ký hiệu phân cách này hoàn toàn không bắt buộc trong mọi hệ quản trị cơ sở dữ liệu. Chẳng hạn nếu các dấu phân cách không được đặc tả, DB2 sẽ xử lý toàn bộ chương trình thực hiện truy xuất cơ sở dữ liệu như một giao tác.

Thí dụ 5.2

Trong phần thảo luận về các khái niệm quản lý giao tác, chúng ta sẽ sử dụng thí dụ về một hệ thống đặt chỗ máy bay. Cài đặt thực tế của ứng dụng này

luôn sử dụng đến khái niệm giao tác. Chúng ta giả sử rằng có một quan hệ **FLIGHT** ghi nhận dữ liệu về các *chuyến bay* (flight), quan hệ **CUST** cho các khách hàng có đặt chỗ trước và quan hệ **FC** cho biết khách hàng nào sẽ đi trên chuyến bay nào. Chúng ta cũng giả sử rằng các định nghĩa quan hệ sau (thuộc tính gạch dưới biểu thị khoá):

FLIGHT (FNO, DATE, SRC, DEST, STSOLD, CAP)
CUST (CNAME, ADDR, BAL)
FC (FNO, CNAME, SPECIAL)

Định nghĩa thuộc tính trong lược đồ này như sau: **FNO** là mã số chuyến bay, **DATE** biểu ngày tháng chuyến bay, **STSOLD** chỉ số lượng ghế (seat) đã được bán trên chuyến bay đó, **CAP** chỉ sức chuyên chở (số lượng hành khách có thể chở được, capacity) trên chuyến bay, **CNAME** chỉ tên khách hàng với địa chỉ được lưu trong **ADDR** và số dư trong **BAL**, còn **SPECIAL** tương ứng với các yêu cầu đặc biệt mà khách hàng đưa ra khi đặt chỗ.

Chúng ta xét một phiên bản đơn giản hoá của một ứng dụng đặt chỗ, trong đó một nhân viên bán vé nhập mã số chuyến bay, ngày tháng, tên khách hàng và thực hiện đặt chỗ trước. Giao tác thực hiện công việc này có thể được cài đặt như sau, trong đó các truy xuất cơ sở dữ liệu được đặc tả bằng SQL:

```
Begin transaction Reservation -- đặt chỗ
begin
   input (flight_no, date, customer_name);
       UPDATE FLIGHT
              SET STSOLD = STSOLD +1
              WHERE FNO = flight_no
        INSERT
              INTO FC (FNO, CNAME, SPECIAL)
              VALUES (flight_no,  customer_name, null);
   output (“reservation completed”)
end
```

5.1.1 Tình huống kết thúc giao tác

Một giao tác luôn luôn phải kết thúc ngay cả khi có xảy ra sự cố. Nếu giao tác có thể hoàn tất thành công tác vụ của nó, chúng ta nói rằng giao tác có *uỷ thác* (commit). Ngược lại nếu một giao tác phải ngừng lại khi chưa hoàn tất công việc, chúng ta nói rằng nó *bị hủy bỏ* (abort). Một giao tác phải tự hủy bỏ vì có một điều kiện làm cho nó không hoàn tất được công việc. Ngoài ra hệ quản trị cơ sở dữ liệu có thể hủy bỏ một giao tác, chẳng hạn do bị *khoá chết* (deadlock). Khi một giao tác bị hủy bỏ, quá trình thực thi sẽ ngừng và tất cả mọi hành động

đã được thực hiện đều phải được "undo", đưa cơ sở dữ liệu trở về trạng thái trước khi thực hiện giao tác. Quá trình này gọi là "rollback".

Vai trò quan trọng của ủy thác biểu hiện ở hai mặt. **Thứ nhất** lệnh ủy thác báo cho hệ quản trị cơ sở dữ liệu biết rằng tác dụng của giao tác đó bây giờ cần được phản ảnh vào cơ sở dữ liệu, qua đó làm cho *các giao tác đang truy xuất các mục dữ liệu đó có thể thấy được chúng*. **Thứ hai**, điểm mà giao tác ủy thác là một *điểm "không đường về"*. Kết quả của một giao tác đã ủy thác bây giờ được lưu cố định vào cơ sở dữ liệu và không thể phục hồi lại được.

Thí dụ 5.3

Một điều chúng ta chưa xét đến là tình huống không còn chỗ trống trên chuyến bay. Để bao quát khả năng này, giao tác cần được viết lại như sau:

```
Begin_transaction Reservation
begin
  input(flight_no, date, customer_name);
  SELECT STSOLD, CAP
            INTO temp1, temp2
            FROM FLIGHT
            WHERE FNO = flight_no

  if temp1 = temp2 then
  begin
      Output("no free seat");
      Abort
  end
  else begin
      UPDATE FLIGHT
                  SET STSOLD = STSOLD +1
                  WHERE FNO = flight_no

      INSERT
                  INTO FC(FNO, CNAME, SPECIAL)
                  VALUES(flight_no,  customer_name, null);
      Commit;
      Output("reservation completed")

      end

end.
```

Qua thí dụ này chúng ta thấy được nhiều điểm quan trọng. Rõ ràng nếu không còn chỗ trống, giao tác phải hủy bỏ. Thứ hai là việc sắp thứ tự các kết quả để trình bày ra cho người sử dụng tùy theo các lệnh **abort** và **commit**. Chú ý rằng nếu giao tác bị hủy bỏ, người sử dụng sẽ được thông báo trước khi hệ quản trị cơ sở dữ liệu được hướng dẫn để hủy bỏ nó. Thế nhưng với trường hợp ủy thác, thông báo cho người sử dụng phải xảy ra sau khi hệ quản trị cơ sở dữ liệu đã thực hiện xong lệnh ủy thác để bảo đảm độ khả tín.

5.1.2 Đặc trưng hoá các giao tác

Chúng ta nhận xét rằng các giao tác đều đọc và ghi một số dữ liệu. Điều này được dùng làm cơ sở nhận biết một giao tác. Các mục dữ liệu được giao tác đọc cấu tạo nên *tập đọc* RS (read set) của nó. Tương tự, các mục dữ liệu được một giao tác ghi được gọi là *tập ghi* WS(write set). Chú ý rằng tập đọc và tập ghi của một giao tác không nhất thiết phải tách biệt. Cuối cùng hợp của tập đọc và tập ghi của một giao tác tạo ra *tập cơ sở* BS (base set), nghĩa là BS = RS $\cup$ WS.

Thí dụ 5.4

Chúng ta xét lại giao tác đặt chỗ của thí dụ 5.3 và thao tác chèn chứa một số thao tác ghi. Các tập nêu trên được định nghĩa như sau:

RS [Revervation] = {FLIGHT.STSOLD, FLIGHT.CAP}
WS[Revervation] = {FLIGHT.STSOLD, FC.FNO,
FC.CNAME, FC.SPECIAL}
BS[Revervation] = {FLIGHT.STSOLD, FLIGHT.CAP, FC.FNO,
FC.DATE, FC.CNAME, FC.SPECIAL}

Chúng ta đã đặc trưng các giao tác chỉ trên cơ sở các thao tác đọc và ghi mà không xem xét các thao tác chèn, xoá. Như thế chúng ta đã thảo luận về khái niệm các giao tác dựa trên các cơ sở dữ liệu tĩnh, không nới rộng hoặc thu lại. Giản lược này được đưa ra để có được tính đơn giản. Các cơ sở dữ liệu động phải giải quyết bài toán *ảnh ảo* (phantom), được giải thích như ví dụ sau.

Xét giao tác T_1, khi thực hiện cần tìm trong bảng **FC** tên những khách hàng đã yêu cầu dùng một bữa ăn đặc biệt. Nó nhận được một tập **CNAME** gồm những khách hàng thỏa thuận điều kiện tìm kiếm. Khi T_1 đang thực hiện, một giao tác T_2 chèn các bộ mới vào **FC** có yêu cầu bữa ăn đặc biệt rồi ủy thác. Nếu sau đó T_1 lại đưa ra câu truy vấn tìm kiếm như cũ, nó sẽ nhận được một tập **CNAME** khác với tập ban đầu mà nó đã nhận. Vì thế các bộ "ảnh ảo" đã xuất hiện trong cơ sở dữ liệu.

5.1.3. Hình thức hoá khái niệm giao tác

Cho đến lúc này, ý nghĩa trực quan của giao tác đã hoàn toàn rõ ràng. Để bàn luận về các giao tác và suy diễn tính đúng đắn của các thuật toán quản lý giao tác, chúng ta cần định nghĩa khái niệm này một cách hình thức. Chúng ta

biểu thị *phép toán* Oj của giao tác T_i khi hoạt tác trên thực thể cơ sở dữ liệu x là $O_{ij}(x)$. Theo qui ước được thừa nhận ở phần trước, $O_{ij} \in$ {read, write}. Các phép toán được giả thiết là *nguyên tử* (nghĩa là mỗi phép toán được thực thi như một đơn vị không thể chia nhỏ được nữa). Chúng ta hãy ký hiệu OS_i là tập tất cả các phép toán trong T_i (nghĩa là $OS_i = U_j O_{ij}$). N_i biểu thị cho tình huống của T_i, trong đó $N_i \in$ {abort, commit}.

Với các thuật ngữ này, chúng ta có thể định nghĩa T_i là một *thứ tự bộ phận* trên các phép toán và tình huống kết thúc của nó. Thứ tự bộ phận $P = \{\sum, \prec\}$ định nghĩa một trật tự giữa các phần tử của $\sum$ (được gọi là *miền*) qua một quan hệ hai ngôi bắc cầu và không phản xạ $\prec$ được định nghĩa trên $\sum$. Trong trường hợp đang xét, $\sum$ bao gồm các phép toán và tình huống kết thúc của một giao tác, trong đó $\prec$ chỉ thứ tự thực hiện của các phép toán này (được chúng ta đọc là "đứng trước theo thứ tự thực thi").

Một cách hình thức, một giao tác T_I là một thứ tự bộ phận $T_i = \{\sum_i, \prec_i\}$, trong đó

1. $\sum_i = OS_i \cup \{N_i\}$.
2. Với hai phép toán bất kỳ $O_{ij}, O_{ik} \in OS_i$, nếu O_{ij} = {R(x) or W(x)} và O_{ik} =W(x) với một mục dữ liệu x nào đó, thế thì $O_{ij} <_i O_{ik}$ hoặc $O_{ik} <_i O_{ij}$.
3. $\forall O_{ij} \in OS_i, O_{ij} <_i N_i$.

Điều kiện thứ nhất về hình thức định nghĩa *miền* như một tập các thao tác đọc và ghi cấu tạo nên giao tác cộng với tình huống kết thúc, có thể là *commit* hoặc *abort*. Điều kiện thứ hai xác định quan hệ thứ tự giữa các thao tác đọc và ghi có tương tranh của giao tác, còn điều kiện cuối cùng chỉ ra rằng tình huống kết thúc luôn đi sau tất cả những thao tác khác.

Có hai điểm quan trọng cần lưu ý trong định nghĩa này. Trước tiên, quan hệ thứ tự $\prec$ được cho trước và định nghĩa này không hề xây dựng nó. Quan hệ thứ tự thực sự phụ thuộc vào ứng dụng. Kế đến, điều kiện thứ hai chỉ ra rằng thứ tự giữa các thao tác có tương tranh phải tồn tại bên trong $\prec$. Hai thao tác $O_i(x)$ và $O_j(x)$ được gọi là có tương tranh nếu O_i = Write hoặc O_j = Write (có nghĩa ít nhất một trong chúng là Write và chúng truy xuất cùng một mục dữ liệu).

Thí dụ 5.5

Xét một giao tác đơn giản T có các bước sau:

Read(x)
Read(y)
x← x+y
Write(x)
Commit

Đặc tả của giao tác này theo ký pháp hình thức đã được giới thiệu ở trên là:

$\sum$ = {R(x), R(y), W(x), C}

$\prec$ = {(R(x), W(x)), (R(y), W(x)), (W(x),C), (R(x), C), (R(y), C)}

trong đó (Oi, Oj), là một phần tử của quan hệ $\prec$, biểu thị rằng Oi < Oj.

Chú ý rằng quan hệ thứ tự tương đối của tất cả các thao tác ứng với tình huống kết thúc. Điều này là do điều kiện thứ ba của định nghĩa giao tác. Cũng cần chú ý rằng chúng ta không mô tả thứ tự giữa mỗi cặp thao tác. Điều đó giải thích vì sao đây là một thứ tự bộ phận.

Thí dụ 5.6

Giao tác đặt chỗ được xây dựng trong thí dụ 5.3 phức tạp hơn. Chú ý cho rằng có hai tình huống kết thúc, tùy vào tình trạng có còn chỗ trống hay không. Trước tiên, điều này dường như mâu thuẫn với định nghĩa của giao tác, đó là chỉ tồn tại một tình huống kết thúc. Tuy nhiên cần nhớ rằng giao tác là một thực thi của một chương trình. Rõ ràng là trong bất kỳ lần thực thi nào, chỉ một trong hai tình huống kết thúc xảy ra. Vì thế điều có thể xảy ra là một giao tác hủy bỏ và một giao tác khác ủy thác. Sử dụng ký pháp hình thức này, giao tác đầu có thể được đặc tả như sau:

$\sum$ = {R(STSOLD); R(CAP),A}

$\prec$ = {(O_1,A), (O_2,A)}

và giao tác sau được đặc tả như sau

$\sum$ = {R(STSOLD), R(CAP), W(STSOLD), W(FNO),
W(DATE), W(CNAME), W(SPECIAL), C}

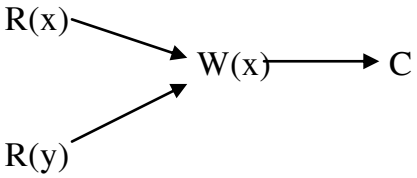
< = {(O_1,O_3), (O_2,O_3), (O_1,O_4), (O_1, O_5), (O_1, O_6),
(O_1, O_7), (O_2, O_4), (O_2, O_5), (O_2, O_6), (O_2, O_7),
(O_1, C), (O_2, C), (O_3, C) (O_4, C), (O_5, C), (O_6, C), (O_7, C)}

trong đó O_1 = R(STSOLD), O_2 = R(CAP), O_3 = W(STSOLD), O_4 = W(FNO), O_5= W(DATE), O_6 = W(CNAME), O_7 = W(SPECIAL).

Một ưu điểm của việc định nghĩa giao tác như một thứ tự bộ phận là sự tương ứng của nó với *đồ thị có hướng không vòng* DAG (directed acylic graph). Như thế một giao tác có thể được đặc tả như một DAG với đỉnh là các phép toán giao tác và cung chỉ ra mối liên hệ thứ tự giữa các cặp phép toán đã cho.

Thí dụ 5.7

Giao tác được thảo luận trong thí dụ 5 có thể được vẽ như một DAG của hình 5.2. Chú ý rằng chúng ta không vẽ các cung được suy ra nhờ tính chất bắc cầu dù rằng chúng ta xem chúng như những phần tử của $\prec$



Hình 5.2. Biểu diễn dạng DAG cho một giao tác.

Trong phần lớn các trường hợp chúng ta không cần phải đề cập đến nhiều miền của thứ tự bộ phận một cách riêng rẽ với quan hệ thứ tự. Vì thế thông thường chúng ta bỏ $\sum$ ra khỏi định nghĩa giao tác và sử dụng tên của thứ tự bộ phận để chỉ đến cả miền lẫn tên của thứ tự bộ phận. Điều đó sẽ tiện lợi bởi vì nó cho phép chúng ta đặc tả thứ tự của các phép toán trong một giao tác nhờ một phương thức khá đơn giản bằng cách dùng thứ tự tương đối của định nghĩa giao tác. Chẳng hạn chúng ta có thể định nghĩa giao tác của thí dụ 5 như sau:

$$T = \{R(x), R(y), W(x), C\}$$

thay vì đặc tả dài dòng như trước.

5.2 CÁC TÍNH CHẤT CỦA GIAO TÁC

Các khía cạnh nhất quán và khả tín của giao tác là do bốn tính chất: (1) *tính nguyên tử* (atomicity), (2) *tính nhất quán* (consistency), (3) *tính biệt lập* (isolation), (4) *tính bền vững* (durability); và chúng ta thường được gọi chung là tính chất ACID.

5.2.1 Tính nguyên tử

Tính nguyên tử liên quan đến sự kiện là một giao tác được xử lý như một đơn vị hoạt tác. Chính vì thế mà các hành động của giao tác, hoặc tất cả đều hoàn tất hoặc không một hành động nào hoàn tất. Điều này cũng được gọi là tính chất "được ăn cả ngã về không" (all-or-nothing). Tính nguyên tử đòi hỏi rằng nếu việc thực thi giao tác bị cắt ngang bởi một loại sự cố nào đó thì hệ quản trị cơ sở dữ liệu sẽ chịu trách nhiệm xác định công việc cần thực hiện đối với giao tác để khôi phục lại sau sự cố. Dĩ nhiên có hai chiều hướng hành động: hoặc nó sẽ được kết thúc bằng cách hoàn tất các hành động còn lại, hoặc có thể được kết thúc bằng cách hồi lại tất cả các hành động đã được thực hiện.

5.2.2 Tính nhất quán

Tính nhất quán (consistency) 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.

Trong định nghĩa dưới đây, 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. Do đó dựa trên khái niệm về dữ liệu rác, ba mức độ được định nghĩa như sau:

Độ 3: Giao tác T thỏa **nhất quán độ 3** nếu:
1. T không đè lên dữ liệu rác của những giao tác khác.
2. T không ủy thác bất kỳ thao tác ghi nào cho đến khi nó hoàn tất mọi thao tác ghi [nghĩa là cho đến lúc cuối giao tác (end-of-transaction, EOT)].
3. T không đọc dữ liệu rác của những giao tác khác.
4. *Những giao tác khác không làm cho dữ liệu mà T đã đọc - trước khi T hoàn tất - trở thành dữ liệu rác*.

Độ 2: Giao tác T thỏa **nhất quán độ 2** nếu:
1. T không đè lên dữ liệu rác của những giao tác khác.
2. T không ủy thác bất kỳ thao tác ghi nào trước EOT.
3. T không đọc dữ liệu rác của những giao tác khác.

Độ 1: Giao tác T thỏa **nhất quán độ 1** nếu:
1. T không đè lên dữ liệu rác của những giao tác khác.
2. T không ủy thác bất kỳ thao tác ghi nào trước EOT.

Đương nhiên độ nhất quán cao bao trùm tất cả độ nhất quán mức thấp hơn. Ý tưởng trong việc định nghĩa nhiều mức nhất quán là cung cấp cho lập trình viên ứng dụng một khả năng linh hoạt khi định nghĩa các giao tác hoạt tác ở những mức khác nhau. Hệ quả là mặc dù một số giao tác hoạt tác ở mức nhất quán Độ 3, các giao tác khác có thể hoạt tác ở những mức thấp hơn, và rất có thể sẽ nhìn thấy các dữ liệu rác.

5.2.3 Tính biệt lập

Biệt lập là tính chất của các giao tác, đòi hỏi mỗi giao tác phải luôn nhìn thấy cơ sở dữ liệu nhất quán. Nói cách khác, một giao tác đang thực thi không thể làm lộ ra các kết quả của nó cho những giao tác khác đang cùng hoạt động trước khi nó ủy thác.

Có một số lý do cần phải nhấn mạnh đến tính biệt lập. Một là duy trì tính nhất quán qua lại giữa các giao tác. Nếu hai giao tác đồng thời truy xuất đến một mục dữ liệu đang được một trong chúng cập nhật thì không thể bảo đảm rằng giao tác thứ hai sẽ đọc được giá trị đúng.

Thí dụ 5.8

Xét hai giao tác đồng thời T1 và T2 cùng truy xuất đến mục dữ liệu x. Giả sử **<u>giá trị của x trước khi bắt đầu thực hiện là 50</u>**.

T1:	Read(x)	T2:	Read(x)
	x← x + 1		x← x + 1
	Write(x)		Write(x)

Commit	Commit

Dưới đây là một dãy thực thi cho các hành động này.

T_1:	Read(x)
T_1:	x← x + 1
T_1:	Write(x)
T_1:	Commit
T_2:	Read(x)
T_2:	x← x + 1
T_2:	Write(x)
T_2:	Commit

Ở trường hợp này không có vấn đề gì; các giao tác T_1 và T_2 được thực hiện lần lượt và giao tác T_2 đọc được giá trị của x là 51. Chú ý rằng nếu T_2 thực thi trước T_1 thì T_2 đọc được giá trị 50. Vì thế nếu T1 và T2 được thực thi lần lượt giao tác này rồi đến giao tác kia, giao tác thứ hai sẽ đọc được giá trị của x là 51 và sau khi kết thúc hai giao tác x có giá trị 52. Tuy nhiên vì các giao tác đang thực thi đồng thời, dãy thực thi sau đây có thể sẽ xảy ra:

T_1:	Read(x)
T_1:	x← x + 1
T_2:	Read(x)
T_1:	Write(x)
T_2:	x← x + 1
T_2:	Write(x)
T_1:	Commit
T_2:	Commit

Trong trường hợp này, giao tác T_2 đọc được giá trị của x là 50. Giá trị này không đúng bởi vì T_2 đọc x trong khi giá trị của nó đang được thay đổi từ 50 thành 51. Hơn nữa giá trị của x sẽ là 51 vào lúc kết thúc các giao tác T_1 và T_2 bởi vì hành động ghi của T_2 sẽ đè lên kết quả ghi của T_1.

Bảo đảm tính biệt lập bằng cách không cho phép các giao tác khác nhìn thấy các kết quả chưa hoàn tất như trong thí dụ trên sẽ giải quyết được vấn đề *cập nhật thất lạc* (lost update). Loại biệt lập này đã được gọi là *tính ổn định con chạy* (cursor stability). Trong thí dụ ở trên, dãy thực thi thứ hai đã làm cho tác dụng của T_1 bị mất. Một lý do thứ hai của tính biệt lập là các *hủy bỏ dây chuyền* (cascading abort). Nếu một giao tác cho phép những giao tác khác nhìn thấy những kết quả chưa hoàn tất của nó trước khi ủy thác rồi nó quyết định hủy bỏ, mọi giao tác đã đọc những giá trị chưa hoàn tất đó cũng sẽ phải hủy bỏ. Xâu mắc xích này dễ làm tăng nhanh và gây ra những phí tổn đáng kể cho hệ quản trị cơ sở dữ liệu.

Cũng có thể xử trí các mức nhất quán đã thảo luận trong phần trước từ quan điểm của tính chất biệt lập (vì thế đã minh hoạ cho sự phụ thuộc giữa tính nhất quán và tính biệt lập). Khi di chuyển lên cây phân cấp các mức nhất quán, các giao tác ngày càng biệt lập hơn. Độ 0 cung cấp rất ít tính chất "biệt lập" ngoài việc ngăn cản các cập nhật thất lạc. Tuy nhiên vì các giao tác sẽ ủy thác trước khi chúng hoàn tất tất cả mọi thao tác ghi của chúng, nếu có một hủy bỏ xảy ra sau đó, nó sẽ đòi hỏi phải hồi lại tất cả các cập nhật trên các mục dữ liệu đã được ủy thác và hiện đang được truy xuất bởi những giao tác khác. Nhất quán độ 2 tránh được các hủy bỏ dây chuyền. Độ 3 cung cấp toàn bộ khả năng biệt lập, buộc một trong các giao tác tương tranh phải đợi cho đến khi giao tác kia kết thúc. Những dãy thực thi như thế được gọi là nghiêm ngặt (strict) và sẽ được thảo luận nhiều hơn trong chương tiếp theo. Rõ ràng là vấn đề biệt lập có liên quan trực tiếp đến tính nhất quán cơ sở dữ liệu và vì thế là đề tài của điều khiển đồng thời.

Ba hiện tượng được đặc tả cho những tình huống có thể xảy ra nếu sự biệt lập thích hợp không được duy trì là:

Đọc rác (Dirty Read): dữ liệu rác muốn nói đến các mục dữ liệu mà giá trị của chúng đã được sửa đổi bởi một giao tác chưa ủy thác. Xét trường hợp giao tác T1 sửa đổi một giá trị dữ liệu rồi nó lại được bọc bởi một giao tác T2 khác trước khi T1 thực hiện Commit hay Abort. Trong trường hợp Abort, T2 đã đọc một giá trị chưa được tồn tại trong cơ sở dữ liệu.

Một đặc tả chính xác trong hiện tượng này như sau (với các cước số chỉ ra tên các giao tác)

..., $W_1(x)$,..., $R_2(x)$,... C_1(hoặc A_1),..., C_2(hoặc A_2)

hoặc

..., $W_1(x)$,..., $R_2(x)$,... C_2(hoặc A_2),..., C_1(hoặc A_1)

Không thể đọc lại (Non-repeatable Read): Giao tác T1 đọc một mục dữ liệu. Sau đó một giao tác T2 khác sửa hoặc xoá mục dữ liệu đó rồi ủy thác. Nếu sau đó T1 đọc lại mục dữ liệu đó, hoặc nó đọc được một giá trị khác hoặc nó không thể tìm thấy được mục đó; vì thế hai hành động đọc trong cùng một giao tác T1 trả về các kết quả khác nhau.

Một đặc tả chính xác của hiện tượng này như sau:

..., $R_1(x)$,..., $W_2(x)$,... C_1(hoặc A_1),..., C_2(hoặc A_2)

hoặc

..., $R_1(x)$,..., $W_2(x)$,... C_2(hoặc A_2),..., C_1(hoặc A_1)

Ảnh ảo (phantom): Điều kiện ảnh ảo trước kia đã được định nghĩa xảy ra khi T1 thực hiện tìm kiếm theo một vị từ và T2 chèn những bộ mới thỏa vị từ đó. Đặc tả chính xác của hiện tượng này là (P là vị từ tìm kiếm)

..., $R_1(P)$,..., W_2(y thuộc P),... C_1(hoặc A_1),..., C_2(hoặc A_2)

hoặc

..., $R_1(P)$,..., R_2(y thuộc P),... C2(hoặc A_2),..., C_1(hoặc A_1)

Dựa trên những hiện tượng này, các mức biệt lập đã được định nghĩa như sau. Mục tiêu của việc định nghĩa nhiều mức biệt lập cũng giống như việc định nghĩa các mức nhất quán.

5.2.4 Tính bền vững

Tính bền vững (durability) muốn nói đến tính chất của giao tác, bảo đảm rằng một khi giao tác ủy thác, kết quả của nó được duy trì cố định và không bị xóa ra khỏi cơ sở dữ liệu. Vì thế hệ quản trị cơ sở dữ liệu bảo đảm rằng kết quả của giao tác sẽ vẫn tồn tại dù có xảy ra sự cố hệ thống. Đây chính là lý do mà trong thí dụ 5.2 chúng ta đã nhấn mạnh rằng giao tác ủy thác trước khi nó thông báo cho người sử dụng biết rằng nó đã hoàn tất thành công. Tính bền vững đưa ra vấn đề *khôi phục* cơ sở dữ liệu (database recovery), nghĩa là cách khôi phục cơ sở dữ liệu về trạng thái nhất quán mà ở đó mọi hành động đã ủy thác đều được phản ánh.

5.3 CÁC LOẠI GIAO TÁC

5.3.1 Giao tác phẳng

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 thí 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.

5.3.2 Giao tác lồng

Đây là mô hình giao tác cho phép một giao tác gồm 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)

Thí dụ 5.10

Chúng ta hãy mở rộng giao tác đặt chỗ của thí dụ 2. Phần lớn các hãng du lịch đều lo cả việc đặt chỗ khách sạn và mướn ô tô ngoài dịch vụ đặt vé máy bay. Nếu người ta muốn mô tả tất cả những công việc này bằng một giao tác, thì giao tác đặt chỗ sẽ có cấu trúc như sau:

```
Begin_transaction Reservation
begin
   Begin_transaction Airline
   ...
      Begin_transaction Hotel
   ...
   end. { Hotel}
```

```
.....
end. {Airline}

   Begin_transaction Car
   ...
   end. {Car }
end.
```

Các giao tác lồng đã được chú ý như một khái niệm giao tác tổng quát hơn. Mức độ lồng nói chung là để ngỏ, cho phép các giao tác con cũng có thể có các giao tác lồng. Tính tổng quát này có ích trong các lãnh vực ứng dụng mà ở đó các giao tác phức tạp hơn so với việc xử lý dữ liệu truyền thống.