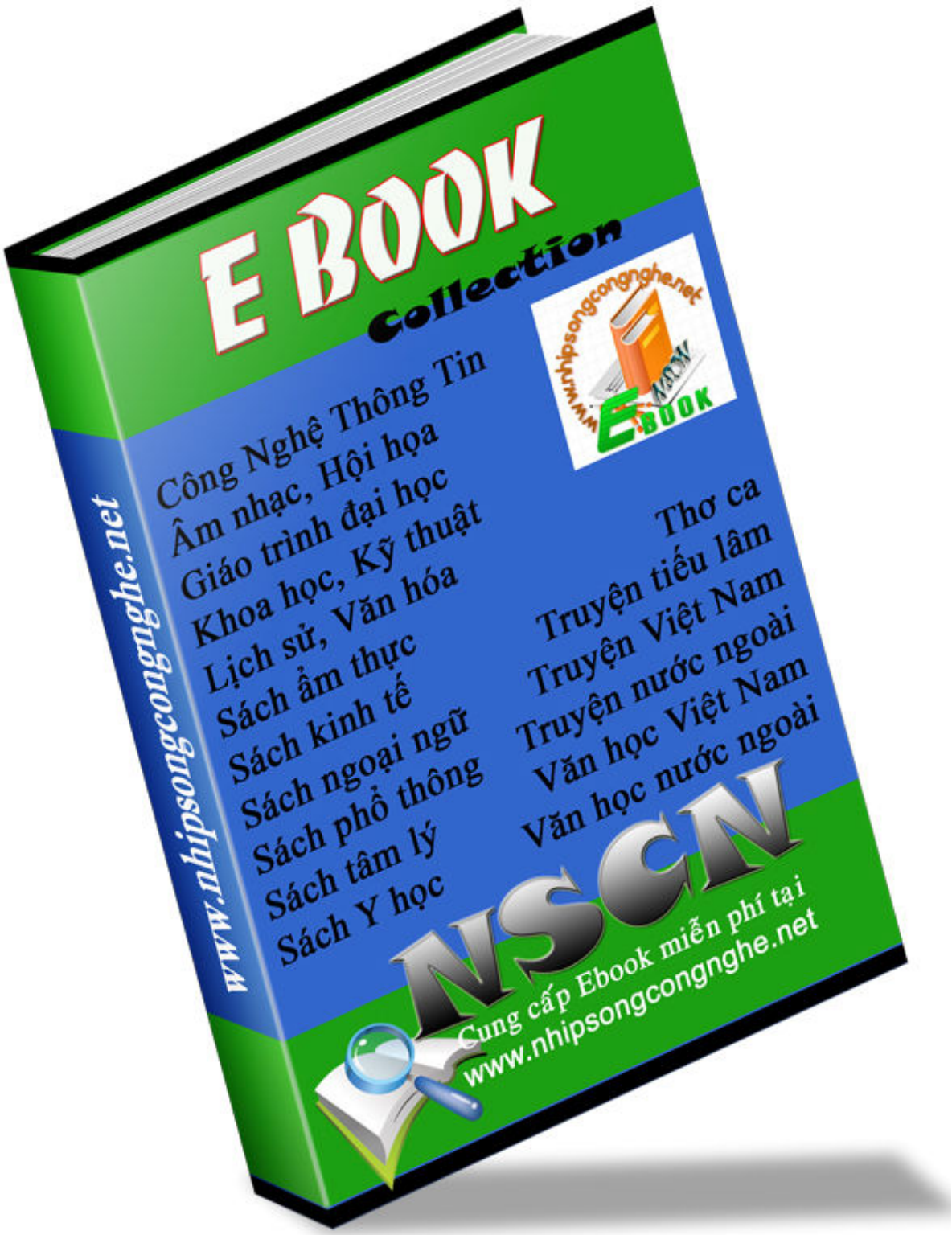

E BOOK
Collection
Công Nghệ Thông Tin
Âm nhạc, Hội họa
Giáo trình đại học
Khoa học, Kỹ thuật
Lịch sử, Văn hóa
Sách ẩm thực
Sách kinh tế
Sách ngoại ngữ
Sách phổ thông
Sách tâm lý
Sách Y học
Thơ ca
Truyện tiếu lâm
Truyện Việt Nam
Truyện nước ngoài
Văn học Việt Nam
Văn học nước ngoài
www.nhipsongcongnghe.net
NSCN
Cung cấp Ebook miễn phí tại
www.nhipsongcongnghe.net

TÍNH ƯU VIỆT CỦA CSDL PHÂN TÁN

Chương 1: Tính ưu việt của CSDL phân tán:

I/Khái niệm về CSDL phân tán:

1/Định nghĩa CSDL phân tán:

Định nghĩa: Một CSDL phân tán là một tập hợp dữ liệu mà về mặt logic tập dữ liệu này thuộc về một hệ thống, nhưng được trải trên các vị trí khác nhau của một mạng máy tính.

Có hai điểm quan trọng được nêu ra trong định nghĩa trên:

-Phân tán: Dữ liệu không cư trú trên một vị trí, điều này giúp chúng ta có thể phân một CSDL phân tán với một CSDL tập trung, đơn lẻ.

-Tương quan logic: Dữ liệu có một số các thuộc tính ràng buộc chúng với nhau, điều này giúp chúng ta có thể phân biệt một CSDL phân tán với một tập hợp CSDL cục bộ hoặc các tệp cư trú tại các vị trí khác nhau trong một mạng máy tính.

Thế nào là phân tán:

Xử lý logic hoặc xử lý nguyên tố được phân tán.

Phân tán theo chức năng: Nhiều chức năng của hệ thống máy tính có thể được uỷ thác cho các phần cứng hoặc phần mềm hoặc cả hai.

Phân tán dữ liệu.

Phân tán điều khiển.

2/Phân lớp các hệ thống tính toán phân tán:

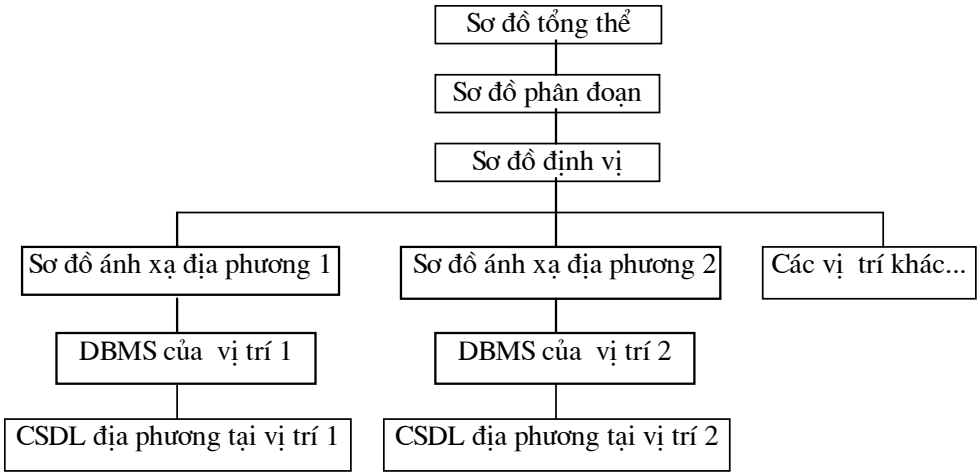
Trình độ mắc nối: Khoảng cách của các yếu tố xử lý được kết nối.

Cấu trúc nối kết với nhau

Sự độc lập đối với nhau giữa các thành phần

3/Kiến trúc cơ bản của CSDL phân tán:

Đây không là kiến trúc tường minh cho tất cả các CSDL phân tán, tuy vậy kiến trúc này thể hiện tổ chức của bất kỳ một CSDL phân tán nào.



Hình 1.I.1 Kiến trúc cơ bản của CSDL phân tán

-Sơ đồ tổng thể: Định nghĩa tất cả các dữ liệu sẽ được lưu trữ trong CSDL phân tán. Trong mô hình quan hệ, sơ đồ tổng thể bao gồm định nghĩa của tập các quan hệ tổng thể.

-Sơ đồ phân đoạn: Mỗi quan hệ tổng thể có thể chia thành một vài phần không gối lên nhau được gọi là đoạn (**fragments**). Có nhiều các khác nhau để thực hiện việc phân chia này. ánh xạ(một nhiều) giữa sơ đồ tổng thể và các đoạn được định nghĩa trong sơ đồ phân đoạn.

-Sơ đồ định vị: các đoạn là các phần logic của quan hệ tổng thể được định vị vật lý trên một hoặc nhiều vị trí trên mạng. Sơ đồ định vị định nghĩa đoạn nào định vị tại các vị trí nào. Lưu ý rằng kiểu ánh xạ được định nghĩa trong sơ đồ định vị quyết định CSDL phân tán là dư thừa hay không.

-Sơ đồ ánh xạ địa phương: ánh xạ các ảnh vật lý và các đối tượng được lưu trữ tại một trạm (tất cả các đoạn của một quan hệ tổng thể trên cùng một vị trí tạo ra một ảnh vật lý).

II/Các đặc điểm của CSDL phân tán đối ngược lại CSDL tập trung:

-Điều khiển tập trung: Trong CSDL phân tán khái niệm này không được nhấn mạnh. Khả năng điều khiển tập trung phụ thuộc vào kiến trúc của CSDL phân tán. Trong CSDL phân tán có khả năng nhận biết cấu trúc điều khiển phân cấp dựa trên một người quản trị CSDL toàn cục (có trách nhiệm trên toàn thể CSDL phân tán), và các người quản trị CSDL cục bộ (có trách nhiệm trên CSDL cục bộ của chúng). Điều này nhấn mạnh rằng các người quản trị CSDL cục bộ có thể có trình độ tự trị cao. Các CSDL phân tán có thể rất khác nhau về trình độ tự trị: từ hoàn toàn tự trị không có bất cứ một hệ quản trị CSDL tập trung nào đến hầu như hoàn toàn điều khiển tập trung.

-Độc lập dữ liệu: Tổ chức thực sự của dữ liệu là trong suốt đối với các chương trình ứng dụng. Các chương trình được viết có một khung nhìn

nhận thức (conceptual) được gọi là sơ đồ nhận thức. Lợi ích chủ yếu là chương trình không bị ảnh hưởng bởi những thay đổi tổ chức vật lý của dữ liệu. Trong CSDL phân tán, đọc lập dữ liệu cũng quan trọng như trong CSDL truyền thống. Tuy nhiên, có một khái niệm mới nảy sinh có tên là trong suốt phân tán. Trong suốt phân tán có nghĩa là một chương trình được viết (trên một CSDL phân tán) như CSDL không được phân tán. Hay nói cách khác chương không bị ảnh hưởng bởi sự di chuyển dữ liệu từ một vị trí các sang vị trí khác, tuy nhiên tốc độ thực hiện của nó bị ảnh hưởng. Độc lập dữ liệu được cung cấp trong CSDL truyền thống được cung cấp qua nhiều mức kiến trúc có sự mô tả về dữ liệu và ánh xạ giữa chúng khác nhau; các khái niệm: sơ đồ nhận thức, sơ đồ lưu trữ, sơ đồ ngoài (external schema). Một cách tương tự như vậy, Trong suốt phân tán đạt được trong CSDL phân tán các mức và các sơ đồ mới:

-Trong suốt phân đoan: Các ứng dụng thực hiện các truy nhập vào CSDL như nó không được phân tán.

-Trong suốt đinh vi: Các ứng dụng phải xác định truy nhập vào đoạn nào của CSDL phân tán. Có thể truy nhập song song vào nhiều đoạn cùng một lúc để tận dụng khả năng song song của CSDL phân tán.

-Trong suốt ánh xa địa phương: Các ứng dụng phải xác định truy nhập vào đoạn nào tại vị trí nào của CSDL phân tán.

-Không trong suốt: Người lập trình ứng dụng phải viết các chương trình có thể chạy được trên hệ thống hệ quản trị CSDL địa phương (DBMSs) được cài đặt tại vị trí ứng dụng cần đọc dữ liệu (trên các vị trí khác nhau các hệ điều hành có thể khác nhau, hoặc DBMSs có thể khác nhau: các bản dịch (release) khác nhau trong cùng một hệ thống, các hệ thống khác nhau trong cùng một kiểu- ví dụ các DBMSs khác nhau trong họ Codasyl -, các họ hệ thống khác kiểu- ví dụ một quan hệ và một hệ thống Codasyl -.), các chương trình này thực hiện yêu cầu các hàm và cài đặt các chương trình phụ trợ tại các vị trí được yêu cầu. ứng dụng phải được viết với một yêu cầu làm hoạt động các chương trình phụ trợ ở xa này thay thế các lệnh SQL.

-Giảm dư thừa: Trong CSDL truyền thống dư thừa được giảm tới mức có thể vì hai nguyên nhân sau:

-Sự mâu thuẫn giữa một vài bản sao của cùng một dữ liệu được tự động tránh vì thực tế chỉ có một bản.

-Tiết kiệm không gian lưu trữ.

Trong CSDL phân tán, có một vài nguyên nhân làm cho việc quan tâm đến sự dư thừa dữ liệu như một đặc điểm ao ước:

-Vị trí của các ứng dụng có thể được tăng nếu dữ liệu được sao bản tại tất cả các vị trí cần đến nó.

-Tính sẵn sàng của hệ thống có thể tăng vì nếu một vị trí lỗi không dừng việc thực hiện của các ứng dụng tại các vị trí khác nếu dữ liệu được sao bản.

Do đó, việc giảm dư thừa đòi hỏi một sự ước lượng định giá khá phức tạp. Và việc sao bản là tỉ lệ thuận với việc tăng số lượng thực hiện các truy nhập sửa đổi dữ liệu vì khi thực hiện một truy nhập sửa đổi trên một dữ liệu chúng ta đồng thời phải sửa đổi dữ liệu trên các sao bản của dữ liệu đó.

-Cấu trúc vật lý phức tạp và việc truy nhập hiệu quả: Các cấu trúc truy nhập phức tạp, ví dụ như các chỉ số (index) thứ hai, các chuỗi tệp có quan hệ với nhau (interfile chain) ..., là mặt chủ yếu của CSDL truyền thống. Hỗ trợ các cấu trúc này là một phần hết sức quan trọng của hệ quản trị CSDL. Nguyên nhân cho việc cung cấp các cấu trúc truy nhập phức tạp là để thu được hiệu quả truy nhập vào dữ liệu. Trong CSDL phân tán các cấu trúc truy nhập phức tạp không là công cụ đúng cho hiệu quả truy nhập. Hiệu quả truy nhập CSDL phân tán không thể được cung cấp bởi các cấu trúc phức tạp các vị trí có quan hệ với nhau.

-Tính toàn vẹn dữ liệu, khôi phục lại và điều khiển tương tranh: Trong CSDL, vấn đề toàn vẹn, khôi phục lại, và điều khiển tương tranh, mặc dù là các vấn đề khác nhau song chúng có quan hệ qua lại chặt chẽ với nhau. Giải pháp cho các vấn đề này chủ yếu là việc cung cấp các giao tác (transaction). Khái niệm giao tác và vấn đề quản lý giao tác sẽ đề cập ở phần sau.

-Biệt lập (Privacy) và bảo mật: Trong CSDL truyền thống, các người quản trị CSDL có điều khiển tập trung, có thể đảm bảo rằng chỉ một truy nhập được uỷ quyền được thực hiện. Lưu ý rằng, dù sao, CSDL tập trunggần như tự chính nó không với một thủ tục điều khiển đặc biệt nào, là nhiều nhược điểm riêng biệt hơn và xâm phạm bảo mật hơn con đường cũ dựa trên các tệp riêng lẻ. Trong CSDL phân tán, các người quản trị cục bộ thực chất đương đầu với các vấn đề giống các người quản trị CSDL trong CSDL truyền thống. Dù sao, hai khía cạnh đặc biệt của CSDL phân tán đáng được đề cập đến:

-Trong một CSDL phân tán với một trình độ tự quản của các vị trí rất cao, các người chủ dữ liệu địa phương cảm giác được bảo vệ hơn vì họ có thể tự chủ các việc bảo vệ thay vì phụ thuộc vào người quản trị CSDL trung tâm.

-Các vấn đề bảo mật là bản chất trong hệ phân tán nói chung, vì các mạng truyền thông có thể biểu hiện một điểm yếu với sự lưu ý bảo vệ.

III/Tính ưu việt của CSDL phân tán:

1/Các lợi ích của CSDL phân tán:

-Khả năng mau phục hồi (Resilience): Việc truy nhập dữ liệu không phụ thuộc vào một máy hay một đường nối trên mạng. Nếu có bất kỳ một lỗi nào thì sau đó vài CSDL có thể được truy nhập trên các nút địa phương, hơn nữa một lỗi trên đường nối có thể tự động chọn đường lại qua các đường nối khác.

-Giảm dòng dữ liệu trên đường truyền cải thiện thời gian trả lời: Cung cấp trả lời có bởi dữ liệu gần sát nơi các người sử dụng thường xuyên dữ liệu.

-Khung nhìn logic đơn cho các câu hỏi: Trong suốt định vị cho phép dữ liệu vật lý có thể được di chuyển mà không thay đổi ứng dụng hay thông báo cho người sử dụng.

-Tự tri địa phương:

-Việc quản lý: Được quản lý một cách độc lập.

-Việc tạo lập: Có các định nghĩa tạo lập và điều khiển có tính cục bộ.

-Điều khiển truy nhập: Có quyền định nghĩa và điều khiển có tính cục bộ.

-Giảm cạnh tranh (reduced politics): bằng cách trao vùng nghiệp vụ tự chủ cục bộ.

-Cách thức mở rộng dễ dàng: Dễ dàng phát triển mở rộng đạt được:

-Nhiều bộ xử có thể được thêm vào mạng.
-Nhiều CSDL có thể được thêm vào trên một nút mạng.
-Cập nhật phần mềm là độc lập với cấu trúc vật lý.

2/Các bất lợi của CSDL phân tán:

- Sự thiếu kinh nghiệm.
- Phức tạp.
- Giá cả: Nâng cấp phần cứng, phần mềm.
- Sự phân tán trong điều khiển.
- Bảo mật: Khó khăn hơn CSDL tập trung.
- Khó khăn trong việc thay đổi: Hiện nay chưa có các công cụ hoặc phương pháp nào để trợ giúp người sử dụng chuyển đổi dữ liệu của họ từ CSDL tập trung sang CSDL phân tán. Nghiên cứu CSDL không thuần nhất và sự thống nhất CSDL được chờ đợi để giải quyết khó khăn này.

3/Các nguyên nhân sử dụng CSDL phân tán:

-Nguyên nhân về tổ chức và kinh tế: Trên thực tế nhiều tổ chức là không tập trung vì vậy CSDL phân tán là con đường thích hợp với cấu trúc tự nhiên của các tổ chức đó. Với sự phát triển gần đây trong các kỹ thuật máy tính, cán cân kinh tế thúc đẩy có hệ lớn, các trung tâm máy tính trở lên đáng nghi ngại. Nguyên nhân về tổ chức và kinh tế là nguyên nhân hết sức quan trọng cho việc phát triển CSDL phân tán.

-Sự liên kết các CSDL dang tồn tai: CSDL phân tán là giải pháp tự nhiên khi có các CSDL đang tồn tại và sự cần thiết thực hiện xây dựng một ứng dụng toàn cục. Trong trường hợp này CSDL phân tán được tạo từ dưới lên (bottom-up) từ các CSDL đã tồn tại trước đó. Tiến trình này có thể đòi hỏi cấu trúc lại cục bộ ở một mức độ nhất định. Dù sao, những sửa đổi này là nhỏ hơn rất nhiều so với việc tạo lập một CSDL tập trung hoàn toàn mới.

-Sự phát triển mở rộng: Các tổ chức có thể mở rộng bằng cách thêm các đơn vị mới, vừa có tính tự trị vừa có quan hệ tương đối với các tổ chức khác. Khi đó con đường CSDL phân tán hỗ trợ một sự mở rộng uyển chuyển với một mức độ ảnh hưởng tối thiểu tới các đơn vị đang tồn tại. Với con đường CSDL tập trung, cũng có thể khởi tạo kích thước lớn cho việc mở rộng trong tương lai. Điều đó rất khó tiên định và thực hiện với một phí tổn lớn, hoặc sự mở rộng này có ảnh hưởng lớn không chỉ trên các ứng dụng mới mà còn trên các ứng dụng đang tồn tại.

-Làm giảm tổng chi phí tìm kiếm: Trên thực tế nhiều ứng dụng cục bộ rõ ràng giảm tổng chi phí truyền thôngvới phương diện một CSDL tập trung. Bởi vậy số tối đa các vị trí của các ứng dụng là một trong các mục đích chính trong thiết kế CSDL phân tán.

-Sự quan tâm hiệu suất (Performance considerations): Sư tồn tại một vài bộ vi xử lý tự trị đưa đến kết quả tăng hiệu suất thông qua một mức độ song song cao. Sự quan tâm này có thể chỉ có thể được ứng dụng cho một hệ thống đa xử lý không nhất thiết phải là một hệ CSDL phân tán. Dù sao, CSDL phân tán có lợi trong sự phân tán dữ liệu phản ánh các tiêu chuẩn phụ thuộc ứng dụng cái số tối đa vị trí các ứng dụng. Trong cách này sự gây trở ngại lẫn nhau giữa các bộ vi xử lý là tối thiểu. Trọng tải được chia sẻ giữa các bộ vi xử lý, và các tắc nghẽn nguy kịch, như mạng truyền thông tự nó hoặc dịch vụ chung cho toàn bộ hệ thống là được tránh. Kết quả này là một hệ quả của đòi hỏi khả năng xử lý tự trị cho các ứng dụng cục bộ đã được phát biểu trong định nghĩa CSDL phân tán.

-Tính tin cậy và tính sẵn sàng: Con đường CSDL phân tán, đặc biệt với dữ liệu dư thừa,nó cũng được sử dụng để đạt được một sự tin cậy và tính sẵn sàng cao hơn. Dù sao, việc thu được mục đích này không phải không rắc rối và đòi hỏi sử dụng các kỹ thuật vẫn chưa được hiểu biết hoàn

chỉnh. Khả năng xử lý tự trị của các vị trí khác nhau tự nó không đảm bảo một tính tin cậy toàn bộ cao của hệ thống, nhưng nó đảm bảo một thuộc tính graceful degration. Nói một cách khác, sự cố trong CSDL phân tán có thể thường xuyên hơn một CSDL tập trung vì có số lượng thành phần lớn hơn, nhưng hậu quả của sự cố được hạn chế tới các ứng dụng sử dụng dữ liệu của vị trí có sự cố, và đổ vỡ hoàn toàn của hệ thống là hiếm xảy ra.

Chương 2: Các kỹ thuật sử dụng trong cơ sở dữ liệu phân tán:

I/Thiết kế cơ sở dữ liệu phân tán:

Thiết kế một hệ thống máy tính phân tán là việc quyết định sắp đặt dữ liệu và chương trình tới các trạm làm việc của mạng máy tính. Trong trường hợp thiết kế DBMSs có hai vấn đề chính là:

- Sự phân bố dữ liệu của DBMS.
- Sự phân bố các chương trình ứng dụng chạy trên nó.

1/Tổ chức của hệ thống CSDL phân tán:

Giả thiết có một mạng máy tính đã được thiết kế. Ta chỉ quan tâm đến việc thiết kế dữ liệu phân tán. Tổ chức của các hệ thống phân tán được nghiên cứu theo 3 chiều trực giao sau:

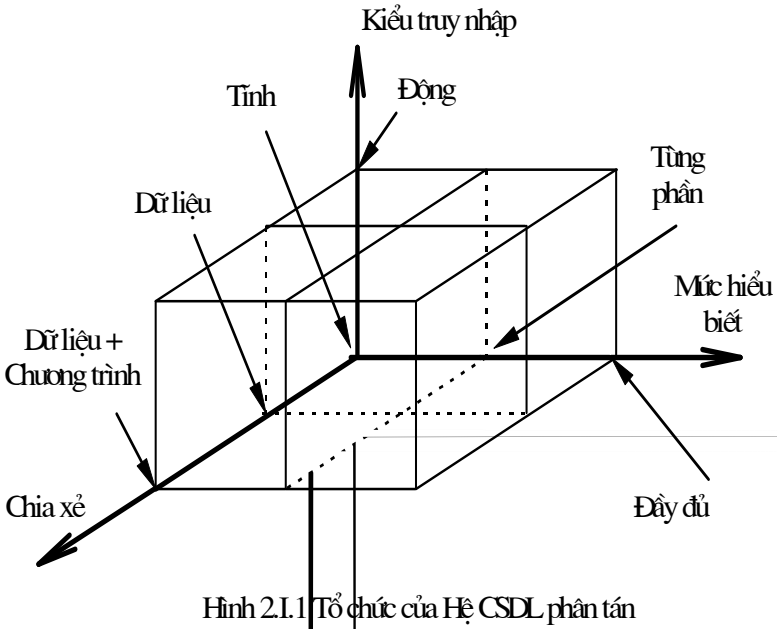
-Tầng chia xẻ.
-mô hình truy nhập.
-Mức hiểu biết.

Trong giới hạn của chiều chia xẻ có ba khả năng sau cho dữ liệu và chương trình:

-Không chia xẻ: Mỗi ứng dụng và dữ liệu của nó thực hiện tại một vị trí, không có sự liên lạc với một chương trình hoặc truy nhập tới một file dữ liệu tại những vị trí khác.

-Chia xẻ dữ liệu: Các chương trình phân phối được tại tất cả các vị trí, nhưng file dữ liệu thì không như vậy, nó vẫn chỉ được thực hiện tại một vị trí.

-Chia xẻ dữ liệu và chương trình: Cả dữ liệu và chương trình đều có thể được chia xẻ, nghĩa là một chương trình từ một vị trí có thể yêu cầu một dịch vụ từ chương trình khác tại vị trí khác, trong khi quay trở lại có thể phải truy nhập một file dữ liệu được xác định tại vị trí thứ ba.



Hình 2.I.1 Tổ chức của Hệ CSDL phân tán

Mô hình truy nhập: Các yêu cầu truy nhập dữ liệu của người sử dụng có thể là tĩnh (không thay đổi theo thời gian) hoặc động. Rõ ràng thiết kế và quản lí các môi trường tĩnh dễ hơn nhiều so với việc thiết kế và quản lí các hệ thống phân tán động. Dọc theo chiều này quan hệ giữa thiết kế CSDL phân tán và xử lý câu hỏi đã được thiết lập.

Mức hiểu biết: Mức hiểu biết về cư xử mô hình truy nhập. Có một khả năng lý thuyết là các người thiết kế không có bất kỳ thông tin người user truy nhập vào CSDL như thế nào.

2/Khung làm việc chung cho thiết kế hệ CSDL phân tán:

Từ sơ đồ kiến trúc của Hệ CSDL phân tán, người ta đưa ra sơ đồ thiết kế chung cho Hệ CSDL phân tán như sau:

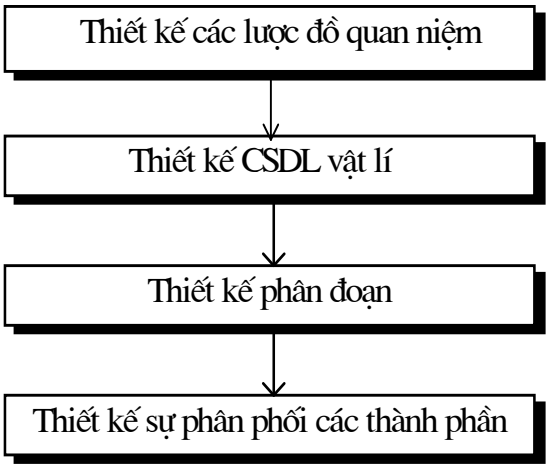
Thiết kế lược đồ quan niệm: Mô tả toàn bộ dữ liệu sẽ được dùng trong ứng dụng.

Thiết kế CSDL vật lí: Là quá trình thực hiện ánh xạ lược đồ quan niệm vào các vùng chứa và xác định cách thức truy nhập thích hợp cho chúng.

Thiết kế phân đoạn: Thực hiện việc phân chia dữ liệu thành các phần, kèm theo cách thức truy nhập thích hợp.

Thiết kế sự phân phối các phần: Các đoạn dữ liệu được đưa vào các vị trí lưu trữ thích hợp với yêu cầu hoạt động thực tế của hệ thống.

Ví dụ: Ta không thể đưa dữ liệu về CANBO cho phòng quản lý Sinh Viên và ngược lại.



Hình 2.I.2

Đối với các ứng dụng của CSDL phân tán cần chú ý:

-Vị trí mà ứng dụng được đưa ra.

-Điều khiển sự hoạt hoá của ứng dụng. Trong phương pháp chung thì ứng dụng có thể được đưa ra tại nhiều vị trí, chúng ta phải điều khiển sự hoạt hoá của ứng dụng tại mỗi vị trí.

-Thống kê phân tán các ứng dụng.

3/Các chiến lược thiết kế hệ CSDL phân tán:

Theo khung làm việc chung cho thiết kế hệ CSDL phân tán, đến nay có hai phương pháp chính là: TOP-DOWN và BOTTOM-UP.

a.Phương pháp TOP-DOWN:

TOP-DOWN: Là phương pháp thiết kế từ trên xuống và được chia ra làm nhiều giai đoạn, mỗi giai đoạn đều có nhiệm vụ riêng, giai đoạn này nối tiếp giai đoạn kia, đầu ra của giai đoạn trước được làm đầu vào cho giai đoạn kế tiếp sau nó.

Quá trình thiết kế hệ theo phương pháp TOP-DOWN bao gồm các bước sau:

Các định nghĩa: Định nghĩa môi trường hệ thống, dữ liệu và các tiến trình cho tất cả những khả năng về dữ liệu của người sử dụng. Tài liệu về những điều kiện cần thiết nằm trong hai tham số: Thiết kế View và Thiết kế mức quan niệm.

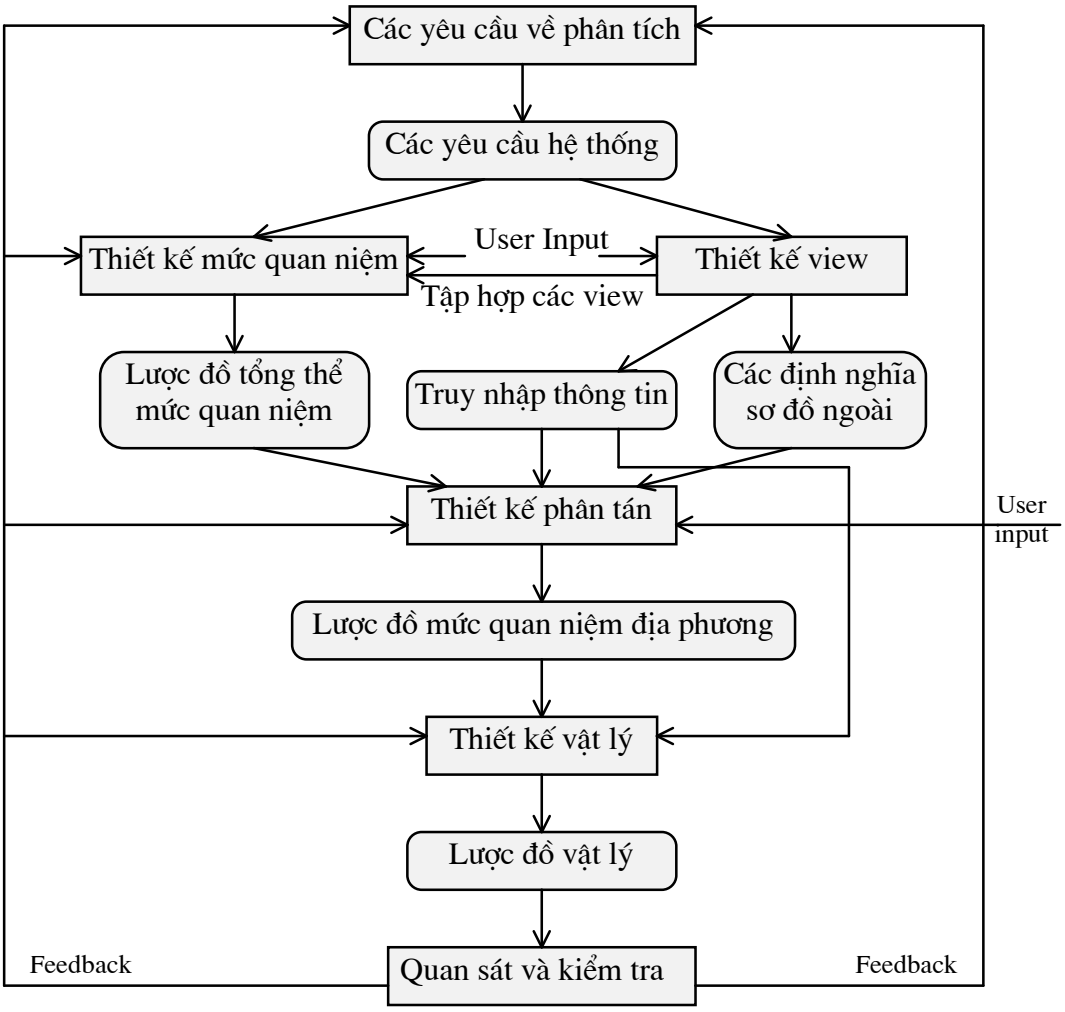
Thiết kế View: Hoạt động phân phối với sự định nghĩa những cái chung cho người sử dụng.

Thiết kế mức quan niệm: Là một tiến trình kiểm tra và xác định rõ hai nhóm quan hệ Phân tích thực thể và Phân tích chức năng:

-Phân tích thực thể: Liên quan tới sự xác định các loại thực thể, các thuộc tính và các mối quan hệ giữa chúng.

-Phân tích chức năng: Xác định các chức năng cơ sở.

Lược đồ tổng thể mức quan niệm, mẫu truy nhập thông tin và External Schema Definition: Tập hợp kết quả của các bước trên, xắp xếp các thực thể trên các vị trí của hệ thống phân tán và chuyển tới bước tiếp theo.



Hình 2.I.3 Sơ đồ thiết kế CSDL phân tán theo mô hình TOP-DOWN

Thiết kế phân tán: Thiết kế phân tán bao gồm hai phần phân đoạn và định vị CSDL.

Lược đồ mức quan niệm: Tạo ra các lược đồ CSDL mức quan niệm.

Thiết kế vật lý: Thực hiện ánh xạ các lược đồ CSDL mức quan niệm ra các đơn vị lưu trữ vật lí có giá trị tại các vị trí tương ứng.

Bộ kiểm tra: Kiểm tra các giai đoạn của quá trình thiết kế CSDL. Nếu một giai đoạn bị sai sẽ tiến hành thiết kế lại.

Phương pháp TOP-DOWN là có hiệu quả khi một hệ thống CSDL được thiết kế từ đầu. Tuy nhiên trong thực tế có một số hệ CSDL đã tồn tại thì nhiệm vụ của người thiết kế là liên kết chúng lại thành một thể thống nhất trong CSDL mới.

b.Phương pháp BOTTOM-UP:

BOTTOM-UP: Là phương pháp được xem là ngược lại với phương pháp TOP_DOWN. Trong thiết kế CSDL phân tán BOTTOM-UP sẽ bắt đầu thiết kế những lược đồ ở mức quan niệm sao cho chúng độc lập với nhau. Sau đó chúng được kết hợp lại trong một sơ đồ tổng thể (Global Conceptual schema).

Phương pháp BOTTOM-UP là phù hợp khi hệ thống CSDL được thiết kế từ những thành phần hỗn hợp.

4/Phân đoạn:

Thiết kế phân đoạn là vấn đề đầu tiên phải được giải quyết trong thiết kế CSDL phân tán. Mục đích của thiết kế phân đoạn là phân chia một quan hệ tổng thể thành các phần không bao trùm lên nhau, mỗi phần đó được gọi là một đoạn.

a.Các điều kiện ràng buộc cho thiết kế phân đoạn:

Một phương pháp thiết kế phân đoạn đúng đắn phải thoả mãn ba điều kiện ràng buộc sau:

-Tính đầy đủ: Toàn bộ dữ liệu thuộc quan hệ tổng thể phải thuộc các đoạn quan hệ và ngược lại.

-Tính rời nhau: Các đoạn phải tối thiểu hoá việc bao trùm lên nhau.

-Xây dựng lại: CSDL của quan hệ tổng thể có thể được làm lại từ các đoạn chứa nó.

b.Các phương pháp phân đoạn:

Có hai phương pháp chính là: Phân đoạn ngang và phân đoạn dọc. Phân đoạn hỗn hợp là phương pháp kết hợp giữa phân đoạn ngang và phân đoạn dọc.

-Phân đoạn ngang:

Phân đoạn ngang cơ sở: Phân đoạn ngang cơ sở tập chung ở các hàng của bảng. Quan hệ tổng thể sẽ được chia thành các quan hệ con có cùng tập thuộc tính nhưng số lượng các hàng là nhỏ hơn.

Chú ý là mỗi hàng của quan hệ thuộc một và chỉ một đoạn.

Ví dụ: Cho quan hệ J có cấu trúc như sau:

J:

JNO	JNAME	BUDGET	LOCATION
J1	Jonh	15 000	New York
J2	Mary	10 000	Paris
J3	Bill	12 000	Montreal
J4	Clark	17 000	Paris

Thực hiện phân đoạn ngang cơ sở thành hai quan hệ J1 và J2:

J1:

JNO	JNAME	BUDGET	LOCATION
J1	Jonh	15 000	New York
J4	Clark	17 000	Paris

J2:

JNO	JNAME	BUDGET	LOCATION
J2	Mary	10 000	Paris
J3	Bill	12 000	Montreal

Như vậy thực chất của quá trình phân đoạn ngang là thực hiện câu lệnh SELECT với các điều kiện cụ thể. Trong ví dụ trên câu lệnh SELECT được thực hiện là:

$$J1 = SELECT_{"BUDGET" = "BUDGET > 15\ 000"} J$$

$$J2 = SELECT_{"BUDGET" = "BUDGET < 15\ 000"} J$$

Phân đoạn ngang suy diễn: Bắt nguồn từ kết quả của quá trình phân đoạn ngang chính, phân đoạn ngang suy diễn được sử dụng để kết nối các đoạn tạo ra CSDL thích hợp cho các ứng dụng. Quá trình kết nối yêu cầu phải có thuộc tính kết nối.

Ví dụ: Cho quan hệ DIENTHOAI có cấu trúc như sau:

DIENTHOAI:

JNO	DIENTHOAI
J1	8.243.654
J2	9.564.734
J3	8.777.253
J4	8.372.564

Thực hiện phân đoạn ngang suy diễn giữa các quan hệ: J1, J2 và DIENTHOAI với trường liên kết là JNO kết quả cho ta hai quan hệ J1_DIENTHOAI và J2_DIENTHOAI như sau:

J1-DIENTHOAI:

JNO	JNAME	BUDGET	LOCATION	DIENTHOAI
J1	Jonh	15 000	New York	8.243.654
J4	Clark	17 000	Paris	9.564.734

J2-DIENTHOAI:

JNO	JNAME	BUDGET	LOCATION	DIENTHOAI
J2	Mary	10 000	Paris	8.777.253
J3	Bill	12 000	Montreal	8.372.564

Như vậy thực chất của quá trình phân đoạn ngang suy diễn là thực hiện phép nửa kết nối từ kết quả của quá trình phân đoạn ngang cơ sở cùng quan hệ mà ta cần kết nối. Trong ví dụ trên quan hệ SV1_DIEM và SV2_DIEM là kết quả của hai phép thực hiện sau:

J1_DIENTHOAI = DIENTHOAI SJ $_{\text{" JNO = JNO "}}$ J1

J2_DIENTHOAI = DIENTHOAI SJ $_{\text{" MASV = MASV "}}$ J2

-Phân đoạn dọc: Phân đoạn tập chung ở các thuộc tính, trong các thuộc tính của quan hệ chọn ra thuộc tính kết nối. Kết quả thu được là một tập các quan hệ con, chúng có thể kết nối lại tạo thành quan hệ tổng thể.

Ví dụ: Thực hiện phân đoạn dọc với thuộc tính liên kết là JNO từ quan hệ J2-DIENTHOAI, ta thu được hai quan hệ QH1 và QH2 như sau:

QH1:

JNO	JNAME	BUDGET
J2	Mary	10 000
J3	Bill	12 000

QH2:

JNO	LOCATION	DIENTHOAI
J2	Paris	8.777.253
J3	Montreal	8.372.564

Quá trình phân đoạn dọc thực chất là thực hiện phép chiếu (Project) các thuộc tính của quan hệ tổng thể thành các quan hệ con. Trong ví dụ trên có hai phép chiếu được thực hiện là:

QH1 = PJ $_{\text{" JNO, JNAME, BUDGET "}}$ J2-DIENTHOAI

QH2 = PJ $_{\text{" JNO, LOCATION, DIENTHOAI "}}$ J2-DIENTHOAI

-Phân đoạn hỗn hợp: Phân đoạn hỗn hợp là sự kết hợp giữa phân đoạn ngang và phân đoạn dọc. Có hai phương pháp phân đoạn hỗn hợp là:

1. Thực hiện phân đoạn ngang trước sau đó phân đoạn dọc.
2. Thực hiện phân đoạn dọc trước sau đó phân đoạn ngang.

Quá trình được thực hiện tuần tự, kết quả thu được từ phép phân đoạn cuối cùng.

III/Quản lý giao tác

Khái niệm giao tác được sử dụng trong lĩnh vực CSDL như đơn vị cơ bản của tính toán nhất quán và đáng tin cậy (xác thực). (CSDL trong trạng thái nhất quán nếu tuân thủ theo các ràng buộc kể đến ở chương 6).
Trong quá trình thực hiện giao tác CSDL có thể tạm thời không nhất quán nhưng CSDL phải nhất quán khi giao tác kết thúc. Tính tin cậy dựa vào cả hai khả năng sau:

Khả năng phục hồi nhanh của hệ thống khi nhiều kiểu lỗi xẩy ra. (Khi các lỗi xẩy ra hệ thống có thể chịu đựng được và có thể tiếp tục cung cấp các dịch vụ.)

Khôi phục: đạt được trạng thái nhất quán. Trở về trạng thái nhất quán trước đó hoặc tiếp tới trạng thái nhất quán mới sau khi xẩy ra lỗi.).
Nhất quán giao tác liên quan tới sự thực hiện các truy nhập trùng nhau.
Việc quản lý giao tác tiếp xúc với các vấn đề luôn giữ CSDL trong trạng thái nhất quán khi xẩy ra các truy nhập trùng nhau và các lỗi.

1/Định nghĩa:

Giao tác là một dãy các hành động được thực hiện bởi một chương trình ứng dụng hay bởi một người sử dụng, mà hoặc phải được thực hiện hoàn toàn hoặc là không được thực hiện một hành động nào.

a.Các điều kiện kết thúc của giao tác:

Một các giao tác luôn luôn kết thúc. Nếu giao tác có thể hoàn thành toàn bộ công việc của nó thành công chúng ta nói giao tác chuyển giao (Commit), ngược lại nếu một giao tác dừng lại không với sự hoàn thành các công việc của nó chúng ta nói giao tác bị loại bỏ (Abort). Một giao tác bị loại bỏ số nguyên nhân:

-Một giao tác bị loại bỏ bởi chính nó vì một điều kiện không thoả mãn cấm không cho giao tác hoàn thành các công việc của nó.

-DBMS loại bỏ giao tác, ví dụ khoá chết hoặc các điều kiện khác.

Khi một giao tác bị loại bỏ các việc thực hiện của nó bị dừng lại và toàn bộ việc đã thực hiện được loại bỏ để đưa CSDL về trạng thái trước khi thực hiện giao tác. Điều này cũng được hiểu như rollback.

b.Các đặc điểm của giao tác:

ReadSet (RS): tập hợp các mục dữ liệu một giao tác đọc.

WriteSet (WS): tập hợp các mục dữ liệu một giao tác ghi.

BaseSet (BS) = RS U WS.

RS và WS không nhất thiết phải loại trừ lẫn nhau. RS, WR sử dụng như cơ sở để mô tả đặc điểm của một giao tác.

2/Các thuộc tính của giao tác :

a.Tính nguyên tố: hoặc là tất cả các hành động, hoặc là không một hành động nào của giao tác được thực hiện. Tính nguyên tố qui định rằng một giao tác bị ngắt bởi một sự cố nào đó thì những kết quả của các lệnh thực thi giao tác đó đã và đang được thực hiện phải bị loại bỏ. Có hai lý do chính khiến một giao tác không được thực hiện hoàn toàn đó là giao tác bị loại bỏ và hệ thống có sự cố. Một giao tác bị loại bỏ nguyên nhân có thể là do yêu cầu từ chính bản thân giao tác đó, có thể do người sử dụng (do một số thông tin đầu vào bị sai, một số điều kiện không được thoả mãn.) và có thể do yêu cầu của hệ thống (do quá tải, tắc nghẽn).

b.Nhất quán:

Bốn mức nhất quán:

Mức 3: Giao tác T nhìn mức nhất quán 3 nếu:

T không ghi đè dữ liệu nháp của giao tác khác

T không chuyển giao bất cứ một việc ghi nào đến khi nó hoàn thành hoàn toàn việc ghi của nó (đến khi kết thúc giao tác EOT).

T không đọc dữ liệu nháp từ các giao tác khác.

Các giao tác khác không nháp vào bất cứ dữ liệu nào đọc bởi T trước khi T hoàn thành.

Mức 2:

T không ghi đè lên dữ liệu nháp của giao tác khác.

T không chuyển giao bất kỳ việc ghi nào trước EOT.

T không đọc dữ liệu nháp từ giao tác khác.

Mức 1:

T không ghi đè lên dữ liệu nháp của giao tác khác.

T không chuyển giao bất kỳ việc ghi nào trước EOT.

Mức 0:

T không ghi đè lên dữ liệu nháp của giao tác khác.

c.Tính trình tự: Nếu nhiều giao tác được thực hiện đồng thời thì kết quả của mỗi giao tác phải như thể là các giao tác đó được thực hiện

một các tuần tự (như thể là một giao tác được thực hiện một cách liên tục). Các hành động đảm bảo tính trình tự của các giao tác được gọi là điều khiển tương tranh.

d.Tính biệt lập: Một giao tác khi đang được thực hiện(chưa được chuyển giao) thì các giao tác khác không thể sử dụng các kết quả trung gian của các giao tác này. Tính chất này là cần thiết để tránh vấn đề mất kết quả cập nhật và vấn đề loại bỏ dây chuyền các giao tác.

Ví dụ về vấnđề mất kết quả cập nhật: Giả sử có hai giao tác như sau:

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

Dẫy thực hiện các thao tác này có thể như sau:

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

Với giá trị của x ban đầu là 20 T_2 đọc giá trị 21 và kết quả cuối cùng (nếu cả hai gia tác chuyển giao thành công) là 22.

Hoặc:

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

Cũng với giá trị x ban đầu là 20 T_2 đọc giá trị không chính xác là 20 và kết quả cuối cùng (nếu cả hai giao tác chuyển giao thành công) là 21 (Điều này đồng nghĩa với việc kết quả cập nhật của T_1 bị mất.).

Tính độc lập tuỳ theo mức nhất quán:

Mức 0: Giao tác chuyển giao trước khi tất cả các việc ghi chuyển giao do đó nếu xẩy ra lỗi thì đòi hỏi phải cập nhật lại.

Mức 2: Tránh loại bỏ dây truyền.

Mức 3: Cung cấp đầy đủ tính biệt lập cho phép các giao tác xung đột đợi đến khi một giao tác trong số chúng kết thúc.

e.Tính bền vững: Mỗi khi giao tác được chuyển giao (được thực hiện hoàn toàn) thì hệ thống phải đảm bảo chắc chắn kết quả sẽ không bị ảnh hưởng bởi các lỗi đến sau.

3/Các loại giao tác: Theo một số chuẩn

-Vùng ứng dụng:

Giao tác thông thường (regular): cập nhật dữ liệu trên một vị trí.

Giao tác phân tán: thao tác trên dữ liệu phân tán.

Giao tác compensating:

Giao tác không thuần nhất: trong môi trường không thuần nhất.

-Khoảng thời gian làm việc:

Giao tác trực tuyến (on-line): thời gian trả lời là rất ngắn.

Giao tác gói (batch): thời gian trả lời dài (hàng phút, hàng ngày).

Giao tác đàm thoại: được thực hiện bằng việc tác đọng qua lại với người sử dụng.

-Cấu trúc:

Giao tác đơn giản: có một điểm bắt đầu, một thân giao tác, một điểm kết thúc (chuyển giao hoặc huỷ bỏ).

Giao tác lồng nhau:

4/Kiến trúc:

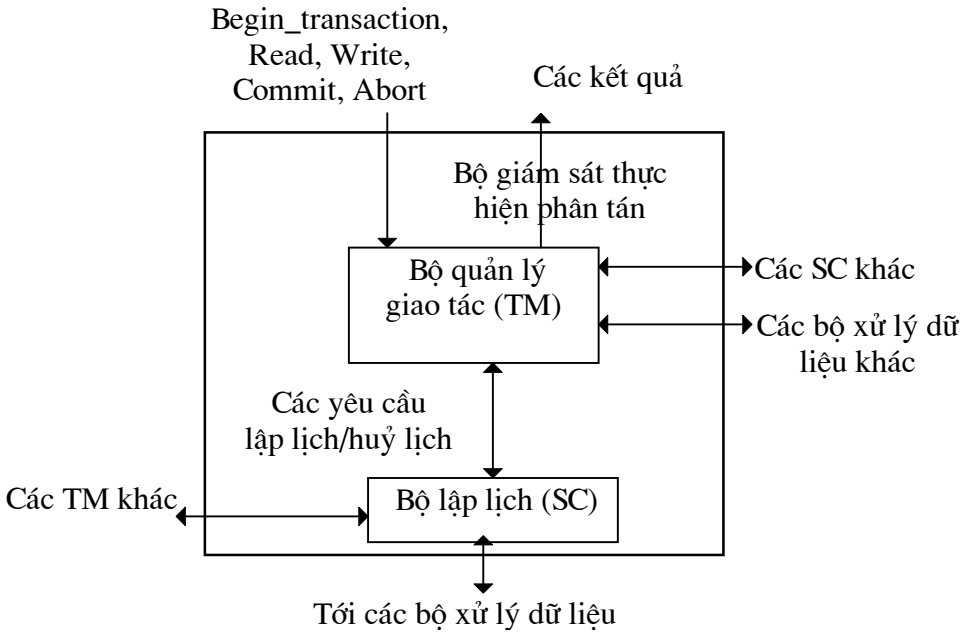
Bộ quản lý giao tác (TM): Thực hiện các thao tác CSDL thay mặt cho ứng dụng.

Bộ lập lịch (Scheduler_SC): Là trách nhiệm cho việc thực hiện một thuật toán điều khiển tương tranh cho đồng bộ các truy nhập vào CSDL.

Tham dự việc quản lý giao tác là hệ quản lý phục hồi giao tác địa phương trên mỗi vị trí.

5 lệnh của một giao tác:

Begin_Transaction, Read, Write, Commit, Abort.



Mô hình chi tiết bộ giám sát thực hiện phân tán

IV/Điều khiển tương tranh phân tán:

Điều khiển tương tranh giao thiệp với tính độc lập và nhất quán của giao tác. Điều khiển tương tranh đảm bảo tính nhất quán. Các thuật toán điều khiển tương tranh chia làm hai loại: Pessimistic và Optimistic.

1/Nguyên tắc phân loại các cơ chế điều khiển tương tranh:

Optimistic: Số giao tác xung đột không nhiều lắm. Trễ việc đồng bộ các giao tác đến khi kết thúc chúng.

Pessimistic: Sẽ có nhiều giao tác xung đột. Đồng bộ việc thực hiện các giao tác tương tranh sớm trong chu kỳ sống việc thực hiện chúng.

Dựa vào khoá (Lock based): Việc đồng bộ các giao tác đạt được bằng cách khoá logic hay vật lý trên phần hoặc hạt nhỏ của CSDL. Kích thước của các phần này là một vấn đề quan trọng. Tuy nhiên, trong các bàn luận tiếp sau chúng ta sẽ bỏ qua điều này và coi hạt chọn là một đơn vị khoá (lock unit). Lớp này chia nhỏ theo việc quản lý khoá:

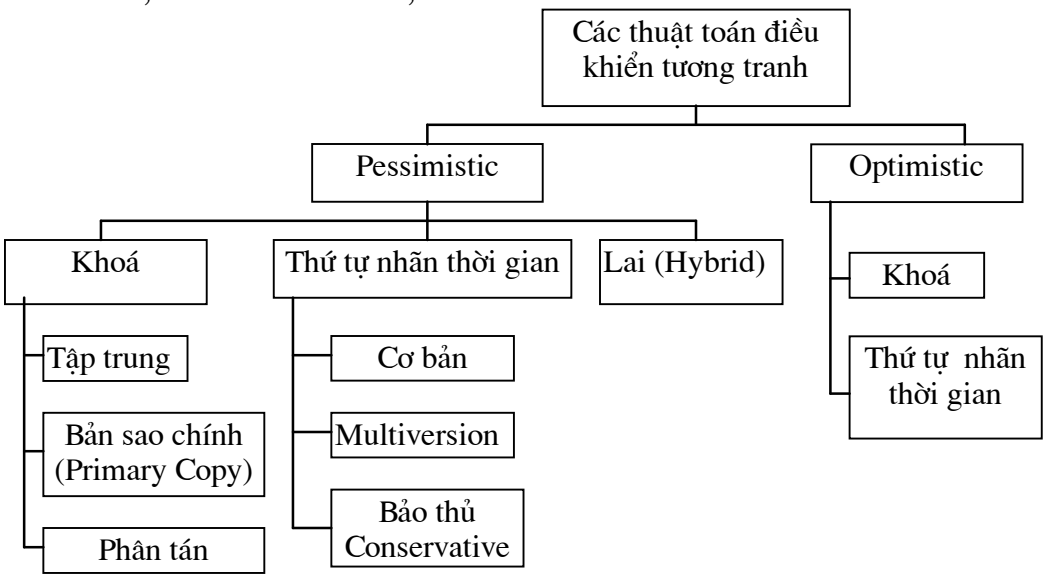
Khoá tập trung: Một vị trí trên mạng được thiết kế như vị trí chính nơi các bảng khoá cho toàn bộ CSDL được cất giữ và gánh vác nhiệm vụ phân phối các khoá cho các giao tác.

Khoá bản sao chính: Một bản sao trong các bản sao (nếu có nhiều bản sao) của mỗi đơn vị khoá sẽ được thiết kế như bản sao chính, và nó giữ khoá cho các ý định truy nhập vào phần này (muốn truy nhập vào bất kỳ bản sao nao của đơn vị khoá này phải giành được khoá của bản sao chính). Nếu CSDL là không sao bản (chỉ có duy nhất một bản cho

mỗi đơn vị khoá), các cơ chế khoá bản sao chính phân tán trách nhiệm quản lý khoá giữa một số vị trí.

Khoá không tập trung: Nhiệm vụ khoá được chia sẻ cho toàn bộ các vị trí của một mạng. Trong trường hợp này, việc thực hiện một giao tác bao gồm việc tham dự và cùng phối hợp của các bộ lập lịch tại nhiều hơn một vị trí. Mỗi một bộ lập lịch địa phương có trách nhiệm cho các đơn vị khoá cục bộ tại vị trí đó. (Một truy nhập vào CSDL phải dành được khoá trên toàn bộ các vị trí trong trường hợp sao bản.).

Thứ tự nhãn thời gian (TO): Bao gồm việc tổ chức thứ tự thực hiện các giao tác đảm bảo tính nhất quán tác động qua lại lẫn nhau. Thứ tự này được duy trì bởi việc phân chia các nhãn thời gian cho cả các giao tác và các mục dữ liệu được lưu trữ trong CSDL. Các thuật toán này có thể là: Basic TO, multiversion TO, conservative TO.



Hình 2.III.2 Sự phân lớp các điều thuật toánkhiển tương tranh

2/Khoá hai pha (Two-phase locking):

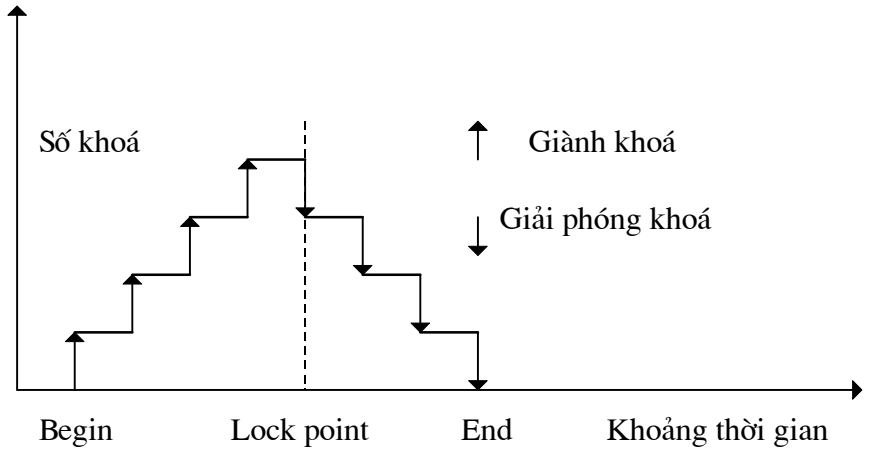
Pha mở rộng: Giai đoạn giao tác dành khoá và truy nhập các mục dữ liệu.

Pha thu hẹp: Giai đoạn giải phóng khoá.

Luật:

Một giao tác không đòi một khoá khi đã giải phóng một khoá.

Một giao tác không giải phóng một khoá khi nó chưa chắn là không đòi khoá nào nữa.



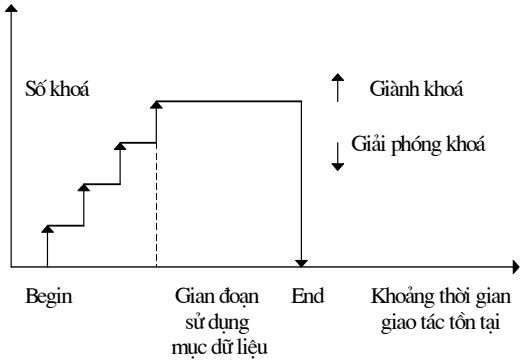
Đồ thị khoá hai pha

Khó khăn:

Xác định thời điểm giữa hai pha.

Có thể dẫn đến loại bỏ dây truyền khi một giao tác phải loại bỏ vì sử dụng dũ liệu khoá sau khi giao tác này giải phóng và bị loại bỏ.

Khoá hai pha nghiêm ngặt: Chỉ giải phóng khoá khi giao tác kết thúc.



Đồ thị khoá hai pha nghiêm ngặt

a.Khoá hai pha tập trung:

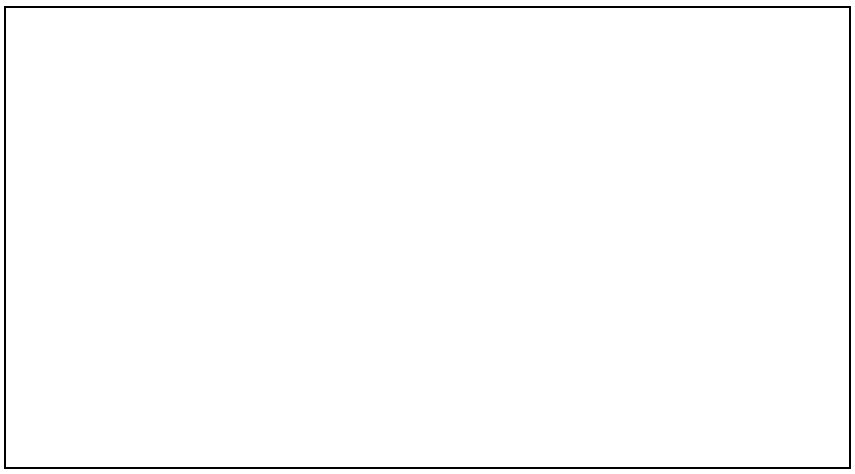
Một vị trí có trách nhiệm quản lý khoá uỷ quyền. Cách tiếp cận này cũng có thể hiểu như thuật toán Primary 2PL.

Chỉ một vị trí có hệ quản lý khoá, các hệ quản lý giao tác tại các vị trí khác giao tiếp với nó.

Sự giao tiếp giữa các vị trí đồng thao tác trong việc thực hiện một giao tác theo một thuật toán C2PL (Centralized 2PL) như hình vẽ dưới đây.

Sự giao tiếp giữa các hệ quản lý giao tác tại vị trí giao tác được khởi tạo (gọi là đồng phối hợp quản lý giao tác.), quản lý khoá tại vị trí trung tâm và các bộ xử lý dữ liệu (data processor) tại các vị trí cùng tham gia khác.

Điểm khác biệt quan trọng giữa thuật toán C2PL-TM và thuật toán 2PL-TM (hình 11.3) thuật toán thực hiện một giao thức điều khiển bản sao nếu CSDL được sao bản. C2PL-LM cũng khác S2PL. Hệ quản lý khoá tập trung không gửi các thao tác tới các bộ xử lý dữ liệu riêng, cách được thực hiện bởi bộ quản lý giao tác đồng phối hợp. Một nhược điểm chung của thuật toán C2PL là một tắc nghẽn có thể tạo thành rất nhanh xung quanh vị trí trung tâm. Hơn nữa, hệ thống có thể giảm độ tin cậy từ lỗi hoặc không truy nhập được vào vị trí trung tâm có thể là nguyên nhân gây một lỗi hệ thống lớn.



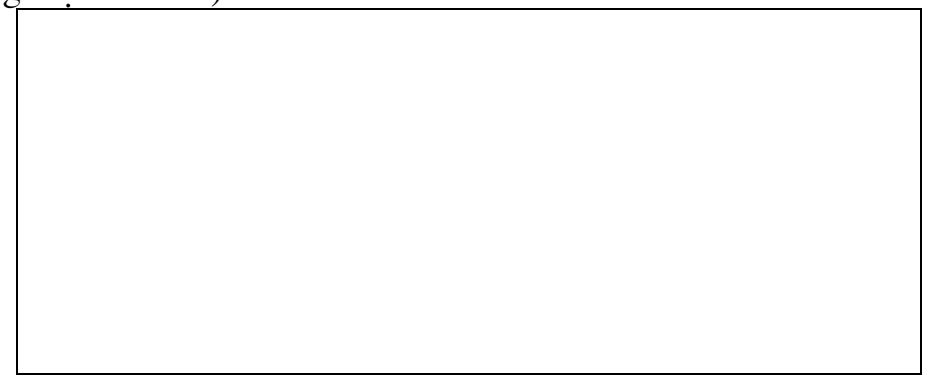
b.Khoá hai pha bản sao chính (Primary copy 2PL):

Phát triển trực tiếp từ C2PL trong một cố gắng chống lại các vấn đề bàn luận ở trên. Về cơ bản, các bộ quản lý khoá ở một số vị trí và mỗi bộ quản lý khoá có trách nhiệm quản lý các khoá nắm giữ tập các đơn vị khoá. Các bộ quản lý giao tác gửi các yêu cầu khoá và không khoá của nó tới các bộ quản lý khoá, các bộ quản lý khoá có trách nhiệm chỉ ra đơn vị khoá. Thuật toán này đối xử một bản sao của mỗi danh mục dữ liệu như là bản sao chính của nó. Chúng ta không đưa ra chia tiết thuật toán này từ sự sửa đổi thuật toán C2PL. Cơ bản chỉ có một thay đổi là việc định vị bản sao chính được chỉ ra cho từng danh mục trước khi gửi yêu cầu khoá hay không khoá cho bộ quản lý khoá tại vị trí này. Đây là một thiết kế quản lý từ điển được đưa ra thảo luận trong chương 4.

c.Khoá hai pha phân tán (Distributed 2PL):

D2PL chờ đợi tính sẵn sàng của các bộ quản lý khoá tại từng vị trí trong CSDL không sao bản, D2PL suy thoái thành thuật toán Primary copy 2PL. Nếu CSDL được sao bản, giao tác thực hiện giao thức điều khiển sao bản ROWA.

Sự liên kết giữa các vị trí đồng thao tác thực hiện các giao tác giao tác theo giao thức D2PL được mô tả ở hình 11.10 (không trình bầy ứng dụng luật ROWA).



Thuật toán quản lý giao tác D2PL là tương tự với thuật toán C2PL-TM với hai sửa đổi chính:

Thông báo gửi tới bộ quản lý khoá vị trí trung tâm trong C2PL-TM được gửi tới bộ quản lý khoá trên toàn bộ vị trí tham gia. Trong D2PL-TM các thao tác không qua bộ xử lý dữ liệu bởi bộ quản lý giao tác đồng phối hợp nhưng bởi các bộ quản lý khoá tham gia. Điều này nghĩa là bộ quản lý giao tác đồng phối hợp không đợi một thông báo yêu cầu cấp khoá ("lock request granted"). Một điểm khác là bộ xử lý dữ liệu tham gia gửi thông báo kết thúc thao tác ("end of operation") tới bộ đồng phối hợp quản lý giao tác. Cách chọn lựa là mỗi một bộ xử lý dữ liệu gửi tới bộ quản lý khoá nó sở hữu, bộ quản lý khoá có thể giải phóng các khoá và thông báo cho bộ đồng quản lý giao tác. Do các sự tương tự, chúng ta không đưa ra các thuật toán Distributed TM và Distributed LM ở đây.

4/Các thuật toán điều khiển tương tranh dựa trên nhãn thời gian (Timestamp-Based):

Không giống như các thuật toán dựa trên khoá, các thuật toán dựa vào nhãn thời gian không cố gắng đảm bảo tính tuần tự bởi sự loại trừ lẫn nhau. Thay thế, chúng ta chọn một thứ tự tuần tự ưu tiên và thực hiện các giao tác theo thứ tự đó. Để kiến tạo thứ tự này, bộ quản lý giao tác chia từng giao tác T_i một nhãn thời gian duy nhất $ts(T_i)$ tại thời điểm khởi tạo giao tác đó.

Một nhãn thời gian là một định danh đơn giản cái máy chủ nhận ra từng giao tác duy nhất và cho phép việc sắp thứ tự. Có hai thuộc tính

duy nhất và đơn điệu. Hai nhãn thời gian được sinh bởi cùng một cùng một bộ quản lý giao tác phải đơn điệu tăng.

Có một số cách đánh nhãn thời gian:

Dùng bộ đếm toàn cục đơn điệu tăng: Trong hệ phân tán đây là một vấn đề không dễ giải quyết.

Tại mỗi vị trí có một bộ đếm (tự quản việc đánh nhãn thời gian). Nhãn thời gian là một bộ đôi giá trị <giá trị đếm cục bộ, định danh vị trí>. Chú ý rằng định danh vị trí được kèm theo tối thiểu vị trí có nghĩa. Sau đây chỉ kể đến việc đánh thứ tự cho các nhãn thời gian của hai giao tác khác nhau, nhãn giống như giá trị cục bộ. Nếu mỗi hệ thống có thể truy nhập vào hệ thống khoá sở hữu bởi nó, có thể sử dụng các giá trị khoá hệ thống thay thế cho các giá trị đếm.

Bình thường, thứ tự nhãn thời gian (TO) có các luật sau:

Hai thao tác xung đột O_{ij} và O_{kl} thuộc về thao tác T_i và T_k ; O_{ij} được thực hiên trước O_{kl} khi và chỉ khi $ts(T_i) < ts(T_k)$. Trong trường hợp này T_i được nói là giao tác cũ hơn và T_k là giao tác mới hơn.

Một bộ lập lịch làm hiệu lực các luật TO kiểm tra từng thao tác mới dựa trên các thao tác xung đột, các thao tác đã được lập lịch. Nếu thao tác mới thuộc về một giao tác trẻ hơn toàn bộ các giao tác xung đột đã được lập lịch, thao tác này được chbấp nhận; ngược lại nó bị loại bỏ, nguyên nhân toàn bộ giao tác khởi động lại với một nhãn thời gian mới.

Một bộ lập lịch thứ tự nhãn thời gian điều khiển trong cách này được đảm bảo để sinh ra một lịch tuần tự. Tuy nhiên, sự so sánh giữa các nhãn thời gian giao tác có thể được thực hiện chỉ nếu lịch nhận được toàn bộ các thao tác để được lập lịch. Nếu các thao tác đưa tới bộ lập lịch tại một thời điểm (trường hợp thực tế), nó là cần thiết để có thể tìm ra nếu một thao tác xẩy ra ngoài sự nối tiếp. Dể dễ dàng việc kiểm tra này, từng mục dữ liệu x được đánh hai nhãn thời gian: nhãn thời gian đọc [rts(x)], nhãn rộng nhất của các nhãn thời gian của các giao tác đọc x, và nhãn thời gian ghi [wts(x)], nhãn rộng nhất của các nhãn thời gian của các giao tác ghi x. Bây giờ nó đủ khả năng so sánh nhãn thời gian của một thao tác với các nhãn thời gian của mục dữ liệu cái nó truy nhập để chỉ ra nếu bất kỳ giao tác với một nhãn rộng nhất đã sẵn sàng truy nhập vào cùng một mục dữ liệu.

a.Thuật toán TO cơ bản:

Thuật toán TO cơ bản là việc thực hiện trực tiếp các luật TO. Bộ quản lý giao tác đồng phối hợp đánh nhãn từng giao tác, xác định các vị trí từng mục dữ liệu được cất giữ, và gửi các thao tác thích hợp tới các vị trí này. Các bộ lập lịch tại từng vị trí đơn giản làm hiệu lực các luật TO.

Như đã chỉ ra, một giao tác chứa các thao tác bị loại bỏ bởi một bộ lập lịch được khởi động lại bởi bộ quản lý giao tác với một nhãn thời gian mới. Điều này đảm bảo rằng giao tác có một cơ hội để thực hiện trong cố gắng tiếp theo của nó. Từ việc các giao tác không khi nào đợi khi chúng giữ quyền truy nhập vào các mục dữ liệu, thuật toán TO cơ bản không bao giờ dẫn đến các khoá chết. Tuy vậy, hậu quả của việc tránh khỏi khoá chết là khả năng phải khởi động lại một giao tác nhiều lần. Việc giảm số lần khởi động này chúng ta sẽ lưu tâm ở đoạn sau.

Một chi tiết khác cần được lưu tâm liên quan đến việc liên kết giữa bộ lập lịch và bộ xử lý dữ liệu. Khi một thao tác được chấp nhận được đi tiếp tới bộ xử lý dữ liệu, bộ lập lịch cần giữ lại việc gửi một thao tác không phù hợp khác, nhưng thao tác có thể được chấp nhận đối với bộ xử lý dữ liệu đến khi thao tác đầu tiên được xử lý và được báo nhận. Có một yêu cầu để đảm bảo bộ xử lý dữ liệu thực hiện các thao tác trong một thứ tự giống thứ tự bộ lập lịch thông qua chúng. Ngược lại, các giá trị nhãn thời gian đọc và ghi cho truy nhập mục dữ liệu có thể không được chính xác.

VD 11.8 (page 303)

Bộ lập lịch có thể hiệu lực thứ tự bằng cách duy trì một hàng đợi cho từng mục dữ liệu, hàng đợi này được sử dụng để trễ sự truyền của các thao tác đã được chấp nhận đến khi một báo nhận được nhận từ bộ xử lý dữ liệu về thao tác trước trên cùng một mục dữ liệu. Chi tiết này không thể hiện trong thuật toán BTO-SC.

Như một sự phức tạp không xuất hiện trong thuật toán 2PL cơ bản bởi vì bộ quản lý khóa sắp xếp có hiệu quả các thao tác bằng việc giải phóng khóa chỉ sau khi thao tác được thực hiện. Trong cảm giác hàng đợi được bộ lập lịch TO duy trì có thể nghĩ đến như một khóa. Tuy vậy, điều này không bao hàm rằng lịch được sinh bởi một bộ lập lịch TO và một bộ lập lịch 2PL có thể luôn luôn ngang bằng. Có một số lịch được một bộ lập lịch TO sinh ra có thể không được chấp nhận bởi một lịch 2PL.

Nhớ rằng trong trường hợp các thuật toán 2PL nghiêm ngặt, việc giải phóng các khóa được trễ lâu hơn đến khi giao tác commit hoặc abort.

b.Thuật toán TO bảo thủ (Conservative):

Chúng ta đã chỉ ra trong đoạn trước thuật toán TO cơ bản không bao giờ dẫn đến các thao tác để đợi, nhưng thay vào đó, khởi động lại chúng. Chúng ta cũng chỉ ra rằng thuật toán TO cơ bản có ưu điểm là không gây ra khóa chết, tuy nhiên gặp một vấn đề là khởi động lại một giao tác nhiều lần. Thuật toán Conservative TO cố gắng giảm số lần khởi động lại.

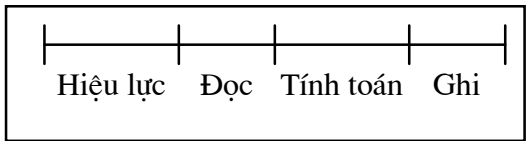
Đầu tiên chúng ta giới thiệu một kỹ thuật chung được sử dụng để giảm khả năng khởi động lại. Nhớ rằng, một bộ lập lịch TO khởi

động lại một giao tác nếu một giao tác xung đột trẻ hơn đã được lập lịch hoặc vừa được thực hiện.

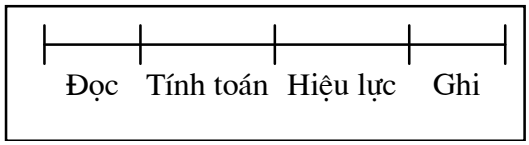
5/Các thuật toán điều khiển tương tranh Optimistic:

Các thuật toán điều khiển tương tranh bàn luận ở trên là các thuật toán Pessimistic. Các thuật toán này không cho phép một giao tác truy nhập vào một mục dữ liệu nếu một giao tác xung đột đang truy nhập vào mục dữ liệu đó.

Trong các thuật toán Optimistic, chúng ta chia một thao tác bất kỳ của một giao tác thành các pha tuần tự sau: pha hiệu lực (V: Validation), pha đọc (R: Read), pha tính toán (C: Computation), pha ghi (W: Write). Các thuật toán Optimistic trễ pha hiệu lực đến trước pha viết (Hình vẽ). Theo cách đó một thao tác đã đệ trình tới một bộ lập lịch Optimistic không bao giờ bị trễ. Các thao tác đọc, tính toán, ghi không bị bó buộc trong hành động cập nhật CSDL. Pha hiệu lực bao gồm việc kiểm tra xem các cập nhật này đảm bảo tính nhất quán của CSDL. Nếu trả lời được lựa chọn thì thay đổi được thực hiện trên toàn cục. Ngược lại, giao tác bị hủy bỏ và khởi động lại.



Các pha của việc thực hiện giao tác Pessimistic



Các pha của việc thực hiện giao tác Optimistic

Có thể thiết kế các thuật toán điều khiển tương tranh Optimistic dựa trên khóa. Chúng ta chỉ mô tả cách tiếp cận Optimistic sử dụng nhãn thời gian. Chúng ta chỉ bàn luận một cách vắn tắt và nhấn mạnh các khái niệm hơn là thực hiện chi tiết vì hai lý do sau:

Hầu hết các công việc hiện nay trên phương pháp Optimistic tập trung chủ yếu trên CSDL tập trung hơn là trên CSDL phân tán.

Các thuật toán Optimistic không được thực hiện trên bất kỳ một môi trường thương mại hoặc giao thức DBMS nào.

Khác với các thuật toán Pessimistic dựa trên nhãn thời gian không chỉ bởi được tối ưu mà còn trong việc đánh các nhãn thời gian. Nhãn thời gian chỉ được liên kết với giao tác, không liên kết với các mục dữ liệu (không có nhãn thời gian đọc và ghi). Hơn nữa các nhãn thời gian không được gán cho các giao tác vào thời điểm khởi tạo chúng mà vào thời điểm bắt đầu bước hiệu lực của chúng. Bởi vì các nhãn thời gian chỉ cần đến trong pha

hiệu lực, và như chúng ta sẽ thấy, việc đánh nhãn thời gian sớm có thể dẫn đến sự loại bỏ giao tác không cần thiết.

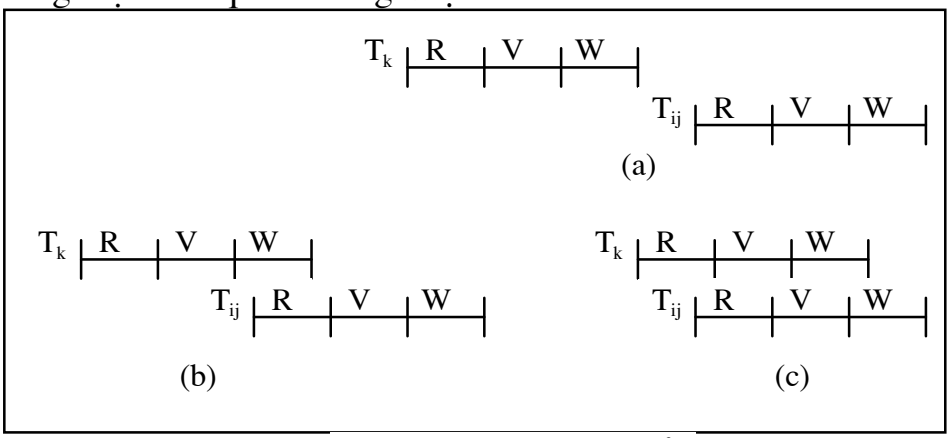
Mỗi giao tác T_i được chia nhỏ bởi bộ quản lý giao tác tại vị trí bắt đầu thành một số các giao tác con, mỗi giao tác con có thể được thực hiện trên nhiều vị trí. Qui ước T_{ij} là một giao tác con của T_i thực hiện trên vị trí j. Tại thời điểm một nhãn thời gian được gán cho giao tác, nhãn thời gian sẽ được sao cho cho toàn bộ các giao tác con của nó. Việc hiệu lực cục bộ của giao tác T_{ij} được thực hiện theo các luật sau, cái loại trừ lẫn nhau.

Luật 1: Nếu toàn bộ các giao tác T_k mà $ts(T_k) < ts(T_{ij})$ đã hoàn thành pha ghi của chúng trước khi T_{ij} bắt đầu pha đọc (hình vẽ), việc hiệu lực thành công, bởi vì các việc thực hiện giao tác trong một thứ tự tuần tự.

Luật 2: Nếu có một giao tác T_k mà $ts(T_k) < ts(T_{ij})$ hoàn thành pha ghi của nó khi T_{ij} đang trong pha đọc, hiệu lực hoàn thành nếu $WS(T_k) \cap RS(T_{ij}) = 0$.

Luật 3: Nếu có một giao tác T_k mà $ts(T_k) < ts(T_{ij})$ hoàn thành pha đọc của nó trước khi T_{ij} hoàn thành pha đọc, hiệu lực thành công nếu $WS(T_k) \cap WS(T_{ij}) = 0$.

Một giao tác được hiệu lực cục bộ địa phương để đảm bảo tính nhất quán CSDL địa phương, giao tác cũng cần được hiệu lực toàn cục để đảm bảo rằng luật nhất quán chung được tuân theo.



Cách thức thực hiện có thể

6/Quản lý khóa chết:

Một công cụ hiệu quả để phân tích các khóa chết là đồ thị chờ đợi (WFG: Wait-For-Graph, chỉ ra quan hệ chờ đợi giữa các thao tác). Khi WFG chứa một chu trình kín thì có khóa chết có ba phương pháp được biết để quản lý khóa chết.

a.Phòng ngừa khóa chết:

Phương pháp phòng ngừa không cho phép các khóa chết xảy ra trong điểm (place) đầu tiên. Bộ quản lý giao tác kiểm tra một giao tác

khi nó được khởi tạo lần đầu và không cho phép nó tiếp tục nếu nó có thể dẫn đến khóa chết. Để thực hiện việc kiểm tra này, nó được yêu cầu tất cả mục dữ liệu sẽ được truy nhập bởi một giao tác được khai báo trước. Bộ quản lý giao tác cho phép một giao tác tiếp tục nếu toàn bộ mục dữ liệu cần truy nhập sẵn sàng. Ngược lại, giao tác không được phép tiếp tục. Bộ quản lý giao tác dành riêng toàn bộ mục dữ liệu được khai báo trước bởi một giao tác đã cho phép để tiến hành.

Không may, các hệ thống như vậy không phù hợp cho các môi trường CSDL. Vấn đề cơ bản là thường rất khó để biết chính xác các mục dữ liệu nào được truy nhập bởi một giao tác. Truy nhập vào các mục dữ liệu nhất định có thể phụ thuộc vào các điều kiện mà các điều kiện này có thể không được giải quyết đến thời gian chạy. Để được an toàn, hệ thống còn cần quan tâm số tối đa tập các mục dữ liệu. Nói cách khác, các hệ thống như vậy yêu cầu hỗ trợ không trong thời gian chạy, cái giảm tổng chi phí. Nó còn kèm thêm một ưu điểm là không cần thiết loại bỏ và khởi động lại khi khóa chết.

b.Tránh khóa chết:

Dùng các kỹ thuật điều khiển tương tranh không bao giờ dẫn đến khóa chết hoặc yêu cầu các bộ lập lịch phát hiện vị trí có khả năng lỗi trước và đảm bảo khóa chết sẽ không xẩy ra. Chúng ta sẽ lưu tâm cả hai trường hợp này.

Cách đơn giản nhất để tránh khóa chết là phân cấp các tài nguyên và nhấn mạnh rằng mỗi tiến trình yêu cầu truy nhập vào các tài nguyên này trong thứ tự này. Giải pháp này, được đề xuất một thời gian dài cho các hệ điều hành. Các thứ tự này có thể là toàn cục hoặc cục bộ. Trong trường hợp là cục bộ cần thiết sắp xếp các vị trí và yêu cầu các giao tác truy nhập vào các mục dữ liệu tại nhiều vị trí yêu cầu các khóa của chúng bằng việc duyệt các vị trí theo thứ tự định nghĩa trước.

Một cách chọn lựa khác, là sự sử dụng nhãn thời gian của các giao tác để định mức ưu tiên các giao tác và giải quyết các khóa chết bởi việc loại bỏ các giao tác với mức ưu tiên cao hơn (hoặc thấp hơn). Theo luật sau:

if $ts(T_i) < ts(T_j)$ thenT_i waits else T_i dies
if $ts(T_i) > ts(T_j)$ then T_j is wounded else T_i waits

c.Phát hiện và giải quyết khóa chết:

Phát hiện được thực hiện bởi việc nghiên cứu GWVFG (Global Wait-For-Graph) cho thông tin các chu trình kín. Giải quyết bằng cách chọn ra một hoặc nhiều giao tác loại bỏ để phá vỡ chu trình kín trong GWFG. Vấn đề chọn tập các giao tác có tổng giá nhỏ nhất cho việc phá vỡ chu trình kín (NP complete). Các nhân tố cho việc lựa chọn:

1.Tổng số sự cố gắng (kết quả) đã được đầu tư trong giao tác. Sự cố gắng này sẽ bị mất nếu giao tác bị hủy bỏ.

2.Giá của việc hủy bỏ giao tác. Giá này nó chung phụ thuộc vào số lần truy nhập mà giao tác đã thực hiện.

3.Tổng số lần cố gắng nó sẽ làm để hoàn thành việc thực hiện giao tác. Bộ lập lịch cần tránh loại bỏ một giao tác gần như được hoàn thành. Để làm việc này, bộ lập lịch phải có thể dự đoán cách chạy của các giao tác hiệu lực.

4.Số chu trình kín chứa giao tác. Từ đó loại bỏ một giao tác phá vỡ toàn bộ chu trình kín chứa nó, giao tác tốt nhất cho việc loại bỏ là giao tác là thành phần của nhiều chu trình kín.

c1.Phát hiện khóa chết tập trung:

Trong cách phát hiện khóa chết tập trung, một vị trí được thiết kế như một bộ phát hiện khóa chết cho toàn hệ thống. Trước tiên, mỗi bộ quản lý khóa chuyển LWFG (Local Wait-For-Graph) của nó cho bộ phát hiện khóa chết, bộ phát hiện khóa chết tạo nên GWFG và các khóa cho chu trình kín trong nó. Sự thật bộ quản lý khóa chỉ cần gửi các thay đổi trong đồ thị của nó (các cung vừa mới tạo hoặc xóa) cho bộ phất hiện khóa chết. Độ dài của các khoảng cho việc truyền các thông tin này là một quyết định của thiết kế hệ thống.

Đơn giản và có thể được chọn một thuật toánđiều khiển tương tranh rất tự nhiên là C2PL.

c2.Phát hiện khóa chết phân cấp:

Một sự lựa chọn cho việc phát hiện khóa chết tập trung là xây dựng một phân cấp các bộ phát hiện khóa chết. Các khóa chết là vị trí đơn lẻ cục bộ có thể được phát hiện tại vị trí này sử dụng LWFG. Mỗi vị trí cũng gửi LWGF tới bộ phát hiện khóa chết tại mức kế tiếp. Như vậy, các khóa chết phân tán bao gồm hai hoặc nhiều vị trí có thể được phát hiện bởi một bộ phát hiện khóa chết trong mức thấp nhất tiếp theo.

Phương pháp phát hiện khóa chết phân cấp giảm sự phụ thuộc vào vị trí trung tâm, Vì vậy giảm giá thành truyền thông. Nhược điểm là phức tạp.

c3.Phát hiện khóa chết phân tán:

Các thuật toán phát hiện khóa chết phân tán giao phó trách nhiệm phát hiện khóa chết đến từng vị trí. Như vậy, như trong phát hiện khóa chết phân cấp có các bộ phát hiện khóa chết cục bộ tại mỗi vị trí, bộ khóa chết cục bộ sẽ truyền LWFG của nó tới một vị trí khác (trên thực tế chỉ các chu trình kín có khả năng khóa chết được truyền). LWFG tại từng vị trí được tạo và sửa đổi như sau:

1.Từ mỗi vị trí nhận các chu trình có khả năng khoá chết từ các vị trí khác, các cung này được cộng vào LWFG.

2.Các cung trong LWFG cái thể hiện các giao tác cục bộ đang đợi các giao tác tại vị trí khác được kết nối với các cung trong các LWFG cái thểb hiện các giao tác ở xa đang đợi một giao tác địa phương.

Các bộ phát hiện khoá chết địa phương tìm kiếm hai điều. Nếu có một chu trình kín không kèm theo các cung ngoài, có một khoá chết địa phương có thể được kiểm soát cục bộ. Nếu một chu trình kín bao gồm các cung ngoài, có khoá chết phân tán tiềm ẩn và thông tin chu trình kín này đã được kết nối tới các bộ phát hiện khoá chết khác.

Một câu hỏi cần trả lời bộ phát hiện nào được truyền thông tin. Rõ ràng nó có thể truyền tới tất cả các bộ phát hiện khoá chết trong hệ thống. Làm thế nào để nhận biết đâu là đầu đâu là cuối của một chu trình kín khoá chết, thông tin có thể được truyền về phía trước hoặc phía sau dọc theo các vị trí trong chu trình kín khoá chết. Việc nhận vị trí sửa đổi của LWFG của nó như thảo luận trên và kiểm tra các khoá chết.

Các thuật toán phát hiện khoá chết phân tán đòi hỏi đồng bộ sự sửa đổi các bộ quản lý khoá tại mõi vị trí. Sự đồng bộ này làm chúng dễ dàng thực hiện. Tuy vậy, có các sự truyền các thông báo thừa. Xẩy ra cho ví dụ ở trên. Vị trí 1 gửi thông tin khả năng khoá chết cho vị trí 2, và vị trí 2 gửi thông tin khoá chết cho vị trí 1. Trong trường hợp này bộ phát hiện khoá chết tại cả hai vị trí sẽ phát hiện khoá chết. Bên cạnh việc dẫn đến việc truyền thông báo không cần thiết, cộng thêm khả năng mỗi vị trí có thể chọn một giao tác loại bỏ khác nhau.

Thuật toán đề xuất trong [Obermarck, 1982] giải quyết vấn đề bằng việc sử dụng nhãn thời gian giao tác theo luật sau. Cho biết đường dẫn có khả năng dẫn đến một khoá chết phân tán trong LWFG của một vị trí là $T_i \rightarrow ... \rightarrow T_j$. Một bộ phát hiện khoá chết cục bộ gửi thông tin chu trình kín chỉ nếu $ts(T_i) < ts(T_j)$. Điều này làm giảm số trung bình việc truyền thông báo bởi một phía. Trong ví dụ ở trên, vị trí 1 có đường $T_1 \rightarrow T_2 \rightarrow T_3$, vị trí 2 có đường $T_3 \rightarrow T_4 \rightarrow T_1$. Do đó 1 gửi cho 2.

7/Kết luận:

Điều khiển tương tranh phân tán cung cấp tính cô lập và nhất quán của giao tác. Đồng thời đảm bảo tính nhất quán của CSDL phân tán. Một vài bỏ sót trong phần này:

1.Thực hiện ước lượng các thuật toán điều khiển tương tranh

2.Các mô hình giao tác khác

3.Các phương pháp điều khiển tương tranh khác: Có một lớp các thuật toán điều khiển tương tranh khác được gọi là "các phương pháp kiểm tra đồ thị tuần tự ", xây dựng một đồ thị phụ thuộc và kiểm tra chu trình kín. Đồ thị phụ thuộc của một lịch S, DG(S), là một đồ thị chỉ dẫn

mô tả các quan hệ xung đột giữa các giao tác trong S là một nút. Một cung (T_i T_j) tồn tại trong DG(S) khi và chỉ khi có một thao tác khác trước đó trong T_j . (hình vẽ). Các bộ lập lịch cập nhật các DG của mình khi một trong các điều kiện sau được thi hành: (1) một giao tác mới bắt đầu trong hệ thống, (2) một giao tác đọc hoặc ghi nhận được bởi bộ lập lịch, (3) một giao tác kết thúc, (4) một giao tác bị loại bỏ.

4.assumptions abort tránactions

5.More "general" algorithms: Có thể nghiên cứu hai mẫu điều khiển tương tranh cơ bản (locking và TO) sử dụng một khung công việc đồng nhất. Ba chính đáng kể ra: (1) có thể phát triển cả hai thuật toán pessimistic và optimistic dựa trên một trong hai mẫu này; (2) một thuật toán strict TO thực hiện tương tự một thuật toán khoá từ nó trễ chấp nhận một giao tác đến khi toàn giao tác già hơn kết thúc. Điều này không có nghĩa là toàn bộ các lịch được sinh ra bởi một bộ lập lịch TO ngặt (strict TO)có thể được phép bởi một bộ lập lịch 2PL; (3) có thể phát triển thuật toán lai, xa hơn nữa có thể phát triển chính xác các luật cho tác động qua lại lẫn nhau của chúng.

6.Các mô hình thực hiện giao tác: Thực hiện tập trung và thực hiện phân tán.

V/Khôi phục

Sự phân tán và sao bản của CSDL làm tăng độ tin cậy và tính sẵn sàng của CSDL.

Tính tin cậy của hệ quản lý CSDL phân tán qui vào thuộc tính nguyên tố và bền vững của giao tác. Hai ảnh hưởng rõ ràng của các giao thức tin cậy cần được thảo luận trong sự quan hệ tới các thuộc tính này là các giao thức chuyển giao và khôi phục.

1/Các lỗi và chịu lỗi trong các hệ phân tán:

Trong đoạn này thảo luận về các lỗi trong các hệ phân tán cũng như các kỹ thuật chịu đựng lỗi được sử dụng đối phó với chúng. Dựa trên thống kê kinh nghiệm và không có nghĩa được hoàn hảo và toàn diện. Nó chỉ giúp và cung cấp một khung chung cho tính tin cậy của CSDL phân tán.

a.Nguyên nhân gây lỗi:

Tỷ lệ phần trăm các loại lỗi như sau:

Nghiên cứu tính tin cậy của hệ điều hành IBM/XA tại Stanford linear accelerator (SLAC): 57% lỗi phần cứng, 12% lỗi phần mềm, 14% lỗi các thao tác, và 17% lỗi do điều kiện môi trường.

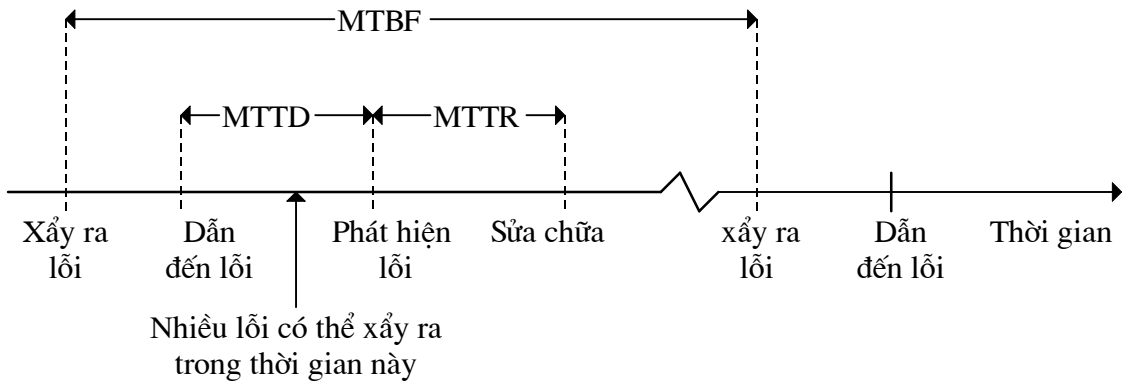
Nghiên cứu của các máy tính Tandem: Lỗi phần cứng trên 18%, lỗi phần mềm 25%, lỗi bảo quản 25%, 17% lỗi các thao tác, 15% lỗi môi trường.

Nghiên cứu của bbộ chuyển mạch số AT&T5ESS: 23,3% lỗi phần cứng, 44,3% lỗi phần mềm, 17,5% lỗi các thao tác, 5,9% không biết.

b.Các kỹ thuật và các cách chịu lỗi cơ bản:

Có hai cách cơ bản để xây dựng một hệ thống tin cậy là chịu lỗi và cấm lỗi. Chịu lỗi đưa ra cách thiết kế hệ thống chấp nhận các lỗi sẽ xẩy ra; nó cố gắng xây dựng hệ thống sao cho các lỗi có thể được phát hiện và tháo bỏ hoặc đền bù trước khi chúng có thể dẫn đến kết quảlà một hệ thống lỗi. Các kỹ thuật cấm lỗi trợ giúp đảm bảo rằng hệ thống hoàn thành sẽ không chứa bất kỳ một lỗi nào. Cấm lỗi có hai hướng. Đầu tiên là tránh lỗi, cách đưa ra các kỹ thuật sử dụng để đảm bảo chắc chắn rằng các lỗi là không mở đầu trong hệ thống. Các kỹ thuật này bao gồm các phương pháp thiết kế chi tiết (giống như thiết kế walkthrought, thiết kế các sự thanh tra ...), và điều khiển chất lượng. Hướng thứ hai của cấm lỗi là tháo bỏ lỗi, cách đưa ra các kỹ thuật được dùng để phát hiện bất kỳ lỗi nào có thể còn sót lại trong hệ thống bất chấp ứng dụng tránh lỗi và tháo bỏ lỗi các lỗi này. Các kỹ thuật đặc biệt được sử dụng trong vùng này là kiểm tra bao quát và các thủ tục làm cho hiệu lực. Chú ý rằng các kỹ thuật tháo bỏ lỗi áp dụng trong việc thực hiện hệ thống trước khi đưa ra quyền sử dụng của hệ thống. Các phạm trù cấm lỗi và tránh lỗi được sử dụng xen kẽ nhau. Một tên chung khác cho cách này là không chấp nhận lỗi. Các kỹ thuật này tập trung vào việc thiết kế các hệ thống sử dụng các thành phần tin cậy cao và phương pháp tinh vi của kỹ thuật gói bởi việc kiển tra bao quát. Như vậy, hạn chế được chờ đợi giảm sự xuất hiện của các lỗi hệ thống nhỏ nhất có thể cái các đặ điểm có thể được nắm giữ bằng tay. Không may, có một số môi trường bảo quản và sửa chữa thủ côngla không thể, hoặc thời gian cần để sửa chữa là không chấp nhận được trong các môi trường này. Thiết kế hệ thống chịu lỗi là cách chọn lựa phù hợp. Cách thứ ba cấu trúc các hệ thống tin cậylà phát hiện lỗi. Được dựa theo bất kỳ kỹ thuật chịu lỗi nào. Cách báo khi một lỗi xuất hiện nhưng không cung cấp bất kỳ biện pháp chịu lỗi nào. Vì vậy, nó có thể được dành riêng cho việc phát hiện lỗi riêng lẻ từ cách chịu lỗi nghiêm ngặt.

Điều quan trọng cần chú ý trong điểm này là các lỗi hệ thống có thể tiềm tàng. Một lỗi tiềm tàng là một lỗi được phát hiện sau thời gian nó xẩy ra. Thời kỳ này được gọi là ẩn lỗi, và thời gian ẩn lỗi trong hình trên một số các hệ thống chính được gọi là thời gian trung bình để phát hiện lỗi (MTTD). Hình vẽ quan hệ của nhiều mức tin cậy với các xuất hiện thực sự các lỗi.



Sự xuất hiện các sự kiện trên thời gian

Nguyên lý cơ bản được dùng trong toàn bộ các thiết kế hệ thống chịu lỗi là cung cấp sự dư thừa trong các thành phần hệ thống. Các thành phần dư thừa có hiệu lực một thành phần, lỗi được đền bù. Tuy nhiên, sự dư thừa là không đủ cho sự chịu đựng lỗi. Cộng thêm và là nguyên lý chịu lỗi là modularization của thiết kế. Từng thành phần của hệ thống được thực hiện như một modul với định nghĩa vào ragiao tiếp với các thành phần khác. Modul hoá có tác dụng cô lập các lỗi trong một thành phần. Đây là kỹ thuật quan trọng trong cả các hệ thống phần cứng và phần mềm.

Hai khái niệm này được dùng trong các hệ thống đặc biệt bởi cách fail-stop modules và các cặp tiến trình. Một fail-stop module tự giám sát thường xuyên, và khi nó phát hiện một lỗi, tự nó chấm dứt. Một tên khác có xu hướng các module như vậy là fail-fast. Việc thực hiện fail-stop module trong phần cứng là vượt quá phạm vi bàn luận của chúng ta, nhưng trong phần mềm chúng có thể được thực hiện bởi các chương trình được bảo vệ (defensive program). Từng phần mềm kiểm tra trạng thái của nó trong quá trình chuyển đổi trạng thái. Một lợi ích của fail-stop modules một sự giảm trong việc tiềm tàng khám phá (detection latency).

Các cặp tiến trình cung cấp sự chịu lỗi bởi sự sao bản các module phần mềm. Tư tưởng loại bỏ các điểm lỗi đơn lẻ bằng việc thực hiện mỗi dịch vụ hệ thống như hai tiến trình cái liên kết và đồng thao tác trong cung cấp dịch vụ một trong các tiến trình gọi là chính và cái kia là backup. Cả hai primary và backup được thực hiện một cách đặc biệt như fail-stop module cái đồng thao tác trong cung cấp một dịch vụ. Có một số cách khác nhau thực hiện các cặp tiến trình, phụ thuộc vào cách thực hiện kết nối giữa primary và backup. Năm kiểu chung là: lock-step, automatic checkpointing, state checkpointing, delta checkpointing và persistent.

Các cặp tiến trình yêu cầu việc liên kết giữa các tiến trình. Sự liên kết giữa các tiến trình có thể được thực hiện bằng cách chia sẻ bộ nhớ. Tuy nhiên khi thiết kế một môi trường phần mềm tin cậy điều quan trọng để thực hiện một hệ điều hành là sử dụng một cơ cấu liên kết giữa các

tiến trình dựa trên thông báo. Như một cách đóng góp vào sự cô lập lỗi từ thực hiện tiến trình trong địa chỉ trống của chính nó, và một lỗi từ trong số chúng có thể dẫn đến sự không truyền lan sang tiến trình khác.

Một điều quan trọng liên quan đến khái niệm là liên kết hướng phiên (session-oriented) giữa các tiến trình. Liên kết hướng phiên giao nhiệm vụ phát hiện và nắm giữ thông báo mất hoặc thông báo nhân bản tới các máy chủ thông báo của hệ điều hành tốt hơn tới các chương trìnhứng dụng. Điều này không chỉ dễ dàng một môi trường phát triển ứng dụng đơn giản nhưng cũng cho phép hệ điều hành để cung cấp môi trường thực hiện tin cậy cho các tiến trình ứng dụng.

3/Các lỗi trong DDBMS:

a.Các lỗi giao tác:

Giao tác có thể lỗi vì một số nguyên nhân:

- Do chính bản thân giao tác: Do mmột điều kiện cấm một công việc hoàn thành. Ví dụ: để đặt chỗ máy bay người ta có thủ tục sau:

SELECT STSOLD, CAP INTO temp1, temp2 FROM FLIGHT WHERE FNO = flight-no AND DATE = date;
IF temp1 = temp2 THEN
BEGIN OUTPUT(“Không còn chỗ trống”);
Abort;
END
ELSE BEGIN ....
Chú thích: STSOLD là số chỗ đã bán, CAP số chỗ của chuyến bay.

-Do DBMS:

+Phát hiện một khoá chết.

+Một số thuật toán không cho phép các giap tác trước hoặc ngang bằng đợi nếu dữ liệu được truy nhập bởi một giao tác khác.

+Xẩy ra một lỗi.

b.Các lỗi vi trí (hệ thống):

-Dữ liệu trong bộ nhớ mất.
-Lỗi vị trí.
-Lỗi toàn bộ các vị trí trong hệ thống phân tán.
-Lỗi một phần các vị trí trong hệ thống phân tán.

c.Các lỗi môi trường:

Các lỗi môi trường qui về các lỗi của các thiết bị lưu trữ thứ hai cái lưu trữ CSDL có thể lỗi hệ điều hành, lỗi phần cứng, lỗi các bộ điều khiển dẫn đến một phần hoặc toàn bộ CSDL có thể bị phá hủy hoặc

không truy nhập được. Khắc phục bằng sao bản và các chức năng khôi phục phân tán.

d.Các lỗi truyền thông:

Có ba kiểu lỗi mô tả ở trên chung cho cả hai CSDL tập trung và CSDL phân tán. Các lỗi truyền thông chỉ cho duy nhất trong trường hợp phân tán. Có một số kiểu lỗi truyền thông. Một kiểu chung nhất là các lỗi trong các thông báo, các thông báo có thứ tự không thích hợp, mất (hoặc không thể giao được) các thông báo, và các lỗi đường truyền. Như đã thảo luận ở chương 3, hai lỗi đầu là trách nhiệm của mạng máy tính chúng ta không quan tâm (communication subnet bao gồm physical, data link, và network layers của kiến trúc ISO/OSI). Như vậy, trong các thảo luận của chúng ta về tính tin cậy của DDBMS, chúng ta trông đợi phần mềm và phần cứng mạng máy tính ở mức dưới đảm bảo rằng hai thông báo gửi tới từ một tiến trình tại một vài vị trí khởi đầu tới tiến trình khác tại vị trí đích là nhận được không với một lỗi và thứ tự chúng đã gửi.

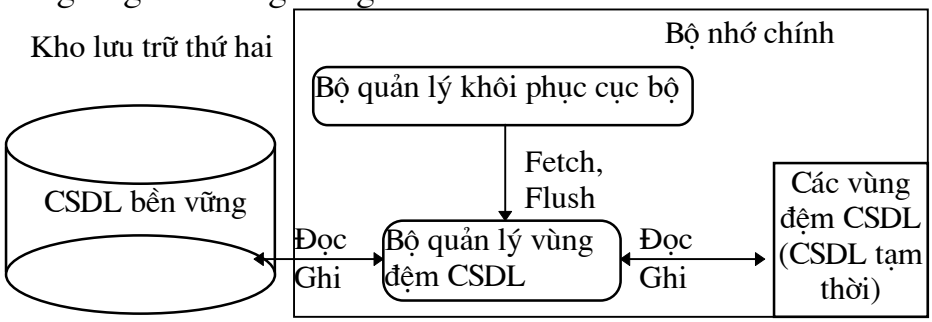
Các lỗi đường truyền dẫn đến sự phân đoạn mạng dẫn đến việc không nhất quán trong CSDL và có thể giải quyết bằng cách đặt timeout.

4/Các giao thức tin cậy cục bộ:

Trong đoạn này chúng ta bàn luận các chức năng được thực hiện bởi bộ quản lý khôi phục cục bộ (LRM) tồn tại trên từng vị trí. Các chức năng đảm bảo tính nguyên tố và tính bền vững của các giao tác cục bộ.

a.Các quan tâm kiến trúc:

Bộ quản lý bộ đệm CSDL giữ một vài truy nhập CSDL mới nhất trong vùng đệm bộ nhớ. Vùng đệm được chia thành các trang có kích thước giống các trang trong CSDL.



Hình 2. Giao diện giữa bộ quản lý khôi phục và bộ quản lý vùng đệm

Volatile database: CSDL lưu trong vùng đệm. LRM thực hiện các thao tác của một giao tác trên violatile database sau đó ghi vào stable database.

Fetch: LRM đọc một dữ liệu. DBM chịu trách nhiệm kiểm tra, load trang này vào vùng đệm.

Bộ quản lý vùng đệm cung cấp một giao diện cái LRM dùng để thực hiện việc ghi trở lại chúng vào các trang đệm. Có thể hoàn thành bằng câu lệnh FLUSH, lệnh xác định các trang LRM muốn ghi lại.

b.Thông tin khôi phục:

Các thảo luận về các kỹ thuật khôi phục cáclỗi môi trường sau. Khi chúng ta giao tiếp với khôi phục CSDL tập trung, các lỗi truyền thông là không áp dụng được.

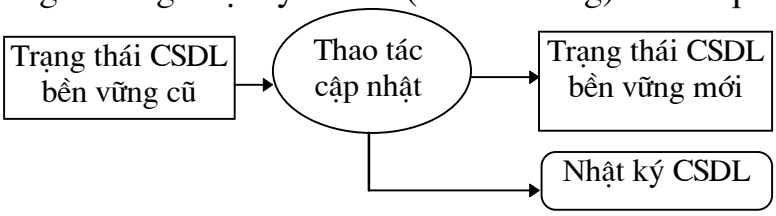
Thông tin khôi phục là thông tin về trạng thái của CSDL tai thời điểm xẩy ra lỗi (đưa CSDL về trạng thái xẩy ra lỗi vì khi xẩy ra lỗi dữ liệu trong bộ đệm bị mất). Thông tin khôi phục lỗi phụ thuộc vào phương pháp của việc thực hiện các cập nhật hai khả năng in-place updating và out-of place updating.

In-place updating: Thay đổi vật lý các giá trị trong stable database.

Out-of place updating: không thay đổi giá trị của các mục dữ liệu trong vùng stable database nhưng duy trì các giá trị mới một cách riêng rẽ.

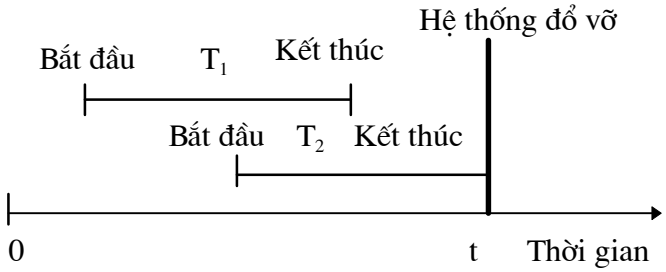
b1.Thông tin khôi phục in-place update:

Các thuật toán LRM và bộ quản lý vùng đệm là giống như các trang tại vùng đệm được ghi vào stable database chỉ khi bộ quản lý vùng đệm cần trang đệm mới. Nói cách khác lệnh FLUSH không được sử dụng bởi LRM và sự quyết định ghi vào stable database được làm tại việc suy xét của bộ quản lý vùng đệm. Trong trường hợp một lỗi xẩy ra khi kết quả T_1 chưa được ghi vào stable database việc khôi phục nhờ vào các thông tin lưu giữ trong nhật ký CSDL (database log) về kết quả của T_1.



Hình 2. Sự thực hiện thao tác cập nhật

Nội dung của nhật ký có thể không giống nhau theo sự thực hiện. Tuy nhiên thông tin tối thiểu cho mỗi một giao tác được cất giữ trong hầu hết các nhật ký CSDL: 1 begin-transaction record, các giá trị mục dữ liệu trước khi cập nhật gọi là ảnh trước (before image), Giá trị mục dữ liệu sau khi cập nhật gọi là ảnh sau (affter image), Termination record (about,commit). Có thể ghi nhật ký toàn bộ các trang hoặc một số lượng nhỏ các đơn vị.



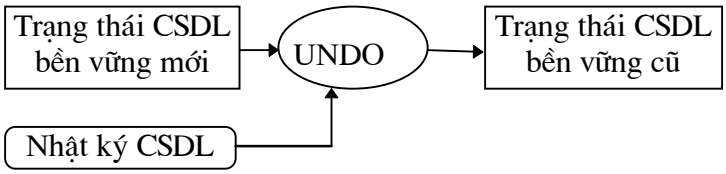
Hình 2. sự xuất hiện một lỗi hệ thống

Tương tự cho volatile database, nhật ký được duy trì trong các vùng đệm bộ nhớ chính (gọi là các vùng đệm nhật ký) và ghi lại vào stable storge (gọi là nhật ký bền vững) tương tự cho các trang đệm CSDL (hình 12.12). Các trang nhật ký có thể ghi vào kho lưu trữ bền vững (stable storage) bằng một trong hai cách sau:

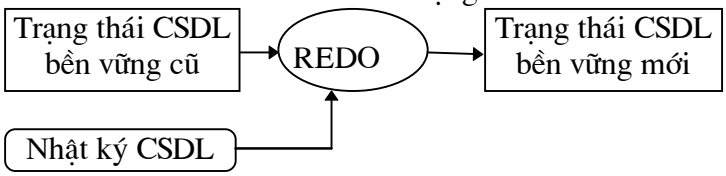
Đồng bộ: việc thêm mỗi một nhật ký đòi hỏi nhật ký được ghi vào nhật ký bền vững.

Không đồng bộ: ghi theo chu kỳ hoặc khi vùng đệm đầy.

Giao thức WAL(Write-Ahead-Logging): Nếu một cập nhật được ghi vào CSDL bền vững mà một lỗi xẩy ra khi chưa hoàn thành việc ghi vào nhật ký bền vững dẫn đến sai thông tin do đó luôn luôn ghi nhật ký bền vững trước ta thấy giao thức WAL đảm bảo cả undo và redo.



Hình 12.10 Hành động REDO



Hình 2. Hành động REDO

b2.Thông tin khôi phuc cập nhật out-of-place:

Như chúng ta đã kể trên, kỹ thuật cập nhật phổ biến là in-place updating. Do đó các kỹ thuật khác chỉ nói sơ lược.

Các kỹ thuật đặc biệt cho out-of place updating là soi bóng (shadowing) và các file khác nhau (differential files). Soi bóng sử dụng hai trang lưu trữ bền vững trong việc cập nhật, trang lưu trữ bền vững cũ gọi là trang soi bóng, được giữ nguyên vẹn và một trang mới với các giá trị mục dữ liệu được cập nhật được ghi vào CSDL bền vững. Các file khác nhau được chỉ ra trong chương 6 trong ngữ cảnh thực hiện toàn vẹn. Nhìn

chung phương pháp này duy trì mỗi file CSDL bền vững như một file chỉ đọc. Nó duy trì một file khác có thể ghi để ghi các thay đổi của file CSDL. F là một file CSDL logic, chỉ rõ phần chỉ đọc của nó FR và file khácтương ứng là DF. DF bao gồm hai phần: một phần thêm, lưu trữ các việc thêm vào F, chỉ ra DF^+, và một phần xoá tương ứng, chỉ ra DF^-. Toàn bộ các cập nhật được đối xử như việc xoá một giá trị cũ và thêm vào giá trị mới. F = (FR U DF^+) - DF^-.

c.Thực hiên các lệnh LRM:

Begin-transaction, read, write khá độc lập với thuật toán LRM và commit, abort, recover phụ thuộc vào thuật toán LRM.

Fix/no-fix (steal/no-steal): bộ quản lý vùng đệm có thể viết các trang đệm được cập nhật bởi một giao tác vào kho lưu bền vững trong khi thực hiện giao tác này hay phải đợi LRM báo tin.

Flush/no-flush (force/no-force): bộ quản lý vùng đệm flush trang đệm vào kho lưu trữ bền vững khi kết thúc giao tác hay theo thuật toán của bộ quản lý vùng đệm.

Các lệnh Begin-transaction, read, write:

Begin-transaction: Lệnh này là nguyên nhân nhiều thành phần của DBMS thoát khỏi chức năng giữ sổ (boockeeping function). Ghi vào nhật ký CSDL (có thể trễ lại đến việc thực hiện lệnh write đầu tiên để giảm truy nhập vào ra).

Read: fetch dữ liệu vào vùng đệm. Cao hơn việc đọc dữ liệu, LRM trả chúng cho bộ lập lịch.

Write: fetch dữ liệu, ghi ảnh trước, ảnh sau và record vào nhật ký. LRM báo tin cho bộ lập lịch khi thao táchoàn toàn thành công.

No-fix/No-flush: Thuật toán LRM kiểu này được gọi là một thuật toán undo/redo (steal/no-force).

Abort: Undo giao tác hoặc undo từng phần. LRM đọc các bản ghi nhật ký và trở lại ảnh trước, bộ lập lịch được báo hoàn thành công việc huỷ bỏ. Danh sách huỷ bỏ cất giữ các định danh của tất cả các giao tác được huỷ bỏ. Thao tác huỷ bỏ được quan tâm để hoàn thanh càng sớm càng tốt các định danh được cộng vào danh sách huỷ bỏ. Chú ý, khi chưa lưu dữ liệu vào CSDL bền vững thì bộ quản lý vùng đệm ghi trang CSDL chưa chính xác vào CSDL bền vững tại một thời điểm tương lai.

Commit: là nguyên dẫn đến một bản ghi kết thúc giao tác được viết vào nhật ký bởi LRM. Không có một hành động nào khác được thực hiện trong quá trình thực hiện lệnh commit ngoại trừ việc báo cho bộ lập lịch về việc hoàn thành hành động chuyển giao. Một cách chọn lựa để viết một kết thúc giao tác vào nhật ký là cộng thêm định danh của giao tác vào một danh sách chuyển giao.

Recover: LRM bắt đầu việc khôi phục bằng việc đến đầu nhật ký và redo cac thao tác của mỗi giao tác nếu tìm thấy cả hai bản ghi bắt đầu và kết thúc giao tác đó. Gọi là redo một phần. Tương tự nó undo các thao tác của các giao tác nếu tìm thấy bản ghi bắt đầu giao tác mà không tìm thấy bản ghi kết thúc giao tác tương ứng. Gọi là undo toàn cục. Nhưng ngược lại với undo giao tác đã bàn luận ở trên khác nhau là kết qủa của tất cả các giao tác không hoàn thành rollback, không trừ giao tác nào. Nếu danh sách chuyển giao và danh sách huỷ bỏ được sử dụng. Việc khôi phục bao gồm redo các thao tác của tất cả các giao tác trong danh sách chuyển giao và undo các thao tác của tất cả các giao tác trong danh sách huỷ bỏ. Lưu ý thứ tự trong trường hợp thứ hai.

No-fix/Flush:

Các thuật toán LRM sử dụng chiến lược này được gọi là undo/no-redo (steal/force).

Abort: Việc thực hiện huỷ bỏ là giống như trường hợp trước. Trên lỗi giao tác, LRM khởi tạo undo một phần cho phần này của giao tác.

Commit: LRM đưa ra một lệnh Flush tới bộquản lý vùng đệm, hiệu lực nó để ghi toàn bộ các trang CSDL cập nhật ở vùng đệm vào CSDL bền vững. Lệnh Commit là được thực hiện bởi việc ghi một bản ghi cào nhật ký hoặc bởi việc thêm một định danh vào danh sách chuyển giao khi toàn bộ công việc được hoàn thành, LRM báo cho bộ lập lịch rằng chuyển giao hoàn thành.

Recover: Từ toàn bộ các trang được cập nhật được ghi vào CSDL bền vững tại thời điểm chuyển giao, không cần thực hiện redo toàn bộ kết quả được ghi vào CSDL bền vững khi giao tác thành công do đó việc khôi phục bao gồm một undo toàn cục.

Fix/No-flush:

Trong trường hợp này LRM điều khiển việc ghi vào các VDB và CSDL bền vững. Cốt lõi ở đây không cho phép bộ quản lý vùng đệm ghi bất cứ trang VDB vào CSDL bền vững đến thời điểm tối thiểu một giao tác chuyển giao. Điều này được hoàn thành bởi lệnh FIX, lệnh này là một phiên bản sửa đổi của lệnh FETCH nhờ đó xác định các trang được fix trong CSDL đệm và không được ghi vào CSDL bền vững bởi bộ quản lý vùng đệm. Bất cứ lệnh fetch tới bộ quản lý vùng đệm để được thực hiện việc ghi được thay thế bởi một lệnh fix. Chú ý rằng điều này loại bỏ cần một thao tác undo toàn cục và do đó được gọi là thuật toán redo/no-undo (no-force/no-steal).

Abort: Từ các trang VBD không được ghi vào CSDL bền vững, không có hành động cần thiết đặc biệt. Để giải phóng các trang đệm các trang được fix bởi giao tác, tuy nhiên, nó là cần thiết cho

LRM gửi một lệnh unfix tới bộ quản lý vùng đệm cho toàn bộ các trang này. Nó là đủ để thoát ra khỏi hành động huỷ bỏ bởi việc viết một bản ghi huỷ bỏ vào nhật ký hoặc cộng thêm định danh giao tác vào danh sách huỷ bỏ, báo cho bộ lập lịch.

Commit: LRM gửi một lệnh unfix tới bộ quản lý vùng đệm cho từng trang VBD trang trước đó được fix bởi giao tác này. Chú ý rằng các trang này có thể được ghi ngay vào CSDL bền vững tại việc suy xét của bộ quản lý vùng đệm. Lệnh chuyển giao được thực hiện cùng bởi việc ghi một bản ghi vào nhật ký hoặc bởi việc thêm vào danh sách chuyển giao. Sau đó báo cho bộ lập lịch.

Recover: Như đã kể trên từ các trang VBD được cập nhật bởi các giao tác là vẫn không được ghi vào CSDL bền vững, không có sự cần thiết cho một undo toàn cục. LRM do đó khởi tạo một redo một phần khôi phục các giao tác đã chuyển giao nhưng các trang VBD có thể chưa được ghi vào CSDL bền vững.

Fix/Flush:

Đây là trường hợp mà LRM cho phép bộ quản lý vùng đệm ghi các trang VBD được cập nhật vào CSDL bền vững tại chính xác điểm chuyểngiao, không trước, không sau. Chiến lược này gọi là no-undo/no-redo (no-steal/force).

Abort: giống như trường hợp fix/no-flush.

Commit: LRM gửi một lệnh unfix tới bộ quản lý vùng đệm cho từng trang VDB trước đó đã được fix bởi giao tác này. Sau đó nó đưa ra một lệnh flush tới bộ quản lý vùng đệm, hiệu lực việc ghi toàn bộ các trang VDB đã unfix vao CSDL bền vững. Cuối cùng, lệnh chuyển giao được tiến hành bởi việc ghi vào nhật ký hoặc thêm vào danh sách chuyển giao. Điểm quan trọng cần chú ý ở đây là cả ba thao tác này được thực hiện như một giao tác nguyên tố. LRM báo cho bộ lập lịch.

Recover: Không cần làm gì trong trường hợp này. Điều này đúng từ CSDL bền vững phản ánh tất cả các kết quả của các giao tác thành công và không một kết quả nào của các giao tác không thành công.

d.Checkpointing:

Trong hầu hết các chiến lược thực hiện LRM, việc thực hiện hành vi khôi phục đòi hỏi phải tìm kiếm toàn bộ nhật ký. Có tổng chi phí đáng kể bởi vì LRM cố gắng tìm toàn bộ các giao tác cần undo và redo. Tổng chi phí có thể được giảm nếu nó có thể xây dựng một bức tường biểu hiện rằng CSDL tại điểm này là đã được cập nhật (up-to-date) và bền vững. Trong trường hợp này, redo bắt đầu từ điểm này đến cuối nhật ký và undo bắt đầu từ cuối nhật ký tới điểm này. Tiến trình xây dựng bức tường được gọi checkpointing. Checkpointing đạt được trong ba bước sau:

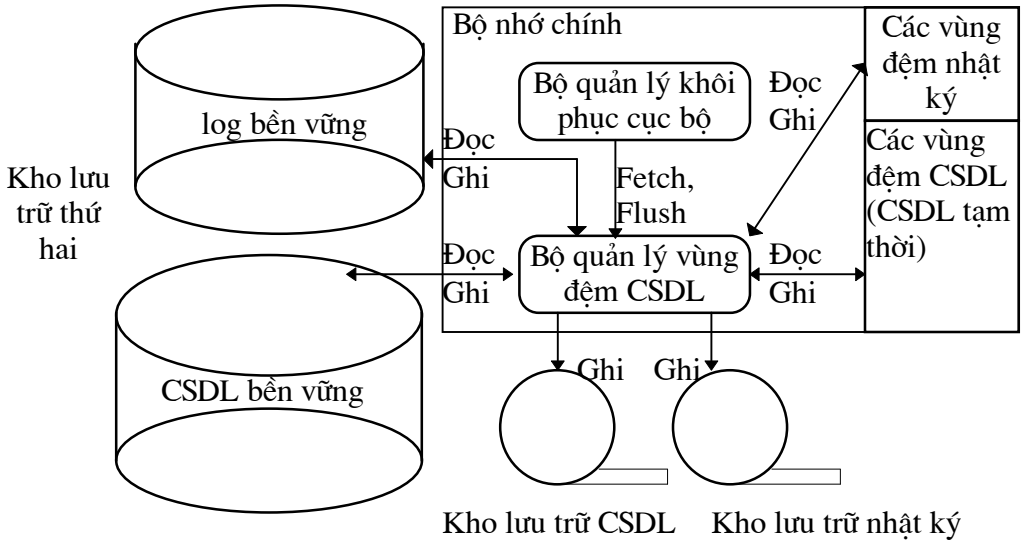
1.Ghi một bản ghi Begin-Checkpointing vào nhật ký.

2.Tập hợp điểm kiểm tra dữ liệu vào kho lưu trữ bền vững.

3.Ghi một bản ghi End-Checkpointing vào nhật ký.

Bước thứ nhất và bước thứ ba ép buộc tính nguyên tố của thao tác Checkpointing. Nếu một lỗi hệ thống xẩy ra trong quá trình Checkpointing, tiến trình khôi phục sẽ không tìm thấy bản ghi End-Checkpointing và sẽ lưu ý Checkpointing này là chưa hoàn thành. Có một số cách khác nhau cho dữ liệu được lựa chọn trong bước thứ hai, chúng được lựa chọn như thế nào và được lưu trữ ở đâu. Chúng ta cần nhắc một ví dụ sau đây, được gọi là điểm kiểm tra tính xác thực của giao tác (Transaction-Consistent checkpointing). Checkpointing bắt đầu bằng việc ghi một bản ghi Begin-Checkpointing vào nhật ký và dừng chấp nhận bất kỳ một giao tác mới bởi LRM. Đầu tiên hiệu lực các giao tác hoàn toàn hoàn thành, tất cả các trang VDB cập nhật được flush vào CSDL bền vững. Tiếp theo bằng việc thêm một bản ghi End-Checkpoimting vào nhật ký. Thuật toán Transaction-Consistent-Checkpointing không là một thuật toán hiệu suất cao, do một trễ đáng kể toàn bộ các giao tác. Có một số lựa chọn sơ đồ Checkpointing như là Action-Consistent checkpointing, Fuzzy Checkpointing ...

e.Quản lý lỗi môi trường:



Hình 2. Phân cấp bộ nhớ đầy đủ quản lý bởi LRM và BM

DBMS giao tiếp với ba mức bộ nhớ phân cấp: bộ nhớ chính, bộ nhớ truy nhập ngẫu nhiên đĩa từ, băng từ.

Hai phương pháp được đề xuất cho giao tiếp với điều này là thực hiện lưu trữ sự hoạt động trùng nhau với xử lý bình thường và lưu trữ CSDL gia tăng như thay đổi sự kiện.

5/Các giao thức tin cậy phân tán:

Như với các giao thức tin cậy cục bộ, các phiên bản phân tán trợ giúp đảm bảo tính nguyên tố và bền vững của các giao tác phân tán được thực hiện trên một số CSDL.

Begin-Transaction được thực hiện chính xácnhư trong trường hợp tập trungbởi bộ quản lý giao tác tại vị trí khởi đầu của giao tác. Lệnh read, write thực hiện theo luật ROWA thảo luận ở chương 11. Tại mỗi vị trí các lệnh được thực hiện theo kiểu miêu tả trong 12.4.3.

Tiến trình đồng điều phối: là tiến trình tại vị trí khởi đầu một giao tác thực hiện các thao tác của giao tác đó.

a.Các thành phần của các giao thức tin cậy phân tán:

Các kỹ thuật tin cậy trong CSDL phân tán bao gồm các giao thức chuyển giao, kết thúc, khôi phục. Trong CSDL tập trung không có giao thức kết thúc.

Giao thức kết thúc: Đảm bảo sự kết thúc các giao tác tại các phần khác nhau.

Giao thức khôi phục: Kiến tạo sự nhất quán giữa các sao bản khi kết nối lại các phần của mạng.

Yêu cầu chính của giao thức chuyển giao là đảm bảo tính nguyên tố của các giao tác phân tán

Giao thức kết thúc không khối (nonblocking) cho phép một giao tác kết thúc tại các vị trí hiệu lực (thao tác-operational) không với việc chờ đợi việc khôi phụccủa vị trí lỗi.

Giao thức khôi phục độc lập xác định sự kết thúc nào của một giao tác được thực hiện trong thời gian lỗi không với việc tra cứu bất kỳ một vị trí nào khác do đó giảm số thông báo trong thời gian khôi phục. Chú ý việc giao tác khôi phục độc lập có thể ám chỉ tồn tại giao tác kết thúc không khối.

b.Giao thức chuyển giao hai pha:

Điều kiện có khả năng khôi phục: các thuật toán SCC..

Đơn phương huỷ bỏ.

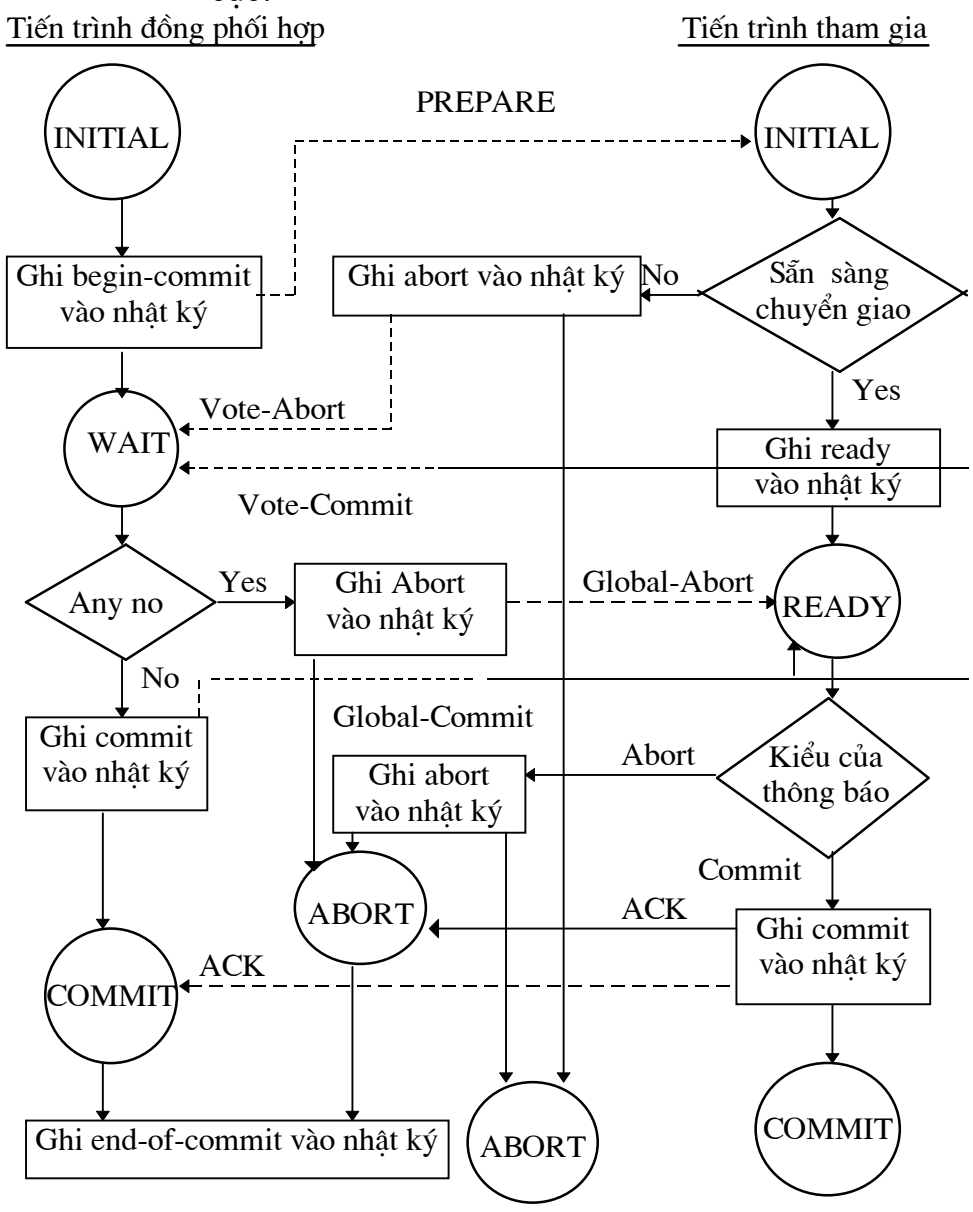
Một mô tả ngắn gọn giao thức chuyển giao hai pha:

Khởi tạo tiến trình đồng điều phối ghi một bản ghi Begin-commit vào nhật ký của nó, gửi một thông báo “prepare” tới toàn bộ các vị trí tham gia, và đi vào trạnh thái đợi. Khi một vị trí tham gia nhận thông báo “prepare” nó kiểm tra xem nó có thể chuyển giao giao tác. Nếu có thể nó ghi một bản ghi ready vào nhật ký, gửi một đề cử “vote-commit” tới tiến trình đồng điều phối, và đi vào trạng thái sẵn sàng. Ngược lại, ghi một bản ghi abort vào nhật ký và gửi đề cử “vote-abort” tới tiến trình đồng điều phối, và đi vào trạng thái sẵn sàng; Ngược lại nó ghi một bản ghi abort vào nhật ký và gửi đề cử “vote-abort” tới tiến trình đồng điều phối. Tiến trình

điều phối quyết định gửi thông báo "Global-commit" hay "Global-abort"; ghi bản ghi End-of-transaction vào nhật ký và đi vào trạng thái chuyển giao hay trạng thái huỷ bỏ. Các tiến trình tham gia chuyển giao hoặc huỷ bỏ giao tác theo các kiến trúc của tiến trình đồng điều phối và gửi trở lại tiến trình đồng điều phối một công nhận (acknowledgement). Luật chuyển giao toàn cục:

1.Một đề cử huỷ bỏ dẫn đến huỷ bỏ toàn cục.

2.Toàn bộ chuyển giao dẫn đến chuyển giao toàn cục.



Hình 2. Các hành động giao thức chuyển giao hai pha

Một vài điểm quan trọng về giao thức chuyển giao hai pha:

Chuyển giao hai pha cho phép huỷ bỏ đơn phương.

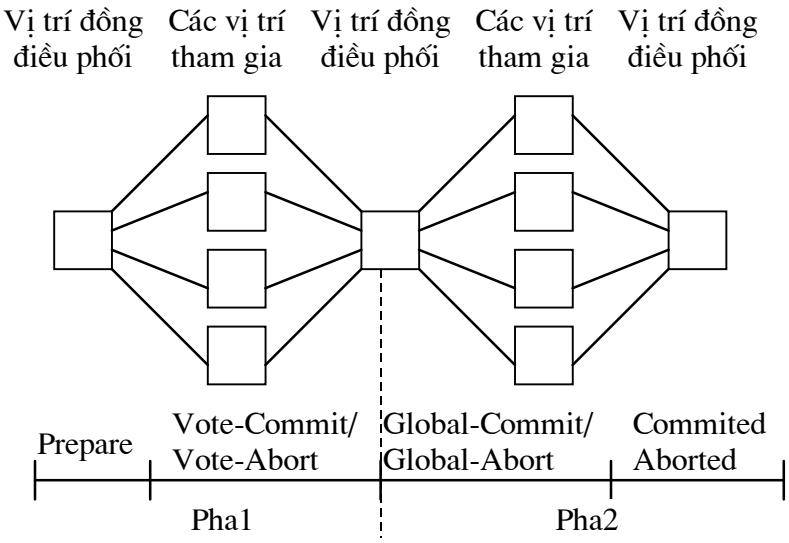
Một tiến trình tham dự đề cử chuyển giao một giao tác, nó không thể thay đổi đề cử của nó.

Trong khi một tiến trình tham gia ở trạng thái sẵn sàng, nó không thể huỷ bỏ hoặc chuyển giao tác.

Quyết định kết thúc toàn cục tạo bởi tiến trình đồng điều phối theo luật chuyển giao toàn cục.

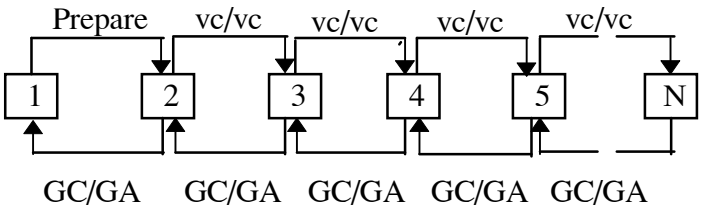
Timeout để thoát một trạng thái nào đó nếu không nhận được một thông báo từ vị trí khác.

Có một số mẫu kết nối khác nhau có thể được sử dụng trong giao thức chuyển giao hai pha. Giao thức chuyển giao hai pha tập trung chỉ có kết nối giữa tiến trình đồng điều phối với các tiến trình tham gia mà không có kết nối giữa các tiến trình tham gia với nhau.



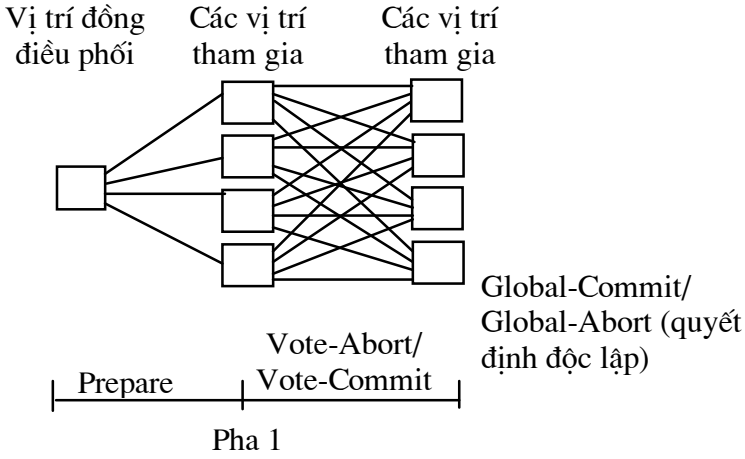
Hình 2. Cấu trúc liên kết của 2PC tập trung

Một lựa chọn khác là linear 2PC (còn gọi là nested 2PC): Các thành phần tham gia có thể kết nối với một thành phần khác. Có một thứ tự giữa các vị trí trong hệ thống cho ý định kết nối. Thứ tự giữa các vị trí tham gia trong việc thực hiện một giao tác là 1, 2, ..., N, tiến trình đồng điều phối là tiến trình đầu tiên trong thứ tự. Giao thức chuyển giao hai pha được thực hiện bởi kết phía trước từ tiến trình đồng điều phối (số 1) tới N.



Hình 2. Cấu trúc liên kết 2PC linear

Giao thức chuyển giao hai pha phân tán: Một cấu trúc kết nối thông thường khác cho thực hiện giao thức chuyển giao hai pha bao gồm kết nối giữa toàn bộ các thành phần tham gia trong pha thứ nhất của giao thức. Tiến trình đồng điều phối gửi thông báo "prepare" cho tất cả các thành phần tham gia gửi đề cử của mình cho tất cả các thành phần tham gia khác và từng thành phần tham gia tự quyết định chuyển giao hay huỷ bỏ theo luật chuyển giao toàn cục. Các thành phần tham gia phải biết định danh của các thành phần tham gia khác. Đáp ứng đòi hỏi này tiến trình đồng điều phối gửi kèm theo thông báo "prepare" một danh sách các thành phần tham gia.



Hinh 2. Cấu trúc liên kết 2PC phân tán

c.Các biến thể của chuyển giao hai pha:

Hai biến thể của chuyển giao hai pha được đề xuất để:

1. Giảm số thông báo giữa tiến trình đồng điều phối và các thành phần tham gia.
2. Giản số thời gian ghi các nhật ký.

c1.Giao thức chuyển giao hai pha phỏng đoán huỷ bỏ (presumed abort 2PCP):

Khi tiến trình đồng điều phối quyết định huỷ bỏ giao tác nó có thể ghi một bản ghi abort vào nhật ký và không chờ đợi các công nhận huỷ bỏ từ các thành phần tham gia. Tiến trình đồng điều khiển không cần phải ghi end-of-transaction sau một huỷ bỏ. Bản ghi abort không

được hiệu lực nếu một lỗi xẩy ra, chương trình khôi phục sẽ kiểm tra nhật ký của giao tảctước đó, và vì huỷ bỏ không có hiệu lực nên nó không tìm thấy các thông tin về giao tác đó. Trường hợp này nó hỏi tiến trình đồng điều phối và được trả lời về giao tác đó.

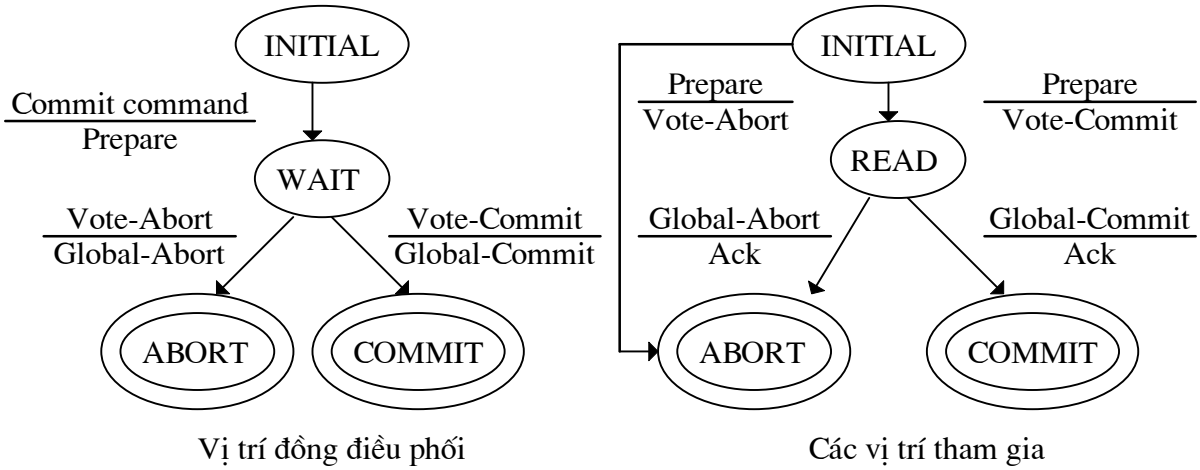
c2.Giao thức chuyển giao hai pha phỏng đoán chuyển giao:

6/Các lỗi vị trí:

a.Các giao thức kết thúc và khôi phục của chuyển giao hai pha:

a1.Các giao thức kết thúc:

Các phương pháp nắm giữ thời gian quá hạn (timeout) phụ thuộc vào thời gian các lỗi cũng như phụ thuộc vào các kiểu lỗi.



Hình 2. Trạng thái các giao tác trong giao thức 2PC

Coordinator timeout:

Timeout trong trạng thái đợi:

Timeout trong các trạng thái chuyển giao hay huỷ bỏ: Gửi lại và chờ trả lời (số lần gửi lại ?).

Participant timeout:

Timeout trong trạng thái khởi tạo: có thể đơn phương huỷ giao tác sau một thời gian có hai cách:

Kiểm tra nhật ký của nó tìm một bản ghi abort, và trả lời một "vote-abort".

Bỏ qua thông báo "prepare". Trong trường hợp này tiến trình đồng điều phối sau một thời gian timeout ở trạng thái WAIT đưa ra quyết định toàn cục.

Timeout trong trạng thái sẵn sàng:

Giao thức chuyển giao hai pha tập trung: bởi tiến trình đồng điều phối và đợi đến khi nhận được trả lời. Nếu tiến trình đồng điều phối lỗi thì nó sẽ duy trì tắc nghẽn (remain blocked).

Giao thức chuyển giao hai pha phân tán: P_i timeout, P_j đã trả lời có ba trường hợp sau:

P_j trong trạng thái khởi tạo: P_j chưa qyuết định và có thể chưa nhận thông báo "prepare". Nó có thểđơn phương huỷ bỏ và trả lời P_i với một thông báo "vote-abort".

P_j trong trạng thái sẵn sàng: Trong trường hợp này nó không thể giúp P_i kết thúc giao tác.

P_j trong trạng thái huỷ bỏ hay trong trạng thái chuyển giao: Gửi "vote-abort" hay "vote-commit".

P_i timeout có các trường hợp sau:

1.P_i nhận một "vote-abort": huỷ bỏ.

2.P_i nhận ra toàn bộ P_j trong trạng thái sẵn sàng: chưa có đủ thông tin để kết thúc chính xác do đó phải chờ (khi nào thì kết thúc hành động chờ ?).

3.Nhận được "global-abort" hoặc "global-commit" thì huỷ bỏ hoặc chuyển giao.

Nếu đang ở blocked có thể vượt qua trong một hoàn cảnh nhất định:

Nếu trong việc kết thúc toàn bộ các thành phần tham gia đã giải phóng mà tiến trình đồng điều phối lỗi thì có thể chọn một người điều phối khác để khởi động lại tiến trình chuyển giao. Có các cách khác nhau để chọn tiến trình đồng điều phối. Nếu có thể định nghĩa một thứ tự toàn bộ các vị trí thì chọn một trong số các tiến trình tiếp theo trong thứ tự, hoặc khởi tạo một thủ tục đề cử trong các thành phần tham gia để chọn ra tiến trình đồng điều phối mới.

Nếu cả tiến trình đồng điều phối và một số tiến trình tham gia cùng bị lỗi.

a2.Các giao thức khôi phuc:

Các trường hợp lỗi cơ bản:

Các lỗi tại vị trí của tiến trình đồng điều phối: Các trường hợp sau có thể xẩy ra:

1.Lỗi xẩy ra trong trạng thái khởi tạo: Trước khi tiến trình đồng điều phối khởi tạo xong thủ tục chuyển giao. Vì vậy nó sẽ bắt đầu tiến trình chuyển giao vào lúc khôi phục.

2.Lỗi xẩy ra trong trạng thái đợi: trong trường hợp này, tiến trình đồng điều phối đã gửi thông báo "prepare" vào lúc khôi

phục tiến trình đồng điều phối sẽ khởi động lại tiến trình chuyển giao cho giao tác này từ việc bắt đầu việc gửi thông báo "prepare" một hay nhiều lần.

3.Xẩy ra trong trạng thái chuyển giao hoặc huỷ bỏ: Không làm gì nếu đã nhận toàn bộ công nhận.

Các lỗi tại các vị trí tham gia:

1. Lỗi trong trạng thái khởi tạo: Đơn phương huỷ bỏ vì timeout tại tiến trình đồng điều phối.

2. Lỗi trong trạng thái: Trong trường hợp này tiến trình đồng điều phối đã gửi tới vị trí lỗi khẳng định quyết định về giao tác trước lúc lỗi. Vào lúc khôi phục, các tiến trình tham gia tại vị trí lỗi có thể coi lỗi này như một timeout trong trạng thái sẵn sàng là gửi giao tác chưa hoàn thành tới giao thúc kết thúc.

Các trường hợp mở rộng:

1.Lỗi tiến trình đồng xử lý sau khi bản ghi begin-commitđược ghi vào nhật ký nhưng trước khi thông báo "prepare" được gửi. Có thể làm lại như một lỗi trong trạng thái đợi và gửi "preoare" vào lúc khôi phục.

2.Một lỗi của tiến trình tham gia sau khi ghi một bản ghi ready vào nhật ký và trước khi gửi "vote-commit". Giống trường hợp 2 trong phần lỗi các vị trí tham gia.

3.Lỗi các vị trí tham gia sau khi ghi một bản ghi abort vào nhật ký và trước khi gửi đề cử "vote-abort". Không phải khôi phục gì, tiến trình đồng điều phối trong trạng thái chờ đợi sẽ gặp một timeout.

4.Lỗi tiến trình đồng điều phối sau khi ghi vào nhật ký các quyết định của nó (chuyển giao hoặc huỷ bỏ) và trước khi gửi thông báo "global-commit" hoặc "global-abort" tới các tiến trình tham gia. Như trường hợp 3 đối với tiến trình đồng điều phối, các tiến trình tham gia coi như một timeout trong trạng thái sẵn sàng.

5.Lỗi tiến trình tham gia sau khi ghi vào nhật ký bản ghi commit hoặc abort và trước khi gửi thông báo công nhận tới tiến trình đồng điều phối. Tiến trình tham gia có thể coi như trường hợp 3, tiến trình đồng điều phối sẽ gặp một timeout trong trạng thái chuyển giao hoặc trong trạng thái huỷ bỏ.

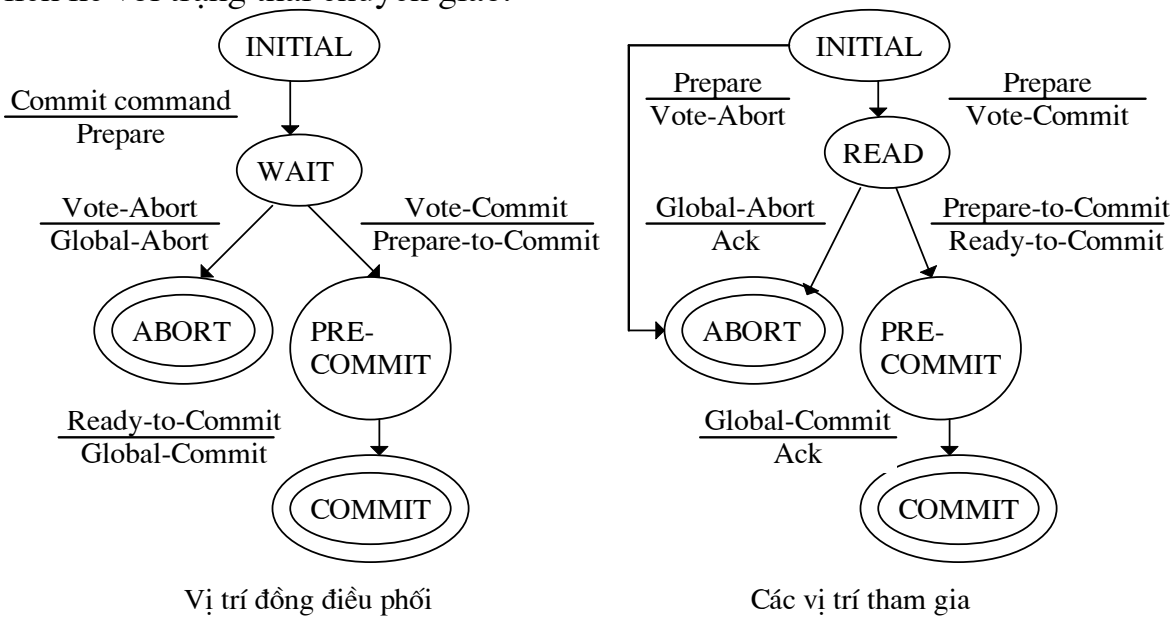
b.Giao thức chuyển giao ba pha:

Giao thức chuyển giao ba pha được thiết kế như một giao thức nonblocking. Chúng ta sẽ xem trong đoạn này cái thực sự là nonblocking khi các lỗi được hạn chế tại vị trí lỗi. Đầu tiên chúng ta xem xét điều kiện cần và đủ để thiết kế các giao thức nonblocking automic commitment (NbACP). Một giao thức chuyển giao được đồng bộ trong một

trạng thái chuyển đổi là nonblocking khi và chỉ khi sơ đồ chuyển đổi trạng thái của nó bao gồm điều kiện sau:

Không có trạng thái liền kề với cả hai trạng thái chuyển giao và huỷ bỏ.

Không có một t rạng thái không thể chuyển giao liền kề với trạng thái chuyển giao.



Vị trí đồng điều phối Các vị trí tham gia

Hình 2. Trạng thái các giao tác trong giao thức 3PC

Rõ ràng chuyển giao ba pha là một giao thức mà tất cả các trạng thái được đồng bộ trong một trạng thái chuyển đổi. Bởi vì các điều kiện cho chuyển giao nonblocking hai pha áp dụng cho chuyển giao ba pha.

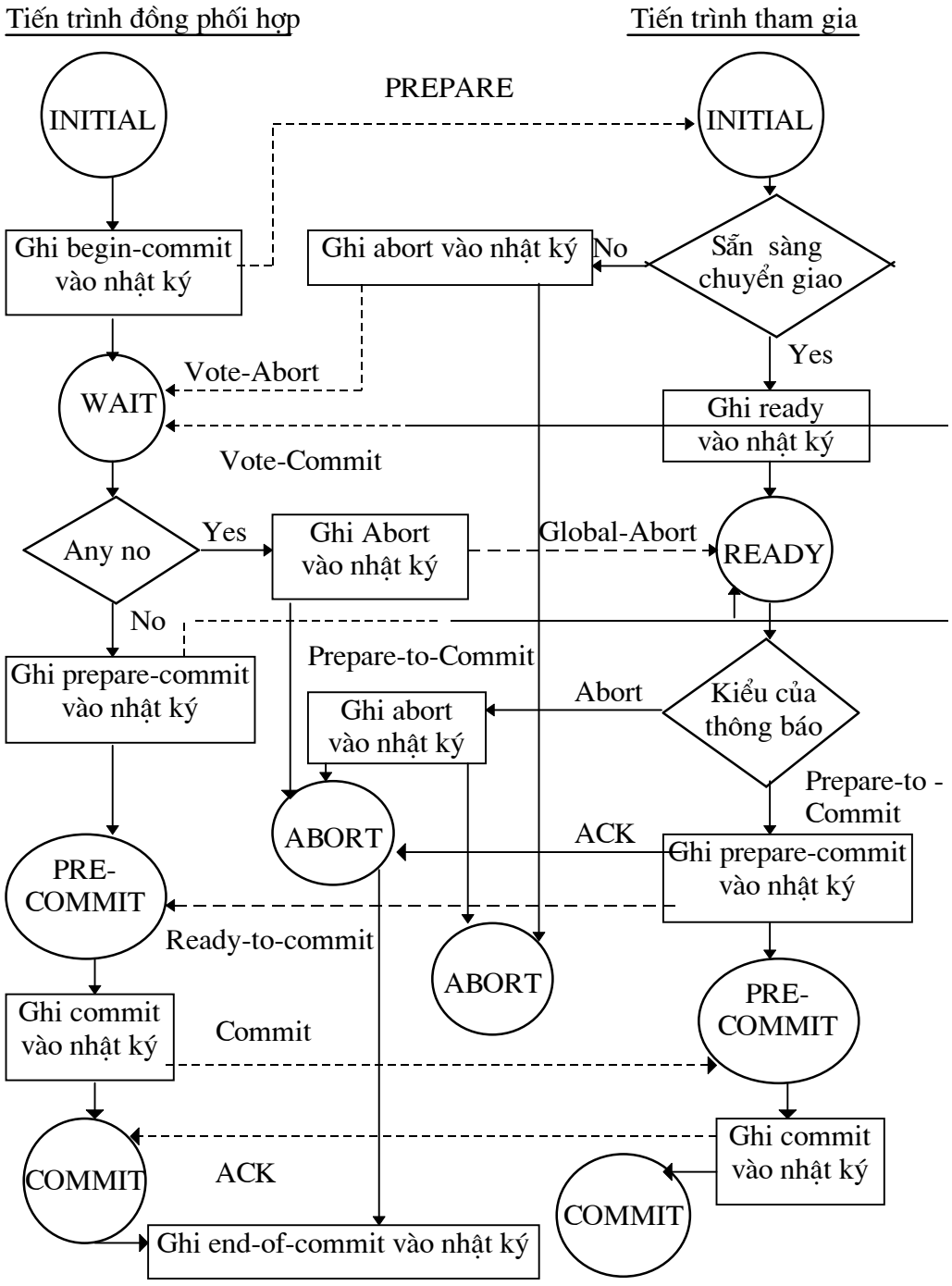
b1.Giao thức kết thúc:

Coordinator timeout:

Timeout trong trạng thái đợi: Đơn phương huỷ bỏ giao tác. Bởi vậy nó ghi một bản ghi abort vào nhật ký và gửi thông báo "global-abort".

Timeout trong trạng thái tiền chuyển giao (precommit): Ghi bản ghi commit vào nhật ký và gửi thông báo "global-commit" vì các tiến trình tham gia đã đề cử chuyển giao và ở trạng thái sẵn sàng (không biết đã ở trạng thái tiền chuyển giao chưa).

Timeout trong trạng thái chuyển giao hoặc trạng thái huỷ bỏ: Tiến trình đồng xử lý không biết các tiến trình tham gia thực hiện lệnh commit (hoặc abort). Tuy nhiên, chúng tối thiểu ở trạng thái tiền chuyển giao (hoặc trạng thái sẵn sàng) do đó giao tác là đồng bộ trong một trạng thái chuyển đổi và có thể sau đó giao thức kết thúc như mô tả trong trường hợp 2, hoặc trường hợp 3 phía dưới. Không phải làm gì đặc biệt.



Hình 2. Các hành động giao thức chuyển giao ba pha

Participant Timeout:

Timeout trong trạng thái khởi tạo: Giống như giao thức kết thúc chuyển giao hai pha.

Timeout trong trạng thái sắn sàng: Tiến trình tham gia đã đề cử chuyển giao nhưng không biết quyết định toàn cục của

tiến trình đồng điều phối do kết nối với tiến trình đồng điều phối lỗi, giao thức kết thúc tiến hành chọn tiến trình điều phối mới.

Timeout trong trạng thái tiền chuyển giao: ở trạng thái này tiến trình tham gia đã nhận thông báo "prepare-to-commit" và đang đợi thông báo cuối cùng "global-commit" từ tiến trình đồng điều phối. Trường hợp này giống như trường hợp 2 kể trên.

Chọn người điều phối mới:

Nếu một người điều phối mới trong trạng thái đợi, nó sẽ huỷ bỏ toàn cục giao tác. Các tiến trình tham gia có thể đang ở trạng thái khởi tạo, sẵn sàng, huỷ bỏ, hoặc tiền chuyển giao. Trong 3 trường hợp đầu thì không có vấn đề gì. Tuy nhiên các tiến trình tham gia ở trong trạng thái tiền chuyển giao đang đợi một thông báo "global-commit", nhưng chúng nhận được một "global-abort" thay thế. Sơ đồ chuyển đổi trạng thái của chúng không chỉ ra bất kỳ một sự chuyển đổi từ trạng thái tiền chuyển giao sang trạng thái huỷ bỏ. Giao tác này cần thiết cho giao thức kết thúc; nó có thể được cộng vào tập các chuyển đổi hợp lệ cái có thể xẩy ra trong khi thực hiện giao thức kết thúc.

Nếu một tiến trình đồng điều phối mới là ở trạng thái t iền chuyển giao, các tiến trình tham gia có thể ở trạng thái tiền chuyển giao, hoặc chuyển giao. Không tiến trình tham gia nào có thể ở trạng thái huỷ bỏ. Vì vậy nó quyết định global-commit và gửi đi một thông báo "global-commit".

Nếu tiến trình điều phối mới ở trạng thái huỷ bỏ thì quyết định toàn cục là huỷ bỏ.

Tiến trình đồng điều phối mới không lưu vết các lỗi của các tiến trình tham gia.Trong tiến trình này nó đon giản hướng dẫn các vị trí tiến tới kết thúc. Nếu một vài lỗi tiến trình tham gia xẩy ra trong lúc này chúng sẽ kết thúc nhờ khôi phục. Nếu lỗi tiến trình đồng điều phối mới thì chọn lại người điều phối mới.

Giao thức kết thúc này rõ ràng là nonblocking. Các vị trí hoạt động có thể kết thúc chính xác toàn bộ những sự kiện xẩy ra trong giao tác và tiếp tục các thao tác của nó. Tính đúng đắn của thuật toán được chứng minh trong [Skeen, 1982b].

b2.Các giao thức khôi phuc:

Có một vài sự khác nhau nhỏ giữa chuyển giao hai pha và chuyển giao ba pha. Chúng ta chỉ chỉ ra các sự khác nhau này.

Coordinator lỗi trong trạng thái đợi: Bàn luận trong giao thức kết thúc. Vì vậyvào lúc khôi phục tiến trình đồng điều phối hỏi về số phận kết thúc của giao tác.

Coordinator lỗi trong trạng thái tiền chuyển giao: Giao thức kết thúc sẽ hướng dẫncác vị trí hoạt động tiến tới kết thúc giao tác.

Do đó nó có thể chuyển từ trạng thái tiền chuyển giao tới trạng thái huỷ bỏ. Trong tiến trình này tiến trình đồng điều phối hỏi về số phận kết thúc của giao tác.

Một tiến trình tham gia lỗi trong trạng thái tiền chuyển giao: Hỏi về kết thúc như thế nào của các tiến trình tham gia khác đã kết thúc giao tác.

Chương 3: Cơ sở dữ liệu phân tán trên Oraclce7:

I/Cập nhật phân tán (2PCP, Điều khiển tương tranh, Khôi phục):

Môi trường phân tán:

Cập nhật phân tán thực tế là có nhiều người sử dụng đang chia sẻ và truy nhập vào dữ liệu tồn tại trên nhiều vị trí, do đó nẩy sinh các điều cần quan tâm sau:

- Sự sao bản (sự lan truyền cập nhật).
- Điều khiển tương tranh.
- Việc quản lý khôi phục giao tác.

Trong một hệ thống phân tán, một đối tượng dữ liệu có thể được mô tả tại nhiều vị trí. Chắc chắn rằng các cập nhật bất kỳ mô tả tại bất kỳ vị trí được lan truyền tới toàn bộ các vị trí khác là trách nhiệm của các cơ chế sao bản của Oracle.

Máy chủ Oracle cung cấp một vài phương pháp cho sự sao bản dữ liệu.

- Snapshot chỉ đọc.
- Symmetric replication facility.

-Snapshot cập nhật được.

-N-way master replication.

Một snapshot là một bản sao đầy đủ của một bảng, hoặc một tập bảng ánh xạ trạng thái gần nhất của một bảng chủ (một bảng trên nút được chỉ ra như nút chủ). Snapshot cập nhật được có thể ánh xạ cập nhật địa phương và do đó cải thiện thời gian trả lời bởi tránh tắc nghẽn mạng. Tuy nhiên phải có một cơ chế đảm bảo cập nhật địa phương không bị mất khi snapshot được làm tươi từ bảng chủ. Symmetric replication facility của Oracle cung cấp cơ chế này. Nó cho pháp nhiều bản sao của dữ liệu được bảo quản tại các vị trí khác nhau trong hệ phân tán.

<u>Điều khiển tương tranh:</u>

Trong một hệ phân tán, có khả năng rất lớn là có nhiều hơn một người sử dụng thực hiện cùng một lúc các giao tác cập nhật vào cùng một dữ liệu. Oracle cung cấp cơ chế khoá để quản lý truy nhập nhiều người vào cùng một dữ liệu. Cơ chế khoá có thể dẫn đến một khoá chết. Oracle

phát hiện khoá chết địa phương bằng các đồ thị "đợi" (LWFG). Và khoá chết toàn cục được phát hiện bằng một thời gian quá hạn (time-out).

Quản lý giao tác phân tán:

Oracle sử dụng giao tác chuyển giao hai pha. Gồm pha chuẩn bị và pha chuyển giao.

Sử dụng cơ chế quản lý giao tác:

Oracle7 điều khiển và giám sát một cách tự động chuyển giao hoặc rollback của một giao tác cập nhật phân tán và đảm bảo tính toàn vẹn dữ liệu của CSDL toàn cục. Cơ chế này hoàn toàn trong suốt đối với người dùng và các ứng dụng. Cơ chế chuyển giao hai pha sẽ đảm bảo toàn bộ vị trí tham gia trong một giao tác phân tán chuyển giao hoặc rollback, điều này đảm bảo tính toàn vẹn của CSDL toàn cục. Cơ chế quản lý giao tác được sử dụng chỉ khi một thay đổi cập nhật bao gồm hai hoặc nhiều CSDL trong hệ phân tán, hoặc có lời gọi thủ tục xa (tham chiếu đến một đối tượng ở xa sử dụng tên đối tượng toàn cục của nó). Khi một vị trí là chỉ đọc, Oracle tự động ghi điều đó, và vị trí này không cần tham gia vào các pha chuẩn bị và chuyển giao. Toàn bộ các thay đổi ngầm được thực hiện thông qua các ràng buộc toàn vẹn, các lời gọi thủ tục xa, và các trigger cũng được bảo hộ bởi cơ chế quản lý giao tác phân tán của Oracle7.

Pha chuẩn bị và pha chuyển giao:

Pha chuẩn bị: vị trí điều phối hỏi các vị trí tham gia đề cử của các vị trí tham gia (chuyển giao hay loại bỏ giao tác). Các vị trí tham gia có thể trả lời một trong ba đề cử sau:

- Prepared: Dữ liệu vừa được sửa đổi bởi một câu lệnh trong giao tác phân tán, và vị trí đã hoàn thành chuẩn bị.
- Read-only: Không có dữ liệu nào tại vị trí đã hoặc có thể được sửa đổi, do đó không có chuẩn bị nào là cần thiết.
- Abort: Vị trí không hoàn thành chuẩn bị.

Để hoàn thành pha chuẩn bị mỗi vị trí phải thực hiện các hành động sau:

- Hỏi các vị trí con của nó để chuẩn bị.
- Kiểm tra xem giao tác tay đổi dữ liệu tại vị trí đó hoặc bất kỳ vị trí con nào của nó không. Nếu không có cập nhật, vị trí này bỏ qua các bước tiếp theo và gửi một thông báo chỉ đọc.
- Vị trí chỉ ra (phân phối) toàn bộ dữ liệu nó cần để chuyển giao giao tác nếu dữ liệu được cập nhật.
- Vị trí đẩy (flushes) bất kỳ sự ghi nào tương ứng các thay đổi làm bởi giao tác đó vào nhật ký redo địa phương.

- Vị trí đảm bảo rằng các khoá nắm giữ cho giao tác đó là có thể tiếp tục tồn tại sau một lỗi.
- Vị trí trả lời vị trí đã tham khảo nó trong giao tác phân tán với một đề cử Prepared hoặc đề cử Abort.

Khi một nút không thể hoàn thành chuẩn bị nó thực hiện các hành động sau:

- Vị trí đó giải phóng các khoá đang được nắm giữ bởi giao tác và rollback phần địa phương của giao tác.
- Vị trí trả lời vị trí tham khảo nó trong giao tác phân tán một đề cử Abort.

Pha chuyển giao:

Trước khi pha này xẩy ra, toàn bộ các vị trí tham khảo trong giao tác đảm bảo rằng chúng đã có các tài nguyên cần thiết để chuyển giao tác. Và tất cả đã hoàn thành pha chuẩn bị. Pha chuyển giao bao gồm các bước sau:

- Vị trí điều phối toàn cục gửi các thông báo yêu cầu chuyển giao tới tất cả các vị trí.
- Tại từng nút, Oracle7 chuyển giao phần cục bộ của giao tác phân tán: giải phống các khoá, ghi các sự ghi vào nhật ký redo địa phương, chỉ ra rằng giao tác đã hoàn thành.

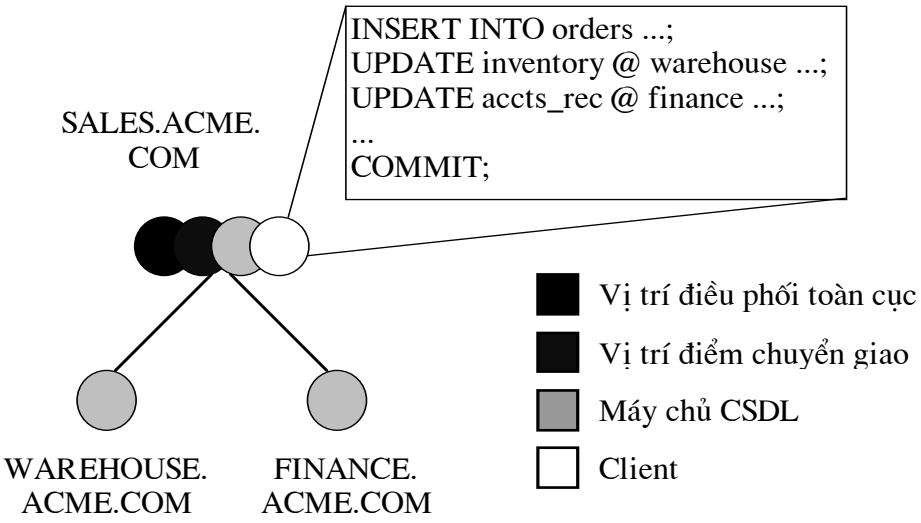
Cây phiên (Sesion tree):

Oracle7 định nghĩa một cây phiên của toàn bộ các vị trí tham gia trong giao tác. Một cây phiên là một mô hình phân cấp mô tả các mối quan hệ giữa các phiên và các vai trò của chúng. Tất cả các vị trí tham gia trong một cây phiên của một giao tác phân tán mang một hoặc nhiều vai trò:

- Client.
- Máy chủ CSDL.
- Điều phối toàn cục.
- Commit point site.

Vai trò một trong giao tác phân tán xác định bởi:

- Có không giao tác là địa phương hoặc ở xa.
- Commit point strength của nút đó.
- Có không toàn bộ dữ liệu câu hỏi là sẵn sàng tại một nút, hoặc có không các nút khác cần được tham khảo để hoàn thành giao tác.
- Có không nút chỉ đọc.



Hình Một ví dụ về một cây phiên

Máy chủ và máy chủ CSDL:

Một máy chủ là một nút đã chỉ dẫn trực tiếp một giao tác phân tán hoặc được hỏi tới nút tham gia trong một giao tác vì nút khác yêu cầu dữ liệu từ CSDL được gọi là máy chủ CSDL.

Các điều phối cục bộ:

Một nút phải chỉ dẫn trên các nút khác để hoàn thành phần của nó trong giao tác phân tán được gọi là nút điều phối. Một nút điều phối cục bộ có trách nhiệm cho việc điều phối giao tác giữa các nút nó liên kết trực tiếp bởi:

- Nhận và truyền thông tin trạng thái giao tác tới và từ các nút này.
- Chuyển các câu hỏi tới các nút này.
- Nhận các câu hỏi từ các nút và chuyển chúng đến các nút khác.
- Đưa lại kết quả của các câu hỏi cho các nút khởi tạo ra chúng.

Điều phối toàn cục:

Nút mà tại đó giao tác bắt đầu (nút ứng dụng CSDL đưu ra giao tác phân tán đã trực tiếp kết nối) được gọi là điều phối toàn cục. Nút này sẽ trở thành gốc của cây phiên. Điều phối toàn cục thực hiện các thao tác sau trong quá trình một gioa tác phân tán:

- Toàn bộ các lệnh của một giao tác phân tán, Các lời gọi thủ tục xa, ... được gửi bởi vị trí điều phối toàn cục tới trực tiếp các nút chỉ dẫn. Theo các đố hình thành cây phiên.
- Vị trí điều phối toàn cục chỉ dẫn toàn bộ các nút chỉ dẫn khác vị trí điểm chuyển giao (commit point site) chuẩn bị giao tác.

- Nếu toàn bộ các nút hoàn thành chuẩn bị, vị trí điều phối toàn cục chỉ dẫn vị trí điểm chuyển giao khởi tạo chuyển giao toàn cục giao tác.
- Nếu có một thông báo Abort, vị trí điều phối toàn cục chỉ dẫn tất cả các nút khởi tạo rollback giap tác.

Vi trí điểm chuyển giao:

Công việc của vị trí điểm chuyển giao là khởi tạo chuyển giao hoặc rollback như chỉ dẫn của vị trí điều phối toàn cục. Người quản trị hệ thống luôn luôn chỉ ra một nút là vị trí điểm chuyển giao trong cây phiên bằng việc phân chia tất cả các nút một commit point strength. Nút được chọn như vị trí điển chuyển giao có thể là nút lưu trữ dữ liệu tới hạn (dữ liệu được sử dụng rộng khắp). Vị trí điểm chuyển giao là riêng biệt với với tất cả các nút khác liên quan đến một giao tác phân tán:

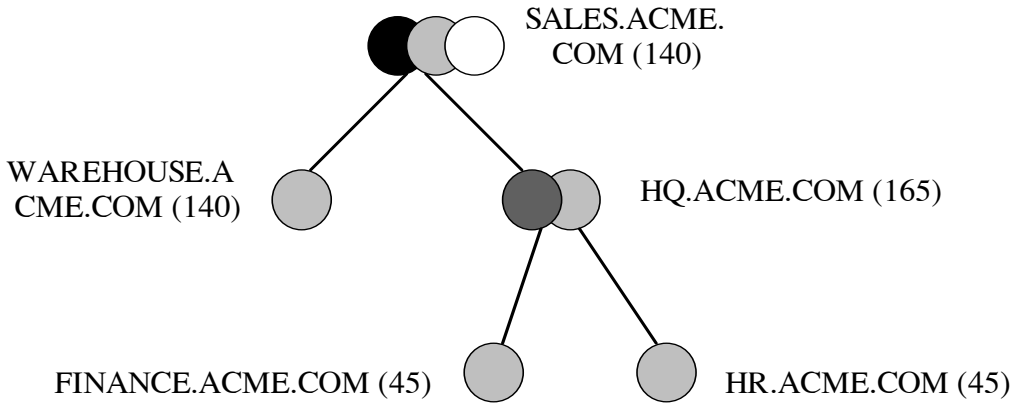
- Vị trí điểm chuyển giao không bao giờ vào trạng thái chuẩn bị. Điều này ẩn chứa một thuận lợi vì nó lưu trữ hầu hết các dữ liệu tới hạn, dữ liệu không bao giờ ở lại trong tình trạng nghi vấn, mạc dù một lỗi xuất hiện (Trong tình huống lỗi, các nút lỗi còn trong trạng thái chuẩn bị nắm giữ các khoá cần thiết trên dữ liệu đến khi sự nghi ngờ được giải quyết.)
- Hậu quả của giao tác phân tán tại vị trí điểm chuyển giao xác định có hay không giao tác tại tất cả các nút đã chuyển giao hoặc rollback. Vị trí điều phối toàn cục chắc chắn rằng toàn bộ các nút hoàn thành giao tác cách tương tự đó vị trí điểm chuyển giao làm.

Một giao tác phân tán được cân nhắc để được chuyển giao một lần tất cả các nút đã được chuẩn bị và giao tác đã chuyển giao tại vị trí điểm chuyển giao (mặc dù một vài nút tham gia có thể vẫn trong trạng thái chuẩn bị và giao tác vẫn chưa thực sự chuyển giao). Nhật ký redo của vị trí điểm chuyển giao được cập nhật sớm như có thể giao tác phân tán được chuyển giao tại nút này. Ngược lại, một giao tác phân tán được cân nhắc không chuyển giao nếu nó không được chuyển giao tại vị trí điểm chuyển giao.

Commit point Strength:

Mỗi nút hành động như một máy chủ CSDL phải được phân chia một Commit point Strength. Nếu một máy chủ CSDL được chỉ dẫn trong một giao tác phân tán, giá trị Commit point Strength của nó xác định vai trò nó thực hiện trong các pha chuẩn bị và chuyển giao. Đặc biệt Commit point Strength xác định có không một nút là vị trí điểm chuyển giao trong giao tác phân tán. Giá trị này được xác định sử dụng tham số khởi tạo COMMIT_POINT_STRENGTH. Vị trí điểm chuyển giao được xác định tại lúc bắt đầu pha chuẩn bị.Vị trí điểm chuyển giao chỉ được chọm từ các vị trí tham gia giao tác. Một khi nó đã được xác định, vị trí điều phối toàn cục sẽ

gửi thông báo tới tất cả các vị trí tham gia. Trong các nút được chỉ dẫn trực tiếp bởi vị trí điều phối toàn cục, nút với Commit point Strength cao nhất được chọn. Sau đó, khởi tạo nút được chọn xác định nếu bất kỳ một máy chủ của nó (các nút khác mà nó lấy thông tin từ đó cho giao tác này) có một Commit point Strength cao hơn. Mỗi nút với Commit point Strength cao nhất được chỉ dẫn trực tiếp trong giao tác, hoặc một trong các máy chủ của nó có ất cao hơn trở thành vị trí điểm chuyển giao.



Hình Các Commit point Strength và xác định vị trí điểm chuyển giao

Các điều kiện sau áp dụng khi xác định vị trí điểm chuyển giao:

- Một nút chỉ đọc không thể được chỉ định như một vị trí điểm chuyển giao.
- Nếu nhiều nút được chỉ dẫn trực tiếp bởi vị trí điều phối toàn cục có cùng Commit point Strength, Oracle7 sẽ chỉ định một trong các nút này như vị trí điểm chuyển giao.
- Nếu giao tác phân tán kết thúc với một rollback, các pha chuẩn bị chuyển giao là không cần, do đó một vị trí điểm chuyển giao không bao giờ được chỉ ra. Thay thế, vị trí điều phối toàn cục gửi một lệnh ROLLBACK tới toàn bộ các nút và kết thúc tiến trình của giao tác đó.

Commit point Strength chỉ xác định vị trí điểm chuyển giao trong một giao tác phân tán. Vì vị trí điểm chuyển giao lưu trữ thông tin về trạng thái của giao tác, vị trí điểm chuyển giao không thể là một nút thường xuyên không tin cậy hoặc không sẵn sàng trong trường hợp cần thông tin về trạng thái của giao tác.

Các số giao dịch hệ thống:

Mỗi giao tác chuyển giao có một số giao dịch hệ thống được kết nối (SCN) để định danh duy nhất các thay đổi được làm bởi các câu lệnh SQL trong giao tác. Trong một hệ phân tán, các số SCN của kết nối các nút được điều phối khi:

- Một kết nối xẩy ra sử dụng đường dẫn được miêu tả bởi một hoặc nhiều database link.
- Một câu lệnh SQL thực hiện.
- Một giao tác chuyển giao.

Việc điều phối các SCN giữa các nút của hệ phân tán cho phép đọc xác thực phân tán toàn cục ở cả hai mức câu lệnh và mức giao tác. Trong pha chuẩn bị, Oracle7 xác định SCN cao nhất tại toàn bộ các nút liên quan trong giao tác. Giao tác sau đó chuyển giao với SCN cao nhất tại vị trí điểm chuyển giao. SCN chuyển giao là sau khi gửi tới tất cả các nút đã chuẩn bị với quyết định chuyển giao.

Các giao tác phân tán:

Có ba trường hợp trong toàn bộ hoặc một phần giao tác phân tán là chỉ đọc:

- Một giao tác phân tán có thể là chỉ đọc một phần nếu:

-Chỉ các câu hỏi được dưa ra tại một hoặc nhiều nút.

-Các cập nhật không sửa đổi bất kỳ một bản ghi nào.

-Các cập nhật rollback do các sự vi phạm các ràng buộc toàn vẹn hoặc các trigger being fired.

Trong từng này trường hợp, các nút chỉ đọc công nhận sự kiện này khi chúng được hỏi thực hiện pha chuẩn bị. Chúng trả lời từng vị trí điều phối cục bộ với đề cử read-only. Bởi việc làm này, pha chuyển giao nhanh hơn vì Oracle loại trừ các nút chỉ đọc từ tiến trình sau.

- Giao tác phân tán có thể là hoàn toàn chỉ đọc (không dữ liệu nào được thay đổi tại bất kỳ nút nào) và giao tác không được bắt đầu với câu lệnh SET TRANSACTION READ ONLY. Trong trường hợp này, tất cả các nút công nhận chúng chỉ đọc trong pha chuẩn bị, và pha chuyển giao không được yêu cầu. Tuy nhiên, vị trí điều phối toàn cục, không biết có hay không toàn bộ các nút chỉ đọc, vẫn phải thực hiện các thao tác liân quan trong pha chuyển giao.
- Giao tác phân tán có thể hoàn toàn chỉ đọc (toàn bộ các câu hỏi tại tất cả vị trí) và giao tác không được bắt đầu với câu lệnh SET TRANSACTION READ ONLY. Trong trường hợp này, chỉ các câu hỏi được phép trong giao tác, và vị trí điều phối đảm nhận pha chuẩn bị và pha chuyển giao. Các cập nhật bởi các giao tác khác không làm suy giảm tính đọc xác thực (read consistency) ở mức giao tác toàn cục, vì nó được đảm bảo tính nguyên tố bởi điều phối của các SCN tại từng nút của hệ phân tán.

Giới hạn số giao tác trên từng nút:

Tham số khởi tạo DISTRIBUTED_TRANSACTION điều khiển số các giao tác có thể trong một instance qui định kiêm vị trí tham gia, cả như một client và một server. Nếu giới hạn này được đạt tới và một người sử dụng

sau đó cố gắng đưa ra một lệnh SQL tham khảo một CSDL ở xa, lệnh này được rollback và một thông báo lỗi được trả lại:

ORA-2024: too many global transactions.

Các vấn đề cập nhật phân tán:

Một mạng hoặc một hệ thống lỗi có thể dẫn đến các vấn đề sau:

- Chuẩn bị/Chuyển giao đang thực hiện một lỗi xẩy ra có thể chưa được hoàn thành tại toàn bộ các nút của cây phiên.
- Nếu một lỗi vẫn còn (ví dụ nếu mạng hỏng trong thời gian dài), dữ liệu dành riêng được khoá bởi các giao tác trong nghi ngờ (in-doubt transaction) là không sẵn sàng cho các câu lệnh của các giao tác khác.

Các lỗi cấm truy nhập dữ liệu:

Khi một người sử dụng đưa ra một câu lệnh SQL, Oracle7 cố gắng khoá các tài nguyên được yêu cầu để thực hiện thành công câu lệnh này. Tuy nhiên, nếu các dữ liệu được yêu cầu đang được nắm giữ bởi các câu lệnh của các giao tác chưa chuyển giao khác và tiếp tục được khoá trong tổng số thời gian quá mức, một quá hạn thời gian xẩy ra. Xem xét hai hoàn cảnh sau:

Giao tác quá hạn thời gian:

Một câu lệnh DML SQL yêu cầu khoá trên CSDL ở xa có thể bị tắc nghẽn từ việc làm như vậy nếu một giao tác (phân tán hoặc không phân tán) đang làm chủ các khoá trên dữ liệu được yêu cầu. Nếu các này tiếp tục làm tắc nghẽn yêu cầu của câu lệnh SQL, một thời gian quá hạn xẩy ra, câu lệnh được rollback, và một thông báo lỗi được trả lại người sư dụng. Khoảng thời gian quá hạn được điều khiển bởi tham số khởi tạo DISTRIBUTED_LOCK_TIMEOUT (đơn vị tính là giây).

In-doubt:

Một truy hỏi hoặc một câu lệnh SQL yêu cầu các khoá trên một CSDL cục bộ có thể bị tắc nghẽn từ việc làm không rõ hạn định tới các tài nguyên bị khoá của một giao tác phân tán còn nghi ngờ. Trong trường hợp này, câu lệnh SQL rollback ngay lập tức.

Overriding In-doubt Transaction:

Một người quản trị CSDL có thể hiệu lực thủ công chuyển giao hoặc rollback của một giao tác phân tán In-doubt. Tuy nhiên, một giao tác In-doubt xác định là thủ công có hiệu lực cao hơn chỉ khi các tình huống sau tồn tại:

- Giao tác in-doubt khoá dữ liệu được yêu cầu bởi các giao tác khác.
- Một giao tác in-doubt ngăn chặn khoảng rộng một đoạn rollback được sử dụng bởi các giao tác khác. Phần đầu của một ID giao tác địa phương của giao tác phân tán In-doubt giao tiếp với ID của đoạn

rollback, như được liệt kê bởi các view từ điển dữ liệu DBA_2PC_PENDING và DBA_ROLLBACK_SEGS.

- Lỗi đã không cho phép các pha chuẩn bị và chuyển giao hoàn thành sẽ không được chính xác trong một giai đoạn được chấp nhận.

Bình thường, một quyết định hiệu lực địa phương một giao tác phân tán In-doubt có thể được làm trong việc tham khảo với các người quản trị tại các các địa phương khác. Một quyết định sai có thể dẫn đến CSDL không thuần nhất điều rất khó tìm vết và bạn phải thao tác chính xác.

Nếu điều kiện trên không được áp dụng, luôn luôn cho phép các khôi phục một cách tự động để hoàn thành giao tác. Tuy nhiên, nếu bất kỳ một chuẩn trên được gặp, người quản trị có thể xem xét một địa phương có quyền cao hơn của một giao tác In-doubt. Nếu một quyết định được làm để hiệu lực địa phương giao tác để hoàn thành, người quản trị CSDL phân tích thông tin sẵn sàng với các đích sau:

- Cố gắng tìm một nút đã chuyển giao hoặc đã rollback giao tác. Nếu có thể tìm thấy một nút đã sẵn sàng giải quyết giao tác, sau đó có thể hành động tại nút này.
- Xem bất kỳ thông tin được giữ trong cột TRAN_COMMENT của DBA_2PC_PENDING cho giao tác phân tán.
- Xem thông tin bất kỳ thông tin được giữ trong cột ADVICE của DBA_2PC_PENDING cho giao tác phân tán. một ứng dụng có thể chỉ ra lời khuyên có hay không hiệu lực chuyển giao hoặc rollbackcủa các phần riêng rẽ của một gia tác phân tán với tham số ADVICE của câu lệnh SQL ALTER SESSION.

Override:

Các đặc điểm khôi phục giao tác phân tán:

Nếu muốn, có thể hiệu lực lỗi của một giao tác phân tán để quan sát RECO, giải quyết một cách tự động phần địa phương của một giao tác. sử dụng tham số COMMENT của câu lệnh COMMIT với cú pháp:

COMMIT COMMENT ‘ORA-2PC-CRASH-TEST-n’; với n là một trong các số sau:

n	Tác dụng
1	Đổ vỡ vị trí điểm chuyển giao sau tập hợp.
2	Đổ vỡ vị trí không phải vị trí điểm chuyển giao sau tập hợp.
3	Đổ vỡ trước prepare (không phải vị trí điểm chuyển giao).
4	Đổ vỡ sau prepare (không phải vị trí điểm chuyển giao).
5	Đổ vị trí điểm chuyển giao trước commit.
6	Đổ vị trí điểm chuyển giao sau commit.
7	Đổ vỡ vị trí không phải vị trí điểm chuyển giao trước

	commit.
8	Đổ vỡ vị trí không phải vị trí điểm chuyển giao sau commit.
9	Đổ vỡ vị trí điểm chuyển giao trước forget.
10	Đổ vỡ vị trí không phải vị trí điểm chuyển giao trước forget.

II/Môi trường sao bản trên Oracle7:

1/Giới thiệu:

Vấn đề sao bản dữ liệu là một trong những đặc trưng cơ bản của các ứng dụng phân tán. Nó có nghĩa là từ một bản CSDL tập chung đầu tiên người ta tiến hành copy ra một số bản sao (ảnh) cần thiết, và đặt chúng tại các vị trí thích hợp trong mạng máy tính phục vụ nhu cầu phân tán dữ liệu.

Môi trường sao bản được định nghĩa là tập hợp các đối tượng được sao bản, các ảnh, và các phương pháp để thực hiện việc sao bản. Khi tạo một môi trường sao bản chúng ta phải quan tâm tới các vấn đề sau:

-Trước tiên phải xác định rõ các đối tượng mà ta muốn sao bản. Các đối tượng phải là thành viên của một nhóm sao bản. Một nhóm sao bản có thể bao gồm các đối tượng từ nhiều lược đồ, nhưng mỗi đối tượng có thể chỉ thuộc về một nhóm đối tượng.

-Tiếp theo, ta phải xác định nơi ta muốn có các sao bản. Ta cũng cần xác định vị trí nào sẽ là vị trí chủ chứa tất cả các sao bản, và xác định vị trí nào là vị trí Snapshot sẽ chứa đựng một tập con các sao bản.

-Xác định cách thức mà Oracle dùng để truyền đi các thay đổi giữa các vị trí sao bản -đồng bộ hay không đồng bộ. Nếu chọn phương pháp không đồng bộ ta phải xác định thường xuyên ta muốn truyền đi các thay đổi như thế nào.

-Cuối cùng, ta phải phân quyền dữ liệu, ngăn ngừa xung đột khi cập nhật, hoặc ta phải chọn một phương pháp để giải quyết các xung đột. Tuy nhiên, nếu tất cả các vị trí chủ và Snapshot đều truyền các thay đổi của mình theo phương pháp đồng bộ thì ta không phải quan tâm đến vấn đề xung đột.

Sau khi đã xác định được các đối tượng muốn sao bản, ta phải có các quyền cần thiết để tạo các đối tượng tại mỗi vị trí. Hơn nữa khi đã xác định được các vị trí sẽ tạo nên môi trường sao bản, ta phải chắc chắn rằng các vị trí đó có thể liên lạc với nhau bằng việc tạo ra các CSDL kết nối cần thiết. Có ba phân cấp dành cho người sử dụng như sau:

•*Người quản trị sao bản:* Tạo ra cấu hình và bảo trì môi trường sao bản. Ta có thể tạo một người quản trị sao bản từ người quản trị tất cả các đối tượng tại một vị trí, hoặc ta có thể có nhiều người quản trị khác nhau cho các đối tượng sao bản trong từng lược đồ. Để tạo quản trị sao bản cho cho một lược đồ đơn, gọi thủ tục DBMS_REPCAT_ADMIN.GRANT_ADMIN_REPGROUP. Ví dụ: Tạo quyền quản trị sao bản cho User ACCTNG trong lược đồ ACCTNG.

```
DBMS_REPCAT_ADMIN.GRANT_ADMIN_REPGROUP
    (userid => 'acctng');
```

Có thể tạo quản trị sao bản cho tất cả các nhóm sao bản (được hiểu là quản trị sao bản tổng thể) bằng các thao tác sau:

-Tạo tài khoản quản trị sao bản. Ví dụ:

```
CREATE USER repadmin IDENTIFIED BY repadminpasword;
```

-Cho quản trị sao bản những quyền cần thiết của quản trị sao bản tổng thể bằng cách gọi thủ tục DBMS_REPCAT_ADMIN.GRANT_ADMIN_ANY_REPGROUP. Ví dụ:

```
DBMS_REPCAT_ADMIN.GRANT_ADMIN_ANY_REPGROUP
(userid => 'repadmin');
```

-Ngoài ra có thể thêm các quyền quản trị khác nếu cần thiết.

•*Symmetric replication faclity:* Bảo đảm sự hoạt động của sao bản như SYS và phải thực hiện trên các vị trí ở xa.

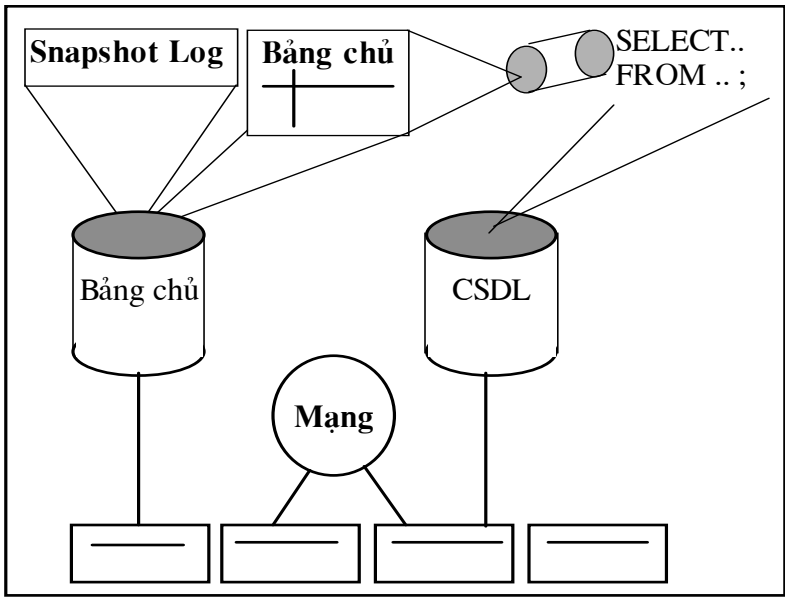
•*Người sử dụng:* Gửi các yêu cầu và cập nhật vào các đối tượng sao bản.

Thực chất, các đối tượng được sao bản là do các bản SQL DDL của chúng được sao bản. Trong một môi trường sao bản, mỗi vị trí chứa một bản sao các dữ liệu cần thiết cho các chức năng của nó thực hiện, cũng tồn tại nhiều vị trí có các bản sao của cùng một dữ liệu. Việc bảo vệ nhiều bản sao dữ liệu tại các vị trí yêu cầu một hệ thống lớn hơn, và đòi hỏi phải có sự phù hợp trong vấn đề yêu cầu các địa phương truy nhập tới dữ liệu từ nhiều địa điểm, đó là việc giải quyết các xung đột trong khi thao tác với dữ liệu của Hệ CSDL phân tán.

2/Các khái niệm cơ bản về Sao bản.

a.Sao bản cơ sở:

Sao bản cơ sở là sao bản sử dụng các Read-only Snapshot và tuân theo một dạng của vị trí sao bản đầu tiên. Dữ liệu ở các Read-only Snapshot sẽ được định kỳ làm tươi, quá trình này được so sánh tương tự như việc cập nhật các thay đổi về dữ liệu từ bảng CSDL chính của các Read-only Snapshot. Toàn bộ quá trình được thể hiện qua Hình 1:



Hình 1: Sao bản CSDL cơ sở trong Oracle

b.Các nhóm sao bản (Replication Groups):

Một đối tượng sao bản (Replication Object): Là một phần CSDL được sao tới một hay nhiều vị trí trong hệ thống phân tán. Các nhóm sao bản:

-Là đơn vị cơ bản cho việc điều khiển và quản lý tiến trình sao bản.

-Là đặc trưng tạo ra bởi người quản trị sao bản cho toàn bộ các đối tượng sao bản đó là: Kết hợp một đặc tính ứng dụng và được sao bản tới một tập các vị trí.

Oracle sử dụng ngôn ngữ thao tác dữ liệu (Data Manipulation Language : DML) để thay đổi từng sao bản trong một giao tác đảm bảo yêu cầu về tính toàn vẹn dữ liệu giữa các bảng. Oracle cho phép sao bản:

- Các bảng.
- Các đối tượng chứa các bảng: Views, Trigges, Packages, Indexes, Sequences, Synonyms.

Ngoài ra, Oracle cho phép định nghĩa, sao bản, và quản lý các nhóm sao bản của các đối tượng như một đơn vị chương trình. Có các chú ý là:

-Các thành viên của một sao bản nhóm có thể nối qua nhiều lược đồ và ngược lại, một lược đồ có thể chứa nhiều các sao bản nhóm.

-Một sao bản đối tượng có thể là thành viên của chỉ một sao bản nhóm.

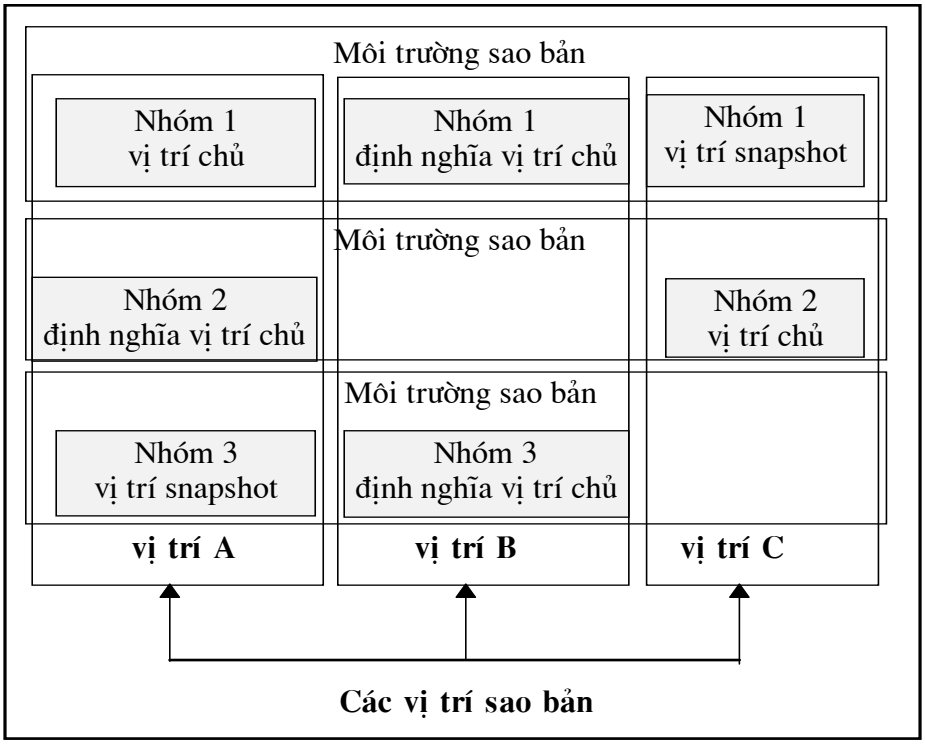
c.Các vị trí sao bản (Replication Sites):

Một nhóm sao bản có thể được sao từ một hoặc nhiều vị trí sao bản. Có hai vấn đề cơ bản cho các vị trí sao bản là:

-Các vị trí chủ (Master sites): Phải chứa một bản sao đầy đủ của tất cả các đối tượng trong sao bản nhóm. Mỗi vị trí chủ sẽ chuyển các thay đổi của nó tới vị trí chủ khác cho các nhóm sao bản.

-Các vị trí Snapshot (Snapshot sites): Là vị trí có thể chứa một hay một tập các đối tượng trong nhóm sao bản.

Các vị trí Snapshot phải có sự cộng tác với vị trí chủ (mặc dù vị trí chủ này có thể thay đổi nếu cần thiết) và không giống các vị trí chủ, các vị trí Snapshot chỉ nhận sự thay từ vị trí chủ cộng tác (Kỹ thuật truyền các thay đổi giữa vị trí chủ và vị trí Snapshot sẽ được giải thích kỹ hơn trong phần sau). Các nhóm sao bản phải có một và chỉ một định nghĩa vị trí chủ. Sự định nghĩa vị trí chủ được sử dụng như cách điều khiển cho hoạt động quản lý thực hiện, và các vị trí chủ có thể được định nghĩa từ bất kỳ vị trí nào trong hệ thống. Hình 2 thể hiện một sao bản có thể tham dự trong nhiều nhóm sao bản. Vị trí A là Snapshot cho 3, là định nghĩa vị trí chủ cho 2, và là vị trí chủ cho 1.



Hình 2 Các vị trí sao bản trong môi trường sao bản

d.Danh mục sao bản:

Sao bản sử dụng một danh mục sao bản thông tin, giống như các đối tượng được sao bản, nơi chúng được sao bản và cập nhật như thế

nào cần được truyền tới danh mục sao bản , từ đó các bảng dữ liệu có thể quay trở lại và tìm được.

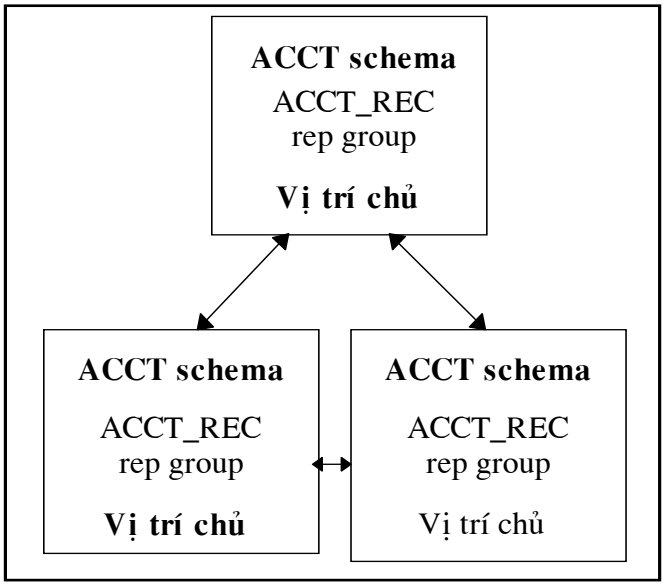
3/Phân loại môi trường sao bản:

a.Đơn chủ với nhiều vị trí Read-only Snapshot:

Có nghĩa là từ một CSDL chính, người ta tiến hành tạo ra các Read-only Snapshot khác nhau, và các bảng chính có quyền truy nhập địa phương cho nhiều vị trí Read-only Snapshot .

b.Đa sao bản chủ:

Từ một vị trí chủ người ta sao bản CSDL ra làm nhiều bản và định vị vào các vị trí mà từ đó sẽ có một tập hợp các vị trí khác sao bản dữ liệu từ nó. Toàn bộ các bảng chủ tại tất cả các vị trí có thể được cập nhật. Khi có sự thay đổi từ một bảng chủ, thay đổi đó sẽ được truyền chính xác tới tất cả các bảng chủ khác theo phương thức đồng bộ hoặc không đồng bộ.

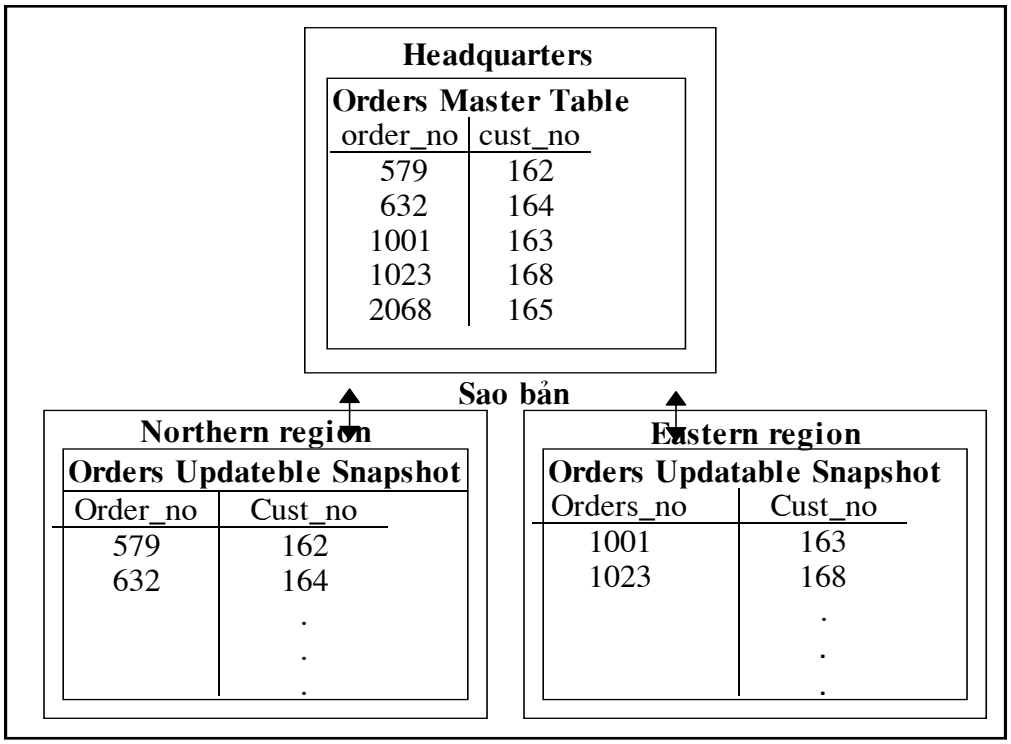


Hình 3: Đa sao bản chủ

c.Đơn chủ với đa vi trí Updatable Snapshot:

Nhiều vị trí Updatable Snapshot có thể sử dụng thông tin từ một vị trí chủ đơn (hình 4). Tuy nhiên các yêu cầu được đưa vào theo thứ tự và được xử lý tại các vị trí cố định.

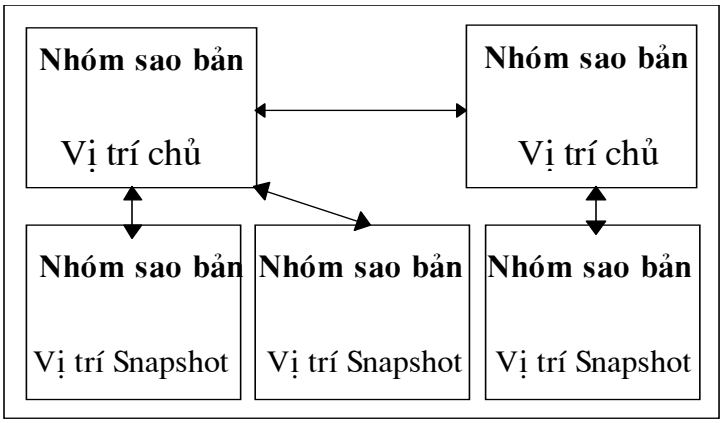
Một Updatable Snapshot được làm tươi từ bảng chủ trong một giao tác thống nhất. Các thay đổi từ Updatable Snapshot có thể được tiến hành từ bảng chủ là đồng bộ hoặc không đồng bộ. Ngoài ra các thay đổi ngay trên các Updatable Snapshot cũng được gửi tới vị trí chủ, tuy nhiên Oracle 7 chưa hỗ trợ việc tạo các Updatable Snapshot.



Hình 4:Các vị trí Updatable Snapshot sử dụng thông tin từ một bảng chủ

d.Đơn chủ với nhiều vị trí Read-Only Snapshot và đa sao bản chủ:

Là sự kết hợp giữa sao bản nhiều bản chính và nhiều các vị trí Read-only Snapshot.



Hình 5 Dạng pha trộn

Đa sao bản Snapshot chủ là: Có một số vị trí Snapshot chủ, bảng đầy đủ và tập các bảng sao bản có thể được kết hợp trong một hệ thống. Hình 5 thể hiện đa sao bản giữa hai Snapshot chủ có thể chứa sao bản bảng đầy đủ giữa hai vị trí chủ chứa hai miền. Các Snapshot có thể được định nghĩa trên các Snapshot chủ từ các bảng sao bản đầy đủ hoặc tập các bảng từ các vị trí thuộc các miền.

Lợi ích của đa sao bản: Các Snapshot có thể được điều khiển từ các vị trí chủ khác. Nếu một vị trí chủ bị lỗi, các Snapshot có thể được làm tươi từ vị trí chủ khác và tiếp tục quá trình xử lí.

e.Sự khác nhau giữa Updatable Snapshot và các sao bản chủ:

-Các sao bản chủ phải gồm dữ liệu của một bảng đầy đủ được sao bản. Các Snapshot có thể sao bản các tập con dữ liệu của bảng chủ.

-Nhiều sao bản chủ cho phép sao bản các thay đổi cho mỗi giao tác như chúng tìm thấy, trong khi các Snapshot là tập cố định, lan truyền thay đổi từ nhiều giao tác hiệu quả hơn.

-Nếu các mâu thuẫn xảy ra như kết quả của các thay đổi tới nhiều bản sao của cùng dữ liệu, các mâu thuẫn được tìm ra và giải quyết bởi các vị trí chủ.

4/Lan truyền sự thay đổi giữa các sao bản:

a.Giới thiệu:

Khi thay đổi một đối tượng trong môi trường sao bản, thay đổi này ngay lập tức sẽ được lan truyền tới tất cả các vị trí chủ, các vị trí Snapshot cũng được thay đổi tương ứng. Lựa chọn giữa hai phương pháp: Phương pháp không đồng bộ và phương pháp đồng bộ chính là sự lựa chọn giữa những thuộc tính sẵn có và những thuộc tính phức tạp hơn. Cả hai phương pháp đồng bộ và không đồng bộ đều có ưu điểm riêng theo yêu cầu và các bản sửa đổi địa phương của dữ liệu.

Với môi trường đồng bộ bạn có thể cập nhật dữ liệu tại tất cả các vị trí, các mâu thuẫn cập nhật không bao giờ xảy ra.

Với môi trường không đồng bộ, ưu điểm là sử dụng những thuộc tính sẵn có. Không có sự phụ thuộc vào các thay đổi ở một vị trí khác. Nếu có một vị trí bị lỗi bạn có thể chuyển tới vị trí khác và tiếp tục làm việc. Những vấn đề bạn cần sẽ xác định bởi một phương pháp truyền thích hợp.

b.Phương pháp đồng bộ:

Có các đặc điểm là:

-Sự thay đổi tại các vị trí sẽ lập tức phản xạ tới vị trí của bạn.

-Mặc dù trong môi trường sao bản, một dữ liệu có thể được cập nhật tại nhiều vị trí, chúng ta không phải quan tâm đến các mâu thuẫn sẽ xuất hiện trong quá trình cập nhật.

-Nếu bạn đã biết khi hệ thống mạng bị lỗi tại một vị trí sao bản, các thay đổi sẽ được truyền đồng bộ, bạn sẽ không thực hiện được việc cập nhật địa phương cho đến khi lỗi mạng được khắc phục hoặc gỡ bỏ các vị trí lỗi trong môi trường sao bản.

-Thời gian trả lời các thay đổi có thể chậm hơn, vì phải trả lời từ tất cả các vị trí trước khi chuyển giao hoặc quay lại một giao tác.

-Các thủ tục truyền dữ liệu đồng bộ là tuỳ chọn cho các môi trường với dữ liệu Read-Often/Write-Occasionally.

c.Phương pháp không đồng bộ:

Có các đặc điểm là:

-Lỗi tại các vị trí hoặc lỗi mạng không làm Snapshot hưởng các vị trí khác.

-Khả năng chịu lỗi có thể liên quan đến một số nhiệm vụ quan trọng.

-Thời gian đáp ứng các thay đổi được cải thiện hơn so với phương pháp đồng bộ vì không phải đợi đáp ứng từ một vị trí ở xa.

-Làm chậm các giao tác được lan truyền tại bất cứ khoảng cách.

-Thay đổi tại các vị trí không ngay lập tức phản xạ tới vị trí của bạn, kết quả tạm thời mâu thuẫn giữa các sao bản.

-Mâu thuẫn thay đổi có thể làm tại nhiều vị trí. Các mâu thuẫn sẽ không được tìm ra trong khi các thay đổi được truyền.

d.Lan truyền sự thay đổi vi trí dữ liệu không đồng bộ:

ORACLE sử dụng hai kỹ thuật chính để lan truyền tới sự thay đổi vị trí dữ liệu giữa các sao bản là: Làm chậm các giao tác, và làm tươi Snapshot.

d1.Làm chậm các giao tác:

Cho nhiều sao bản chủ và sao bản từ Updatable Snapshot tới các vị trí chủ. ORACLE sinh ra một Trigger và thủ tục lưu trữ bảng chứa sao bản của các vị trí dữ liệu bị thay đổi. Khi có một thay đổi địa phương các thủ tục được gọi và thực hiện việc cập nhật các thay đổi.

d2.Làm tươi Snapshot:

Các Snapshot sử dụng kỹ thuật làm chậm giao tác được miêu tả trong phần trước để lan truyền vị trí dữ liệu tới các Updatable Snapshot từ các bảng chính của chúng.

Lan truyền các thay đổi từ bảng chính tới các Read-Only Snapshot hoặc Updatable Snapshot, ORACLE sử dụng kỹ thuật làm tươi Snapshot thay cho việc sao bản vị trí hàng. Thực hiện làm tươi Snapshot:

-Một thay đổi tại vị trí chủ từ khi Snapshot được tạo hoặc được làm tươi trước đó được truyền tới Snapshot.

-Sự thay đổi các giao tác được lan truyền có hiệu quả, xử lý định hướng từng đợt.

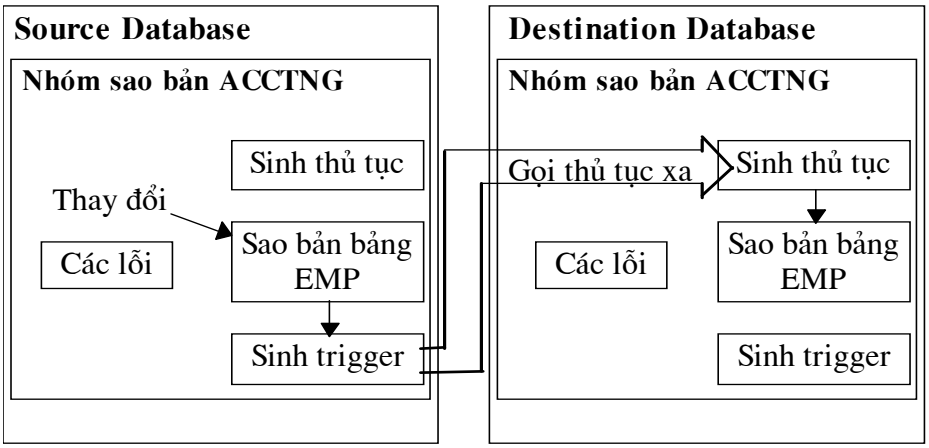
Vì Snapshot là một đơn vị cơ sở được sao hoặc làm tươi, nên nếu muốn làm tươi hai hoặc nhiều Snapshot cùng một thời điểm

thì ta chỉ cần tạo ra một nhóm các Snapshot được làm tươi (Snapshot Refresh Groups).

e.Kỹ thuật sao bản luân phiên:

Kết hợp với phương pháp làm chậm giao tác và làm tươi Snapshot, ORACLE cung cấp hai phương pháp luân phiên cho việc truyền sự thay đổi vị trí dữ liệu giữa các sao bản: Sao bản vị trí hàng đồng bộ, sao bản thủ tục.

Sao bản đồng bộ sử dụng như sao bản vị trí hàng, truyền các thay đổi vị trí dữ liệu như kỹ thuật làm chậm giao tác nhưng không sử dụng hàng đợi làm chậm giao tác. Hình 7: Khi có sự thay đổi từ một sao bản bảng, ORACLE kích hoạt một Trigger. Trigger gọi các thủ tục thực hiện tại mỗi vị trí chủ tương ứng với các thay đổi.



Hình 7 Truyền sự thay đổi vị trí dữ liệu đồng

Các thay đổi thành công gắn với bảng địa phương và một sao bản chụp từ bảng địa phương đó. Sao bản đồng bộ là có ích khi mạng máy tính ổn định và yêu cầu vị trí sao bản còn lại tiếp tục được sao bản đồng bộ.

Sao bản thủ tục: Gọi thủ tục lưu trữ được sử dụng để cập nhật dữ liệu. Sao bản thủ tục không sao bản các bảng tự cập nhật.

f.Lựa chọn phương pháp lan truyền:

Khi thêm một vị trí CSDL mới, ta phải lựa chọn cho nó một phương pháp lan truyền ngầm định vì phương pháp lan truyền này sẽ xác định cách thức vị trí mới này sẽ nhận và gửi đi các thay đổi từ tất cả các vị trí khác, và thứ tự vị trí được thêm vào là rất quan trọng. Ta có thể thay đổi bằng một phương pháp lan truyền khác khi cần bằng cách gọi DBMS_REPCAT.ALTER_MASTER_PROPAGATION. Ví dụ:

```
DBMS_REPCAT.ALTER_MASTER_PROPAGATION
(gname                    => ' acct ' ,
 master                   => ' site_a' ,
```

```
dblink_list          => ' site_b, site_c ,
propagation_mode     => ' Synchronous ') ;
```

III/Các Read-Only Snapshot:

1/Các khái niệm cơ bản về Snapshot.

a.Snapshot:

Là một yêu cầu phân tán tham chiếu tới một hay nhiều bảng chính, các View, hoặc các Snapshot khác. Mỗi sao bản của bảng chính được gọi là một Snapshot vì thông tin có được tại bất kỳ thời điểm nào có thể định kỳ được "làm tươi ", nghĩa là nó có trạng thái tương tự trạng thái mới nhất của bảng chính.

-Snapshot đơn: Là Snapshot căn cứ trên một bảng đơn lẻ ở xa và không kèm theo: Sự khác biệt hoặc tập hợp các hàm; Các nhóm (GROUP BY) hay sự kết nối (CONNECT) bởi các mệnh đề, tập câu hỏi, các kết nối, hoặc tập các phép tính toán. Ngược lại một Snapshot bao gồm các mệnh đề hoặc các phép tính được gọi là một Snapshot hoàn chỉnh.

-Read-only Snapshot: Là một bản sao đầy đủ của một bảng hay một tập các bảng. Nó là sự phản ánh đầy đủ tình trạng mới nhất của bảng chính.

-Snapshot Updatable: Có thể sửa đổi bản sao của bảng chủ và được định nghĩa bao hàm bản sao đầy đủ của bảng chủ hoặc tập các hàng trong bảng chủ.

b.Ưu điểm của các Read-Only Snapshot:

Việc bảo trì các Read-Only Snapshot của bảng chính giữa các trạm của CSDL phân tán là hữu ích vì:

+Các câu hỏi có thể được đưa ra trái ngược ở các Snapshot địa phương, kết hợp với thực hiện câu hỏi là nhanh dữ liệu được yêu cầu không phải chuyển qua mạng.

+Nếu vị trí chủ không có giá trị vì mạng bị lỗi chẳng hạn, bạn có thể tiếp tục làm việc với các bản Read-Only Snapshot của dữ liệu này.

c.Bảng so sánh Read-Only Snapshot và Updatable Snapshot:

Read-Only Snapshot	Updatable Snapshot
Chỉ cho các yêu cầu	Cho các yêu cầu và cập nhật
Loại Snapshot đơn giản hoặc hoàn chỉnh	Loại Snapshot đơn giản

2/Các thao tác chính với Read-Only Snapshot.

a.Quy tắc đặt tên cho Snapshot:

Các Snapshot được lưu trữ trong lược đồ về người sử dụng vì vậy tên của các Snapshot phải là duy nhất. Mặc dù tên của Snapshot có thể dài 30 bytes, nhưng chỉ đặt tên cho Snapshot lớn nhất là 19 bytes, quá 19 bytes ORACLE sẽ tự động cắt bỏ và thêm tổ hợp của bốn con số sao cho đảm bảo cho tên Snapshot là duy nhất.

b.Tao Read-Only Snapshot:

Muốn tạo một Snapshot ta sử dụng câu lệnh CREAT SNAPSHOT. Tương tự như việc tạo các bảng, các SNAPSHOT tạo ra có thể được định rõ sự lưu trữ các kí tự, kích thước Extent và sự phân phối, Tablespace hoặc Cluster chứa Snapshot. Cũng có thể nói rõ Snapshot sẽ được làm tươi và các yêu cầu phân tán như thế nào. Ví dụ 1:

Định nghĩa một Snapshot địa phương được sao từ bảng chính EMP định vị trên NY.

```
CREAT SNASPHOT emp_sf
PCTFREE 5 PCTUSED 60
TABLESPACE users
STORAGE (INITIAL 50K NEXT 50K PCTINCREASE 50)
REFRESH FAST
TART WITH sysda
NEXT sysdate + 7
AS SELECT * FROM scott . emp@sales . ny. com ;
```

Sau câu lệnh ORACLE tự động thực hiện bảng cơ sở với các hàng được khai báo trong các yêu cầu định nghĩa của Snapshot. Sau đó, Snapshot được làm tươi bởi câu lệnh REFRESH (Vấn đề làm tươi Snapshot sẽ được bàn kỹ hơn trong phần sau). Ta xét thêm ví dụ 2 để giải thích cách thức tạo Snapshot của ORACLE. Tổng quát hoá quá trình như sau: Khi có yêu cầu tạo Snapshot, ORACLE tạo một số các đối tượng trong lược đồ của Snapshot. Tại vị trí Snapshot, một bảng cơ sở được tạo và có tên là SNAP$_tên Snapshot, chứa các hàng được khôi phục bởi sự định nghĩa Snapshot. Cho các Snapshot đơn giản ORACLE cũng tạo một chỉ số (index) trên cột ROWID của bảng cơ sở đặt tên là I_SNAP$_tên Snapshot.

ORACLE tạo các khung nhìn (View) Read-Only của bảng cơ sở, nó được sử dụng khi có yêu cầu Snapshot. Khung nhìn này sử dụng tên là kết quả đưa ra từ câu lệnh tạo Snapshot.

ORACLE tạo ra khung nhìn địa phương thứ hai có tên là MVIEW$_tên Snapshot trên bảng chủ ở xa. ORACLE sử dụng khung nhìn này khi làm tươi Snapshot. Các kết quả của yêu cầu được lưu trữ trong bảng cơ sở, thay thế các dữ liệu Snapshot trước đó.

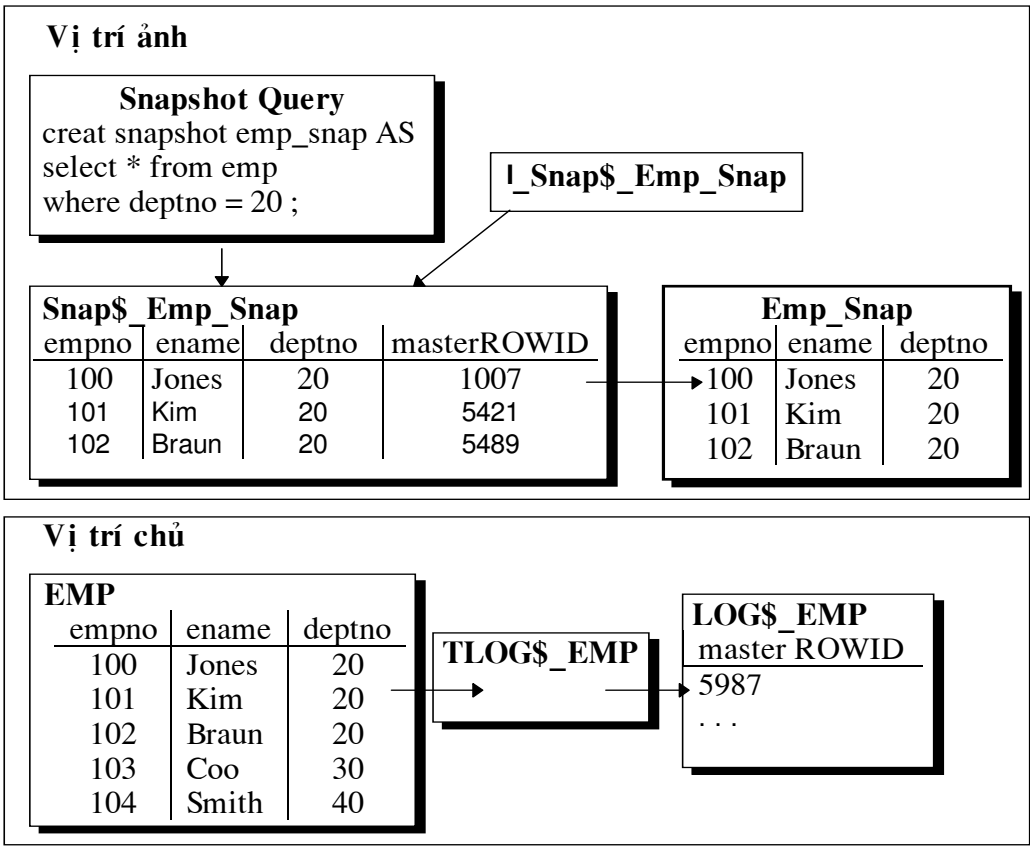
Cho các Snapshot đơn, bạn có thể chọn để tạo Snapshot log cho bảng chủ. Phần này đặt tên là TLOG$_tên bảng chủ và Trigger sử dụng cho việc thay đổi Log có tên là TLOG$_tên bảng chủ. Thông tin trong Log cho phép làm tươi nhanh Snapshot đơn.

Làm tươi nhanh Snapshot chỉ thay đổi các hàng của Snapshot. Mỗi khi có sự thay đổi bảng chủ, ORACLE tìm các thay đổi trong Snapshot log bao gồm cả ROWID của các hàng thay đổi. Việc sinh ra index (I_SNAP$) trên cột ROWID của bảng chủ cho phép các thay đổi kèm theo các Snapshot. Một Snapshot hoàn chỉnh hoặc Snapshot đơn không có Snapshot log, phải sinh lại từ bảng chính trong thời gian làm tươi Snapshot. Ví dụ 2:

Giả sử có câu lệnh tạo Snapshot như sau:

```
CREAT SNAPSHOT emp_snap AS
   SELECT * FROM emp
   WHERE deptn0 = 20;
```

Toàn bộ quá trình ORACLE được minh hoạ qua hình vẽ:



Hình: Kiến trúc Snapshot

Khi tạo một Snapshot, ta phải tuân theo các yêu cầu sau:

-Nếu tạo Snapshot trong lược đồ của riêng mình, ta phải có quyền hệ thống cho phép thực hiện các câu lệnh CREAT

SNAPSHOT, CREAT TABLE, và CREAT VIEW, cũng như SELECT trên các bảng chủ.

-Nếu tạo Snapshot trong lược đồ của User khác, ta phải có quyền CREAT ANY SNAPSHOT, cũng như SELECT trên bảng chủ. Và chủ nhân của Snapshot phải có khả năng tạo Snapshot.

c.Sửa đổi các Snapshot:

Như đối với các bảng, các Snapshot cũng có thể sửa đổi, ta có thể đặt lại các các biến lưu trữ bằng câu lệnh ALTER. Ví dụ:

ALTER SNAPSHOT emp PCTFREE 10;

Tuy nhiên để sửa đổi các biến lưu trữ , Snapshot phải nằm trong lược đồ của bạn hoặc phải có quyền ALTER ANY SNAPSHOT và ALTER ANY TABLE trong hệ thống.

d.Xoá Snapshot:

Chỉ có người là chủ hoặc các User có quyền DROP ANY SNASPHOT có thể xoá Snapshot. Ta có thể xoá Snapshot không phụ thuộc vào bảng chủ của nó hoặc Snapshot log. Câu lệnh xoá một Snapshot địa phương là DROP SNAPSHOT. Ví dụ:

DROP SNAPSHOT emp;

e.Index Snapshot:

Để tăng việc thực hiện yêu cầu khi sử dụng Snapshot, có thể tạo index cho Snapshot. Index một cột (hoặc nhiều cột) của Snapshot, ta phải index trên bảng "SNAP$" được tạo để lưu giữ các hàng của Snapshot. Ta không cần sử dụng các ràng buộc để tạo index. Ví dụ:

Sử dụng câu lệnh: CREATE index

Không sử dụng câu lệnh : CREATE unique index

f.Quản lý các Snapshot:

Điều khiển trên bảng chủ hướng tới các Snapshot:

-Toàn bộ các thay đổi tạo ra bởi các câu lệnh INSERT, UPDATE, DELETE được đưa ra từ một bảng được phản hồi trong các Snapshot khi mà các Snapshot được làm tươi.

-Nếu xoá một bảng chủ, các Snapshot được giữ nguyên tuy nhiên Snapshot log của bảng chủ cũng bị xoá. Khi ta thử làm tươi Snapshot, ORACLE sẽ thông báo lỗi.

-Nếu khôi phục lại bảng chủ, Snapshot có thể được làm tươi trở lại. Tuy nhiên ta không thể thực hiện làm tươi nhanh Snapshot, cho đến khi tạo lại Snapshot log. Nếu sau khi tạo lại bảng chủ vẫn không làm tươi được Snapshot thì phải xoá và tạo lại Snapshot.

g.Sử dụng Snapshot:

Các yêu cầu gửi tới Snapshot giống như các yêu cầu được gửi tới table hoặc View. Ví dụ:

SELECT * FROM emp;

Tuy nhiên không có thao tác dữ liệu trong bảng cơ sở của Read-only Snapshot. Ta không thể đưa câu lệnh INSERT, UPDATE, DELETE khi sử dụng Read-Only Snapshot, nếu sử dụng sẽ có thông báo lỗi, mặc dù các câu lệnh trên vẫn được đưa ra từ bảng cơ sở tới Snapshot, và làm thay đổi các Snapshot. Việc cập nhật chỉ cho phép trên bảng chủ, sau đó các Snapshot sẽ được làm tươi. Nếu muốn thay đổi Snapshot bạn phải tạo nó như một Updatable Snapshot sẽ được bàn luận trong phần sau.

h.Tạo View và Synonyms dựa trên Snapshot:

View hoặc Synonyms có thể được định nghĩa dựa trên Snapshot. Dưới đây là câu lệnh tạo một View dựa trên Snapshot EMP. Ví dụ:

```
CREAT VIEW sales_dept AS
    SELECT ename, empno
    FROM    emp
    WHERE  deptno = 10;
```

3/Updatable Snapshot:

Oracle tạo Updatable Snapshot các bước đầu tương tự như khi tạo Read-Only Snapshot và thêm hai bước sau:

-Oracle tạo một bảng đặt tên là USLOG$_tên của Snapshot chứa ROWID và timestamp (nghĩa?) của các hàng đã cập nhật trong Snapshot. The timestamp column không được cập nhật cho đến khi có một log được sử dụng trong quá trình làm tươi Snapshot.

-Oracle tạo một trigger AFTER ROW trên Snapshot dựa vào bảng chèn ROWID và timestamp của các hàng được cập nhật và xoá vào trong Updatable snapshot log. Trigger được đặt tên là USTRG$_tên snapshot.

Sự khác nhau chính giữa Read-Only Snapshot và Updatable Snapshot là Oracle tạo Read-Only View cho Read-Only Snapshot còn Writable View cho Updatable Snapshot. Ví dụ: Tạo Updatable Snapshot emp

```
CREATE SNAPSHOT emp FOR UPDATE8
  AS SELECT * FROM scott. emp@sales. ny.com
    WHERE empno > 500;
```

4/Các vấn đề cơ bản về Snapshot log.

a.Định nghĩa: Snapshot log là một bảng mà các hàng của nó ghi danh sách những thông tin được thay đổi của bảng chủ, và những thông tin về các Snapshot đã cập nhật hoặc chưa cập nhật những thay đổi trên.

Việc tạo các Snapshot log làm giảm số lượng xử lí và thời gian cần thiết để làm tươi Snapshot đơn . Snapshot log không sử dụng cho các Snapshot hoàn chỉnh.

Một Snapshot log được kết hợp với một bảng chủ; Cũng như vậy một bảng chủ có thể chỉ có một Snapshot log. Nếu nhiều Snapshot log dựa trên cùng một bảng chủ thì chúng được sử dụng như là một Snapshot log.

Tiếp sau đây chúng ta sẽ tìm hiểu cách tạo, quản lí và xoá các Snapshot log.

b.Tạo các Snapshot log:

Đặt tên Snapshot log: Oracle tự động tạo Snapshot log trong lược đồ chứa bảng chủ nếu ta không chỉ rõ tên của Snapshot log.

Tạo một Snapshot log trong CSDL như các bảng chủ sử dụng câu lệnh CREATE SNAPSHOT LOG. ta có thể đặt các tuỳ chọn vùng lưu trữ cho các đoạn dữ liệu của Snapshot log, cỡ của Extent và địa phương, các Tablespace lưu trữ Snapshot log.

Đặt tuỳ chọn vùng lưu trữ như sau:

-Đặt PCTFREE từ 0, và PCTUSED từ 100.

Đặt các biến lưu trữ Extent tuỳ theo sự cập nhật (số các câu lệnh INSERT, UPDATE, DELETE), trên bảng chủ. Ví dụ 1: Tạo Snapshot log của bảng EMP.

```
CREATE SNAPSHOT LOG ON scott.emp
    TABLESPACE users
    STORAGE (INITIAL 10K PCTINCREASE 50)
    PCTFREE 5;
```

Cách thức thực hiện của Oracle khi tạo Snapshot log:

-Oracle tạo một bảng, đặt tên là MLOG$_ tên_bảng_chủ, lưu trữ ROWID và các hàng được cập nhật trong bảng chủ.

-Oracle tạo một Trigger AFTER ROW trên bảng chủ thực hiện việc chèn ROWID và các thay đổi của các hàng vao trong Snapshot log chủ. Trigger được đặt tên là TLOG$_tên_bảng_chủ.

Điều kiện để tạo một Snapshot log:

Nếu tạo trong bảng chủ của chính mình ta cần phải có quyền CREATE TABLE và CREATE TRIGGER. Nếu tạo Snapshot log cho một bảng trong lược đồ của User khác ta phải có quyền hệ thống là CREATE ANY TABLE và CREATE ANY TRIGGER.

c.Sửa đổi các tham biến của Snapshot log:

Ta có thể thay đổi các tham biến lưu trữ của Snapshot log. Tuy nhiên chỉ có người chủ của bảng chủ, hoặc các User có quyền hệ thống là ALTER ANY TABLE có thể thay đổi. Ví dụ 2:

```
ALTER SNASPHOT LOG sale-price
```

PCTFREE 25
PCTUSED 40;

d.Xoá các Snapshot log:

Ta có thể xoá một Snapshot log độc lập với bảng chủ hoặc các Snapshot đang tồn tại. Ta có thể quyết định xoá một Snapshot log nếu các điều sau đây là đúng: Tất cả các Snapshot đơn của bảng chủ đã được xoá, và các Snapshot đơn của bảng chủ được làm tươi hoàn chỉnh, không phải là làm tươi nhanh.

Để xoá một Snapshot log địa phương, sử dụng câu lệnh DROP SNAPSHOT LOG, và chỉ chủ nhân của bảng chủ hoặc các user có quyền hệ thống DROP ANY TABLE. Ví dụ:

DROP SNAPSHOT LOG emp_log;

e.Quản lý Snapshot log:

Oracle tự động theo dõi các hàng trong Snapshot log đã được sử dụng trong suốt quá trình làm tươi của các Snapshot, và lọc các hàng từ log để cho log không tăng một cách vô hạn. Vì nhiều Snapshot đơn có thể sử dụng cùng một Snapshot log, các hàng sử dụng trong việc làm tươi của một Snapshot vẫn có thể cần được làm tươi cho Snapshot khác; Oracle không xoá các hàng trong log trừ khi tất cả các Snapshot đã sử dụng xong. Đặc điểm tự động này có thể dẫn tới sự phát triển vô hạn định một Snapshot log nếu Snapshot kết hợp với nó không bao giờ được làm tươi. Ví dụ:

Snapshot EMP_B thường xuyên được làm tươi. Nhưng Oracle không thể lọc các hàng đã sử dụng trong suốt quá trình làm tươi của Snapshot EMP_B vì Snapshot EMP_A cần chúng cho việc làm tươi sắp tới của nó. Tình huống này xảy ra khi có một số các Snapshot đơn giản dựa trên cùng một bảng chủ và:

-Một Snapsshot không được đặt tự động làm tươi bởi Oracle; Khi đó Snapshot phải được làm tươi "bằng tay".

-Một Snapshot có khoảng thời gian làm tươi lâu, có hai vấn đề là:

e1. Mạng bị lỗi ngăn cản quá trình tự động làm tươi của một hay nhiều Snapshot dựa trên bảng chủ.

e2. Mạng hoặc một vị trí lỗi ngăn cản quá trình xoá Snapshot từ bảng chủ của nó.

5/Giới thiệu về các nhóm làm tươi Snapshot:

Trong phần này ta tìm hiểu các thủ tục được cung cấp trong DBMS_REFRESH, các thủ tục này cho phép ta tạo, sửa đổi, và xoá các nhóm làm tươi, các thông tin về sự tự động làm tươi các Snapshot ...

a.Tạo nhóm làm tươi Snapshot: Ghi rõ các thành viên của nhóm và khoảng thời gian xác định khi các thành viên của nhóm cần được làm tươi, Và gọi thủ tục MAKE của DBMS_REFRESH. Ví dụ:

Tạo nhóm làm tươi ACCTG với hai thành viên ACCT_REC và ACCT_PAY. Hai Snapshot thành viên sẽ được làm tươi mỗi giờ.

```
DBMS_REFRESH.MAKE(
    name             => 'acctg' ,
    list             => 'acct_rec, acct_pay' ,
    next_date        => SYSDATE,
    interval         => 'SYSDATE +1/24' ,
    implicit_destroy =>TRUE);
```

b.Sửa đổi nhóm làm tươi Snapshot:

DBMS_REFRESH chứa các thủ tục phục vụ cho việc tạo thêm thành viên mới, di chuyển, từ nhóm là tươi, và sửa đổi tự động là tươi định kỳ cho một nhóm là tươi.

-Thêm một thành viên mới từ nhóm làm tươi: Để thêm các Snapshot vào nhóm làm tươi, gọi thủ tục ADD trong DBMS_REFRESH. Ví dụ:

```
DBMS_ REFRESH.ADD
  (name  => 'acctg' ,
   list  => 'acct_bill' ,
   lax   => TRUE);
```

Tương tự như vậy ta có thể di chuyển sửa đổi và xoá các nhóm làm tươi tuần tự theo các thủ tục sau: DBMS_REFRESH.SUBTRACT, DBMS_REFRESH.CHANCE, DBMS_REFRESH.DESTROY.

6/Vấn đề làm tươi các Snapshot:

a.Giới thiệu:

Làm tươi một Snapshot là làm cho Snapshot phản ánh được tình trạng mới nhất của bảng chủ. Oracle có hai cách làm tươi: Làm tươi nhanh và làm tươi hoàn chỉnh. Làm tươi nhanh sử dụng Snapshot log của bảng chủ làm tươi Snapshot đơn bằng cách truyền đi các thay đổi cho các Snapshot cập nhật. Chỉ các Snapshot đơn (tập hợp các hàng và các cột của một bảng đơn) có thể thực hiện làm tươi nhanh. Làm tươi hoàn chỉnh thay thế toàn bộ dữ liệu trong Snapshot đơn hoặc Snapshot đầy đủ. Cũng như vậy các Snapshot được làm tươi tự động hoặc bằng tay, làm tươi đơn lẻ hoặc làm tươi theo nhóm. Các vấn đề cần quan tâm là:

-Các Snapshot đơn nói chung sử dụng phương pháp làm tươi nhanh vì chúng mang lại hiệu quả hơn phương pháp làm tươi hoàn chỉnh.

-Nếu các bảng chủ nhận được dự báo cập nhật, nó sẽ tự động làm tươi các Snapshot theo một khoảng thời gian thích hợp.

-Sau khi chuyển các thay đổi từ các bảng chủ, thực hiện làm tươi bằng tay các Snapshot dựa vào các bảng chủ, đây là quá trình truyền các hàng mới của bảng chủ tới các Snapshot.

-Nếu ta cần làm tươi tập hợp các Snapshot từ một vị trí nào đó, cũng như có một quan hệ cha/con giữa một phần của các Snapshot, sử dụng các nhóm làm tươi Snapshot.

b.Các điều kiện để thực hiện làm tươi Snapshot:

Muốn làm tươi được Snapshot ta phải có hai điều kiện sau:

-Phải là chủ của Snapshot đó hoặc phải có quyền sửa đổi Snapshot (ALTER ANY SNAPSHOT) trong hệ thống.

-Phải có quyền vào (SELECT) bảng chủ và cho làm tươi nhanh trên Snapshot log.

c.Tự động làm tươi Snapshot:

Nếu muốn Snapshot được tự động làm tươi định kỳ ta phải thực hiện:

-Định rõ khoảng thời gian và phương pháp làm tươi.

- Phải có một hoặc nhiều tiến trình ngầm SNP giúp cho việc thực hiện định kỳ làm tươi các Snapshot.

c1.Định khoảng làm tươi Snapshot:

Nếu muốn làm tươi tự động một Snapshot, ta phải định rõ khoảng thời gian làm tươi bằng cách dùng hai tham biến START WITH và NEXT trong mệnh đề REFRESH của câu lệnh CREAT SNAPSHOT hoặc ALTER SNAPSHOT. Sau đó Oracle sẽ tự động tạo ra nhóm làm tươi chỉ chứa chính xác một Snapshot, và có tên gọi là tên của chính Snapshot mà nó chứa.

Nếu muốn làm tươi tự động một tập hợp Snapshot từ một vị trí đơn, ta phải tạo nhóm làm tươi bằng cách sử dụng thủ tục DBMS_REFRESH.MAKE. Làm tươi tự động nhóm làm tươi, cung cấp hai giá trị NEXT_DATE và INTERVAL khi ta tạo nhóm. Khi định khoảng làm tươi cho Snapshot ta phải biết:

-Hai tham biến START WITH, và NEXT (của một Snapshot đơn lẻ) hoặc INTERVAL, và NEXT_DATE (trong thủ tục gọi cho một nhóm làm tươi) chứa các kỳ hạn.phải định lượng từ một thời điểm trong tương lai. Giá trị INTERVAL được định lượng trước khi quá trình làm tươi bắt đầu. Như vậy ta phải chọn khoảng thời gian lớn hơn thời gian

yêu cầu thực hiện một lần làm tươi. Một định lượng sai phải được chứa trong lời trích dẫn.

-Nếu một Snapshot được định kỳ làm tươi trong một tập hợp các khoảng thời gian, sử dụng hai tham biến NEXT hoặc INTERVAL với kỳ hạn đơn giản từ "SYSDATE+7".

Ví dụ: Nếu ta đặt khoảng thời gian tự động làm tươi từ "SYSDATE+7" và vào ngày thứ hai, nhưng có một vài lý do xảy ra như mạng bị lỗi, Snapshot không được làm tươi cho đến ngày thứ ba. Nếu bạn muốn làm tươi nhóm Snapshot tự động theo định kỳ, không chú ý đến lần làm tươi cuối hai tham biến NEXT hoặc INTERVAL phải định rõ một kỳ hạn đơn từ "NEXT_DAY(TRUNC(SYSDATE),'MONDAY')".

Ví dụ 1: Tạo Snapshot SNAP, và nó được định kỳ làm tươi 7 ngày một lần kể từ ngày được làm tươi gần nhất , lần làm tươi đầu tiên vào ngày 01/6/1994.

```
CREATE SNAPSHOT snap
    . . .
    REFRESH COMPLETE
        START WITH '01-JUN-94'
        NEXT sysdate + 7
    AS . . . ;
```

Ví dụ 2: Nhóm làm tươi ACCT gồm ba Snapshot được định kỳ làm tươi vào thứ hai hàng tuần.

```
dbms_refresh.make(
    name                => 'acct' ,
    list                => 'sctt.acct, scott.finance,
scott.inventory' ,
    next_date           => SYSDATE ,
    interval            => 'next_day (SYSDATE + 1,
"MONDAY")'
    implicit_destroy    => TRUE ,
    lax                 => TRUE ,
```

c2.Xác đinh phương pháp làm tươi:

Khi làm tươi một Snapshot ta có thể định rõ cho Oracle thực hiện phương pháp FAST, COMPLETE, hoặc FORCED (nhanh, hoàn chỉnh, hay bắt buộc). Chỉ định một trong ba phương pháp trên bằng cách sử dụng mệnh đề REFRESH trong câu lệnh CREATE SNAPSHOT hoặc ALTER SNAPSHOT. Các Snapshot trong một nhóm làm tươi có thể không cùng chung một phương pháp làm tươi, nếu ta không định rõ phương pháp làm tươi cho từng Snapshot, Oracle sẽ tự động thực hiện làm tươi theo phương pháp FORCED (phương pháp này thực hiện nhanh hay hoàn chỉnh nếu có thể).

c3.Khởi tạo một tiến trình ngầm:

Điều kiện thuận lợi làm tươi Snapshot bằng cách sử dụng các hàng đợi công việc từ bản liệt kê sự thực hiện định kỳ của thủ tục DBMS_ REFRESH.REFRESH. Hàng đợi công việc yêu cầu có ít nhất một tiến trình ngầm SNP thực hiện. Tiến trình ngầm này làm việc định kỳ, kiểm tra hàng đợi công việc và thực hiện công việc được coi là quan trọng nhất. Tiến trình ngầm SNP được điều khiển bởi hai tham biến JOB_QUEUE_PROCESSES và JOB_QUEUE_INTERVAL.

d.Làm tươi Snapshot bằng tay:

Có hai tuỳ chọn là:

-Làm tươi một nhóm làm tươi Snapshot.

-Làm tươi một hoặc nhiều Snapshot, các Snapshot có thể hoặc không là một phần của một hay nhiều nhóm làm tươi.

d1.Làm tươi bằng tay một nhóm làm tươi:

Ta chỉ cần gọi thủ tục REFRESH trong DBMS-REFRESH. Quá trình làm tươi sẽ được thực hiện ngay lập tức thay vì phải đợi tự động làm tươi theo định kỳ và không Snapshot hưởng tới các lần tự động làm tươi nhóm về sau. Ví dụ làm tươi nhóm ACCTG:

```
DBMS_REFRESH.REFRESH('acctg');
```

d2.Làm tươi bằng tay một hay nhiều Snapshot:

Để làm tươi một hay nhiều Snapshot không phải là thành viên của cùng một nhóm làm tươi, sử dụng thủ tục REFRESH trong DBMS_SNAPSHOT sẽ cho phép ta cung cấp danh sách các Snapshot mà ta muốn làm tươi từ một giao tác trong thời điểm bất kỳ. Các Snapshot có thể thuộc về các nhóm làm tươi Snapshot khác. Việc làm tươi chúng sử dụng thủ tục này sẽ không Snapshot hưởng đến danh sách làm tươi định kỳ nếu chúng là một phần của nhóm làm tươi Snapshot định kỳ. Ví dụ sau thực hiện làm tươi hoàn chỉnh SCOTT.EMP, làm tươi nhanh SCOTT.DEPT và làm tươi ngầm định SCOTT.SALARY.

```
DBMS_ SNAPSHOT. REFRESH
  ( list   => ' scott.emp, scott.dept, scott.salary, method => 'CF' ) ;
```

IV/Vấn đề giải quyết xung đột trong Oracle 7:

Oracle luôn luôn phát hiện và ghi vào sổ nhật ký các xung đột cập nhật, các xung đột không duy nhất, và xoá các xung đột. Thêm vào, Oracle cung cấp các thủ tục giải quyết xung đột được hệ thống định nghĩa, các thủ tục hiệu lực một môi trường sử dụng sao bản mức hàng đồng bộ để giải quyết xung đột cập nhật và các xung đột không duy nhất của bạn. Phần này bao phủ các phần sau:

- Làm thế nào để phát hiện xung đột.

- Làm thế nào nhóm các cột được sử dụng trong việc phát hiện xung đột.
- Oracle hỗ trợ các phương pháp giải quyết xung đột.
- Làm thế nào để thiết kế một phương pháp giải quyết xung đột.
- Nhận diện xung đột.
- Chọn một chiến lược giải quyết xung đột.
- Sử dụng các nhóm cột.
- Sử dụng các nhóm ưu tiên.
- Sử dụng các vị trí ưu tiên.

Khi nào sử dụng các phương pháp giải quyết xung đột:

Các mục đích cho sự giải quyết xung đột là:

Đảm bảo một sự hội tụ dữ liệu.

Tránh các lỗi dây truyền.

Đảm bảo sự hội tụ là toàn bộ các vị trí trong môi trường sao bản của bạn tán thành và có cùng một dữ liệu. Tránh các lỗi dây truyền đảm bảo rằng hệ thống của bạn sẽ chạy một cách êm thấm.

Chú ý: Nếu một hoặc nhiều hàng trong một giao tác dẫn đến một xung đột không giải quyết được, toàn bộ giao tác được ghi vào sổ nhật ký lỗi. Các giao tác xảy ra sau phụ thuộc vào giao tác gốc có thể xung đột lúc này, và lần lượt, được ghi vào số nhật ký lỗi.

Nếu một hoặc nhiều vị trí trong môi trường sao bản của bạn lan truyền các thay đổi không đồng bộ, các xung đột có thể xảy ra nếu hai hoặc nhiều vị trí cập nhật cùng một dữ liệu bản sao. Thậm chí, môi trường của bạn được thiết kế để tránh xung đột (ví dụ bằng sự chia phần quyền làm chủ dữ liệu), nó là sự thận trọng để:

- Giám sát sự xuất hiện của bất kỳ xung đột không giải quyết được.
- Sử dụng một phương pháp khai báo để biết bất kỳ xung đột không chờ đợi.

Chú ý: Nếu bạn không thiết kế một phương pháp giải quyết xung đột, Oracle ghi bất kỳ một xung đột không giải quyết được như một lỗi giao tác trong DefError view của vị trí nhận.

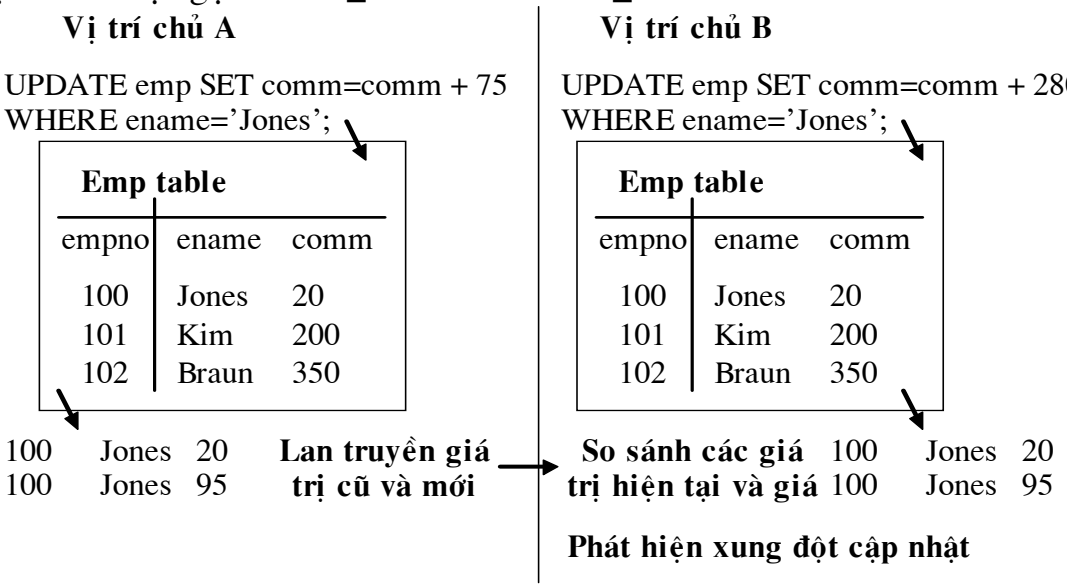
Nếu toàn bộ các vị trí của bạn lan truyền các thay đổi đồng bộ và bạn có vị trí snapshot không cập nhật, sự xung đột cập nhật không thể xảy ra, và bạn không cần thiết kế một phương pháp giải quyết xung đột.

Việc xoá các xung đột:

Khi bạn lan truyền các thay đổi trong một môi trường sao bản bởi việc cố gắng làm trì hoãn hàng đợi giao tác, symmetric replication facility của Oracle gọi một thủ tục từ xa trong gói được sinh ra tại vị trí nhận.

Oracle sử dụng thủ tục xa để phát hiện các xung đột. cho ví dụ, hai vị trí sửa đổi cùng một hàng trước sự truyền lan các cập nhật của chúng tới các vị trí khác, một xung đột xẩy ra. (Nếu bạn đặt một phương pháp giải quyết xung đột, Oracle cố gắng giải quyết xung đột). Cho mỗi hàng thay đổi Oracle sớm biết giá trị cũ và mới của từng cột trong hàng.
Như hình vẽ dưới, Oracle tại vị trí nhận so sánh giá trị cũ và giá trị hiện tại của hàng ra một xung đột nếu có bất kỳ sự khác nhau giữa các giá trị này cho bất kỳ cột nào trong hàng.

Chú ý: Vì một hàng có thể có các ROWID khác nhau tại các vị trí khác nhau, Oracle sử dụng các khoá chính của hàng để xác định hàng nào để so sánh. Nếu bạn không muốn sử dụng khoá chính, thiết kế một hay nhiều cột khác được gọi DBMS_REPLICAT.SET_COLUMNS.



Phát hiện các xung đột

Nếu thủ tục tại vị trí nhận phát hiện không xung đột, máy chủ tại vị trí nhận ghi các giá trị mới. Nếu một xung đột được phát hiện, Oracle áp dụng thủ tục giải quyết xung đột thích hợp, nếu một thủ tục là sẵn sàng. Bất kỳ xung đột không giải quyết được ghi vào nhật ký trong DefError view tại vị trí nhận.

Khi bạn sao bản một bảng sử dụng sao bản mức hàng bạn có thể thiết kế một hoặc nhiều phương pháp giải quyết xung đột. Oracle áp dụng các phương pháp này trong thứ tự ưu tiên bạn định nghĩa đến khi xung đột giải quyết xong, hoặc không còn thủ tục nào sẵn sàng.

Chú ý: Cho sao bản thủ tục bạn phải hỗ trợ một phương pháp giải quyết xung đột như phần của thủ tục sao bản của bạn.

Các kiểu xung đột:

Có ba kiểu xung đột dễ dàng được phát hiện trong sao bản đối xứng:

-Xung đột cập nhật.
-Xung đột không duy nhất.
-Xung đột xoá.

Các thủ tục tại vị trí nhận phát hiện một xung đột cập nhật nếu có một sự khác nhau giữa các giá trị cũ và các giá trị hiện thời của một hàng sao bản giống hàng tại vị trí nhận.

Một xung đột không duy nhất được phát hiện nếu một ràng buộc duy nhất bị chống lại trong khi một thao tác chèn hoặc sửa đổi của một hàng sao bản.

Một xung đột xoá được phát hiện nếu bạn thay đổi một hàng tại một vị trí xa sau khi bạn xoá hàng này từ vị trí địa phương. Xung đột xoá xẩy ra vì giá trị của hàng tại vị trí địa phương không hợp giá trị hiện thời của hàng đó tại vị trí xa.

Việc hiểu các nhóm cột:

Sự ánh xạ đối xứng sử dụng nhóm cột để phát hiện và giải quyết xung đột cập nhật. Một nhóm cột liên kết một tập các cột trong một bảng thành một cột logic đơn. Một nhóm cột có thể bao gồm một cột đơn, bất cứ số cột, hoặc toàn bộ các cột trong bảng. Mỗi cột, tuy nhiên, chỉ có thể thuộc về một nhóm cột. Có thể thiết kế một phương pháp giải quyết xung đột cho các cột bạn ấn định vào một nhóm cột.

Các nhóm cột vô hình:

Bất kỳ cột bạn không ấn định vào nhóm cột nào được tự động ấn định vào một nhóm cột vô hình cho phát hiện xung đột. Một nhóm cột vô hình không trông thấy được từ người dùng. Bạn không thể thiết kế ấn định phương pháp giải quyết xung đột tới các cột trong nhóm cột vô hình. Không sử dụng một nhóm cột vô hình cho các cột nếu bạn chờ đợi các xung đột xẩy ra trên các cột này.

Thiết kế một phương pháp giải quyết xung đột:

Có các nhóm cột cho phép bạn thiết kế các phương pháp khác để giải quyết xung đột cho các kiểu dữ liệu khác. Ví dụ, dữ liệu số thường kèm theo phương pháp giải quyết số học, và dữ liệu ký tự thường kèm theo phương pháp giải quyết nhãn thời gian.

Sự hội tụ dữ liệu đối với toàn ven dữ liệu:

Oracle đánh giá từng nhóm cột một cách riêng rẽ, một vài phần của một hàng có thể được cập nhật sử dụng dữ liệu từ vị trí gốc, khi các phần khác có thể duy trì các giá trị của dữ liệu tại vị trí đích. Khi bạn sử dụng nhiều nhóm cột, một cơ chế giải quyết xung đột có thể kết quả trong sự hội tụ dữ liệu (toàn bộ các vị trí có cùng một giá trị cho một hàng đã cho) không với sự cần thiết kết quả trong toàn vẹn dữ liệu (dữ liệu hội tụ trên một giá trị thích hợp). Cho ví dụ, nếu cột zipcode sử dụng phương pháp giải quyết

nhãn thời gian trong khi cột city sử dụng phương pháp giải quyết vị trí ưu tiên, toàn bộ các vị trí có thể hội tụ trên một zipcode phù hợp với city.

Chú ý: Nếu hai hoặc nhiều cột trong bảng phải duy trì nhất quán với lưu ý tới mỗi cái khác.

Phát hiện xung đột cập nhật trong một nhóm cột:

Khi khảo sát một hàng để xác định nếu một xung đột cập nhật đã xẩy ra, sao bản sử dụng thuật toán sau:

•Bắt đầu với nhóm cột đầu tiên, khảo sát từng trường để xác định nếu nó đã thay đổi và, nếu có một xung đột giữa các giá trị mới, cũ, hiện thời.

•Nếu không có xung đột xẩy ra, tiếp tục với nhóm cột tiếp theo. Nếu một xung đột xẩy ra, gọi một thủ tục giải quyết xung đột với ấn định thấp nhất dãy số cho nhóm cột này.

•Nếu thủ tục giải quyết xung đột giải quyết thành công xung đột, nắm giữa giá trị thích hợp cho các cột còn để treo sự xác định trạng thái.

•Nếu thủ tục không giải quyết được các xung đột, sao bản tiếp tục với thủ tục tiếp theo trong thứ tự ưu tiên cho đến khi xung đột được giải quyết hoặc không còn thủ tục nào sẵn sàng.

•Sau khi đánh giá toàn bộ các nhóm cột (bao gồm cả nhóm cột vô hình) và giải quyết thành công các lỗi, sao bản đối xứng lưu trữ giá trị mới cho các cột.

•Nếu sao bản không có khả năng giải quyết xung đột cho một nhóm cột (bao gồm nhóm cột vô hình. Không được thiết kế trong phương pháp giải quyết xung đột), nó ghi một lỗi vào DefError view và không thay đổi hàng địa phương.

Chọn một phương pháp giải quyết xung đột:

Cơ chế giải quyết xung đột khai báo của sao bản đối xứng cung cấp các thủ tục định nghĩa hệ thống cho giải quyết các xung đột cập nhật và xung đột không duy nhất.

Các thủ tục hệ thống định nghĩa không hỗ trợ các tình thế:

-Xung đột xoá.

-Thay đổi các cột khoá chính.

-Các giá trị Null trong các cột bạn chỉ ra giải quyết xung đột.

-Chống lại ràng buộc toàn vẹn tham chiếu

Cho các tình thế này, cung cấp các thủ tục giải quyết xung đột của bạn hoặc xác định một phương pháp giải quyết các lỗi này sau khi chúng được ghi vào DefError view.

Các thủ tục giải quyết xung đột cập nhật hệ thống định nghĩa:

Bảng sau chỉ ra các phương pháp giải quyết cho xung đột cập nhật đảm bảo sự hội tụ trong 3 kiểu của môi trường sao bản.

Bất kỳ số vị trí chủ (4 phương pháp)	Một hoặc hai vị trí chủ (9 phương pháp)	Một vị trí chủ và nhiều vị trí snapshot (12 phương pháp)
Thời gian chậm nhất		
Cộng thêm		
Giá trị tối thiểu (luôn luôn giảm)		
Giá trị tối đa (luôn luôn tăng)		
	Thời gian sớm nhất	
	Giá trị tối thiểu	
	Giá trị tối đa	
	Vị trí ưu tiên cao nhất	
	Giá trị ưu tiên cao nhất	
		Trung bình
		Phá huỷ từ vị trí snapshot
		Ghi đè vị trí chủ

Ghi chú: các phương pháp giải quyết bạn ấn định cần để đảm bảo sự hội tụ dữ liệu và cung cấp kết quả thích hợp cho nghiệp vụ sử dụng dữ liệu của bạn.

Các thủ tuc giải quyết xung đột ràng buộc không duy nhất hệ thống đinh nghĩa:

Symmetric replication facility cung cấp 3 phương pháp cho giải quyết xung đột không duy nhất:

-Gắn tên toàn cục của vị trí gốc tới giá trị cột từ vị trí gốc.

-Gắn vào số thứ tự sinh ra tới giá trị cột từ vị trí gốc.

-Huỷ bỏ giá trị hàng từ vị trí gốc.

Tại sao sử dụng nhiều phương pháp giải quyết:

Các nhóm cột cung cấp nhiều phương pháp cung cấp cho một hàng đơn. Bạn cũng có thể sử dụng nhiều phương pháp giải quyết xung đột cho từng nhóm cột.

-Có một hoặc nhiều phương pháp backup.

-Nhận thông báo của các xung đột.

Nhiều phương pháp giải quyết được áp dụng trong dãy bạn đặt.

Các phương pháp ưa hơn của bạn có thể không luôn luôn thích thành công. Bạn có thể chỉ ra một phương pháp backup một bộ tạo ra thay đổi giải quyết xung đột không cần thao tác thủ công xen vào.

Vài phương pháp giải hệ thống định nghĩa, như là nhãn thời gian chậm nhất, thỉng thoảng đòi hỏi một phương pháp backup để giải quyết thành công xung đột.(ưu tiên vị trí là một phương pháp backup). Phương pháp nhãn thời gian chậm nhất sử dụng một cột nhãn thời gian đặc biệt để xác định và ghép vào thay đổi mới nhất. Trong sự kiện không may bởi tại vị trí gốc và bởi tại vị trí khác thay đổi tại chính xác cùng một giây, bạn phải cung cấp phương pháp backup.

Ghi chú: Thời gian lưu trữ của Oracle là một phần nhỏ của giây. Bạn cũng có thể cung cấp một phương pháp người dùng định nghiã phương pháp và ghi thông tin xung đột hoặc dự báo xung đột DBA nếu xung đột không được giải quyết. Bạn có thể sắp đặt để nhận thông báo cho toàn bộ các xung đột, hoặc chỉ cho các xung đột không giải quyết được. Bạn có thể hoà trộn số các thủ tục giải quyết xung đột người dùng định nghĩa và hệ thống định nghĩa.

Tổng kết các phương pháp giải quyết xung đột:

Sự hội tụ có nghĩa là toàn bộ vị trí phù hợp cuối cùng trên cùng một giá trị. Bảng sau tổng kết các phương pháp giải quyết xung đột hệ thống định nghĩa, và trong bất kỳ trường hợp nào đảm bảo sự hội tụ giữa nhiều vị trí chính và các vị trí snapshot liên đới.

Kiểu xung đột	Các phương pháp giải quyết	Sự hội tụ?	Sự hội tụ với nhiều hơn 2 vị trí chủ?
Cập nhật	Tối thiểu	Có	Không, trừ khi luôn luôn giảm
	Tối đa	Có	Không, trừ khi luôn luôn tăng
	Nhãn thời gian sớm nhất	Có	Không
	Nhãn thời gian chậm nhất	Có	Có, với phương pháp backup
	nhóm ưu tiên	Có	Không, trừ khi luôn luôn tăng
	Ưu tiên vị trí	Có	Không
	Ghi đè	Có, chỉ 1 vị trí chủ	Không
	Phá huỷ	Có, chỉ 1 vị trí chủ	Không
	Trung bình	Có, chỉ 1 vị trí chủ	Không

	Cộng thêm	Có	Có
	Gán tên vị trí	Không	Không
	Gán số thứ tự	Không	Không
	Bỏ qua phá huỷ	Không	Không
Xoá	Không có		

Các phương pháp giải quyết xung đột cập nhật tối thiểu và tối đa:

Khi sao bản phát hiện một xung đột với một nhóm cột và gọi thủ tục giải quyết xung đột giá trị nhỏ nhất, nó so sánh giá trị mới từ vị trí gốc với giá trị hiện thời tại vị trí đích cho một cột đã được chỉ ra trong nhóm cột. Bạn phải thiết kế cột này khi bạn chọn thủ tục giải quyết xung đột giá trị nhỏ nhất.

Nếu giá trị mới của cột được chỉ ra là thấp hơn giá trị hiện thời, các giá trị nhóm cột từ vị trí gốc được áp dụng tại vị trí đích (lạc quan rằng tất cả các lỗi khác được giải quyết thành công cho hàng này). Nếu một giá trị mới của cột được chỉ ra lớn hơn giá trị hiện thời, xung đột được giải quyết bởi việc loại bỏ các giá trị hiện thời của các nhóm cột không thay đổi. Thiết kế một phương pháp giải quyết xung đột backup được sử dụng trong trường hợp này. Phương pháp giá trị tối đa là giống như phương pháp giá trị tối thiểu, trừ các giá trị từ vị trí gốc chỉ được áp dụng nếu giá trị của cột được chỉ ra tại vị trí gốc là lớn hơn giá trị của cột được chỉ ra tại vị trí đích. Không có hạn chế kiểu dữ liệu của các cột trong nhóm cột. Sự hội tụ cho nhiều hơn hai vị trí chính chỉ là sự đảm bảo nếu:

-Cho phương pháp tối đa, giá trị cột luôn luôn tăng.

-Cho phương pháp tối thiểu, giá trị cột luôn luôn giảm.

Ghi chú: bạn không thể làm cho có hiệu lực một giới hạn luôn luôn tăng bằng sử dụng ràng buộc kiểm tra bởi vì ràng buộc có thể gây trở ngại giải quyết xung đột.

Các phương pháp giải quyết xung đột cập nhật nhãn thời gian chậm nhất và sớm nhất:

Các phương pháp nhãn thời gian sớm nhất và chậm nhất là biến thể của các giá trị tối đa và tối thiểu. Cho phương pháp nhãn thời gian, cột chỉ định phải là kiểu DATE. Bất cứ cột nào trong nhóm cột được cập nhật, ứng dụng của bạn cập nhật giá trị của cột nhãn thời gian với SYSDATE địa phương. Cho một thay đổi áp dụng từ một vị trí khác, giá trị nhãn thời gian sẽ được đạt tới giá trị khác nhautừ vị trí gốc.

Lưu ý các dẫy sự kiện này:

1.Một khách hàng trong Phoenix gọi một người bán hàng địa phương và cập nhật thông tin địa chỉ của cô ta.

2.Sau khi nhấc điện thoại, khách hàng nhận ra rằng postal code người bán hàng địa phương cô ta giữ là sai.

3.Khách hàng cố gắng gọi cho người bán hàng địa phương với postal code chính xác, nhưng không được.

4.Khách hàng gọi cho trụ sở chính tại New York. vị trí New York khá hơn vị trí Phoenix, cập nhật thông tin địa chỉ chính xác.

5.Mạnh kết nối trụ sở chính với các người bán hàng địa phương tại Phoenix go down tạm thời.

6.Khi mạng New York/Phoenix kết nối trở lại (comes back up), Oracle coi hai cập nhật cho cùng một địa chỉ, và pháp hiện ra một xung đột tại mỗi vị trí.

7.Sử dụng phương pháp nhãn thời gian chậm nhất, Oracle chọn cập nhật gần nhất, và áp dụng địa chỉ với postal code chính xác.

Ghi chú: Nếu môi trường sao bản xuyên qua các vùng thời gian, ứng dụng của bạn chuyển đổi toàn bộ các nhãn thời gian thành một vùng thời gian chung. Nếu không, mặc dù dữ liệu của bạn sẽ hội tụ, bạn không thể áp dụng cập nhật gần nhất.

Oracle không hiệu lực đồng bộ thời gian, cái có thể được cung cấp bởi một cơ chế khác.

Phương pháp nhãn thời gian sớm nhất áp dụng các thay đổi từ vị trí với nhãn thời gian sớm nhất, và phương pháp nhãn thời gian chậm nhất áp dụng các thay đổi từ vị trí với nhãn thời gian chậm nhất.

Sự gợi ý: Chỉ ra một phương pháp backup, như một vị trí ưu tiên, được gọi nếu hai vị trí có cùng một nhãn thời gian như nhau. Cơ chế chuẩn hoá nhãn thời gian của bạn; cho ví dụ, bạn có thể chuyển nhãn thời gian sang vùng thời gian chỉ định, giống như Greenwich Mean Time (GMT).

Một đồng hồ đếm các giây như một giá trị tăng. Lạc quan rằng bạn có thể thiết kế cơ chế nhãn thời gian hợp lý và khởi tạo một phương pháp back up trong trường hợp hai vị trí có cùng một nhãn thời gian như nhau, phương pháp nhãn thời gian chậm nhất (giống phương pháp giá trị tối đa) đảm bảo sự hội tụ. Phương pháp nhãn thời gian sớm nhất, tuy nhiên, không đảm bảo cho hơn hai vị trí chính.

Các phương pháp giải quyết xung đột cập nhật cộng thêm và trung bình:

Các thủ tục cộng thêm và trung bình làm việc với các nhóm cột bao gồm một cột số đơn. Các thủ tục cộng thêm cộng các sự khác nhau giữa các giá trị cũ và mới tại vị trí gốc tới giá trị hiện thời tại vị trí đích.

giá trị hiện thời = giá trị hiện thời + (giá trị cũ giá trị mới).

Pương pháp giải quyết xung đột cộng thêm cung cấp sự hội tục cho bất kỳ số vị trí chính.

Phương pháp giải quyết xung đột trung bình tính trung bình giá trị cột mới từ vị trí gốc với giá trị hiện thời tại vị trí đích.

giá trị hiện thời = (giá trị hiện thời + giá trị mới)/2.

Phương pháp trung bình không đảm bảo sự hội tụ nếu môi trường sao bản nhiều hơn một vị trí chính. Phương pháp này hiệu quả cho một môi trường với một vị trí chính đơn và nhiều snapshot có thể cập nhật.

Các phương pháp giải quyết xung đột cập nhật nhóm ưu tiên và ưu tiên vị trí:

Các nhóm ưu tiên cho phép gán một mức ưu tiên tới từng giá trị của một cột cá thể. Nếu một xung đột được phát hiện, bảng có cột ưu tiên có giá trị thấp hơn sẽ được cập nhật sử dụng dữ liệu từ bảng với giá trị ưu tiên cao hơn.

Customer Table:

Custno	name	add1	add2	site
153	Kelly	104 First St.	Jones, ny	New_york.world
118	Klein	22 Iris Ln.	Planes, NE	Houseton.world
121	Lee	71Blue Ct.	Aspen, CO	Houseton.world
204	Potter	181 First Av.	Aspen, CO	Houseton.world

Rppriority:

Priority-Group	Priority	...	Value
Site-Priority	1		Houston.world
Site-Priority	2		New_york.world
Order-Status	1		Ordered
Order-Status	2		Shipped
Order-Status	3		Billed

Khi chọn phương pháp nhóm ưu tiên cho giải quyết xung đột, bạn chỉ ra cột nào trong bảng của bạn là cột ưu tiên.

Ưu tiên vị trí là một loại đặc biệt của nhóm ưu tiên. Với ưu tiên vị trí, cột ưu tiên bạn chỉ ra một cách tự động được cập nhật với tên CSDL toàn cục của vị trí khi cập nhật được phát sinh.

Khi bạn sử dụng ưu tiên vị trí, sự hội tụ với nhiều hơn hai vị trí chính được đảm bảo. Bạn có thể đảm bảo sự hội tụ nhiều hơn hai vị trí chính khi sử dụng các nhóm ưu tiên, tuy nhiên, nếu giá trị của cột ưu tiên là luôn luôn tăng. Điều này có nghĩa là giá trị trong cột ưu tiên phản ánh một thứ tự các sự kiện; ví dụ: ordered, shipped, billed.

Các phương pháp giải quyết xung đột cập nhật phá huỷ và ghi đè:

Các phương pháp phá huỷ và ghi đè bỏ qua các giá trị từ một trong hai vị trí gốc hoặc vị trí đích và vì vậy có thể không đảm bảo sự hội tụ với nhiều hơn một vị trí chính. Các phương pháp này được thiết kế để sử

dụng bởi một vị trí chính và nhiều vị trí snapshot, hoặc với một vài form người dùng định nghĩa.

Thủ tục ghi đè thay thế giá trị hiện thời tại vị trí đích với giá trị mới từ vị trí gốc. Ngược lại, phương pháp phá huỷ bỏ qua giá trị mới từ vị trí gốc.

Các phương pháp giải quyết xung đột không duy nhất kèm thêm sequence/kèm thêm tên vị trí:

Các kèm thêm tên vị trí và kèm thêm sequence làm bởi việc kèm thêm một chuỗi vào một cột cái sinh ra một.

Phương pháp giải quyết xung đột không duy nhất phá huỷ:

Thủ tục giải quyết xung đột không duy nhất phá huỷ giải quyết các xung đột không duy nhất bởi việc phá huỷ đơn giản hàng từ vị trí gốc hàng dẫn đến lỗi. Phương pháp này không đảm bảo sự hội tụ với nhiều vị trí chính và có thể được sử dụng với một phương tiện nhận biết. Không giống như phương pháp kèm thêm, phương pháp không duy nhất phá huỷ tối thiểu hoá sự lan truyền của dữ liệu đến khi độ chính xác được kiểm tra.

Nhận biết xung đột:

Một thủ tục nhận biết xung đột là một thủ tục giải quyết xung đột người dùng định nghĩa cung cấp một sự nhận biết, hơn là giải quyết. Bạn có thể có thông tin xung đột được ghi trong khung nhìn CSDL, hoặc bạn có thể viết một thủ tục , ví dụ, gửi một thư điện tử tới DBA (or dials a beeper).

Bạn có thể đặt sự nhận biết xẩy ra khi bạn muốn:

- Sớm như Oracle phát hiện một xung đột.
- Trước khi Oracle cố gắng một thủ tục giải quyết chỉ ra.
- Chỉ nếu Oracle không thể giải quyết xung đột.

Chú ý: Nếu xung đột không thể phân tách được để giải quyết, toàn bộ giao tác, kèm thêm bất cứ cập nhật vào bảng nhận ra, sẽ được roll back.Bạn có thể thiết kế cơ chế nhận biết của bạn để sử dụng gói Oracle DBMS_PIPES hoặc giao diện tới Oracle Office để chắc chắn sự nhận biết xẩy ra.

Khai báo phương pháp giải quyết xung đột:

Như bạn tạo ra một bảng bản sao, bạn có thể chỉ ra một hoặc nhiều phương pháp giải quyết bất kỳ xung đột tiềm ẩn nào. Để khai báo phương pháp giải quyết xung đột, đầu tiên hoàn thành pha thiết kế:

- Phân tích dữ liệu của bạn để xác định các nhóm cột được chiếm giữ, và cái mà các phương pháp giải quyết xung đột được chiếm giữ từng nhóm cột.
- Tạo các cột, giống như nhãn thời gian, và trigger bảo quản (maintenance trigger) như cần thiết bởi các phương pháp phương pháp bạn chọn.

•Nếu mong muốn, các gói nhận biết định nghĩa.

-Tạo một bảng nắm giữ thông tin nhận biết xung đột tại từng vị trí chính.

-Tạo thủ tục PL/SQL để ghi để ghi các nhận biết xung đột trong bảng.

-Cộng các giải quyết xung đột người dùng định nghĩa vào gói hoặc cộng các thủ tục để tự động nhận biết.

•Nếu bất kỳ nhóm cột trong bất kỳ bảng nào sẽ sử dụng ưu tiên vị trí hoắc nhóm ưu tiên cho giải quyết xung đột, định nghĩa các mức ưu tiên cho từng vị trí hoặc giá trị.

Sau việc thiết kế, gọi các thủ tục thích hợp trong gói DBMS_REPLICAT:

1.Nếu bạn cộng thêm giải quyết xung đột vào môi trường sao bản đã tồn tại, bạn đầu tiên phải đình chỉ toàn bộ sao bản hiệu lực. Nếu bạn tăng một nhóm các sao bản, sau việc xây dựng cho việc tăng một nhóm sao bản và các đối tượng được sao bản Bắt đầu.

2.Định nghĩa các nhóm cột cho từng bảng.

3.Phân tán các gói nhận biết xung đột và bảng ghi nhật ký nhận biết để các vị trí xa bởi việc ghi chúng như một đối tượng sao ban trong các nhóm đối tượng như vậy như bảng giám sát xung đột.

4.Gán một hoặc nhiều phương pháp xung đột cho từng nhóm cột.

5.Sinh hỗ trợ cho các bảng bản sao.

6.Tạo ra các nhãn thời gian và các trigger bảo quản nếu cần.

7.Lấy lại hiệulực sao bản.

Phát triển một chiến lược giải quyết xung đột:

Trước khi chọn hoặc viết một thủ tục giải quyết xung đột, bạn đầu tiên chắc chắn rằng bạn hoàn thành mọi thứ có thể để tránh xung đột trong vị trí đầu tiên.Đoạn này có hình dạng như thế nào để:

•Nhận biết và tránh xung đột.

•Chọn một thủ tục giải quyết xung đột thích hợp.

<u>Định nghĩa các ranh giới chức năng:</u>

Khi thiết kế một môi trường sao bản, các đường lối chỉ đạo cho thiết kế lược đồ CSDL đơn tốt áp dụng:

-ứng dụng có thể được modular, với các ranh giới chức năng và các chức năng phụ thuộc được định nghĩa không có trở ngại gì.

-Dữ liệu được chuẩn hoá để giảm tổng các chức năng phụ thuộc ẩn giữa các modul.

Để giảm các xung đột tiềm tàng sử dụng:

-Một mô hình vị trí khoá cơ bản để chia sẻ dữ liệu giữa các môdul, mô hình cho phép chỉ một modul cập nhật dữ liệu, khi các modul khác đọc dữ liệu.

-Một mô hình vị trí khoá cải tiến, nơi quyền sở hữu của dữ liệu được phân đoạn ngang.

Sử dụng khoá chính được sản sinh:

Sử dụng các số tuần tự được sản sinh làm khoá chính cho từng bảng. Bằng việc sử dụng các số tuần tự duy nhất tại mỗi vị trí, có thể tránh xung đột duy nhất và xác định chủ của các dòng dựa trên khoá chính. Mặc dù việc phân chia đơn giản các số tuần tự giữa các vị trí, điều này có thể dẫn đến vấn đề khó giải quyết như số của các vị trí, hoặc số của các thực thể tăng. Thay vào, cho phép mỗi vị trí hiệu quả thứ tự của các giá trị tuần tự, và kèm một định danh vị trí duy nhất như một phần của khoá chính.

Các phương pháp giải quyết xung đột cho quyền sở hữu động:

Nếu quyễn sở hữu vị trí khoá hoặc truy nhập phân tán vào dữ liệu là không thích hợp, cân nhắc quyền sở hữu động của dữ liệu. Quyền sở hữu động cho phép chỉ một CSDL (người chủ) cập nhật dữ liệu tại một thời điểm. Quyền sở hữu dữ liệu được phép chuyển đổi giữa các vị trí, nhưng chỉ bằng cách đảm bảo rằng người chủ có dữ liệu hầu như tân thời nhất.

Không chủ có thể có dữ liệu lỗi thời, và xung đột thứ tự có thể xẩy ra, nhưng xung đột loại này được giải quyết chính xác và dễ dàng nếu sử dụng phương nhóm ưu tiên hoặc phương pháp tối đa. Lưu ý rằng các phương pháp vị trí chủ đơn như ghi đè, có thể gây ra sự không nhất quán.

Quyền sở hữu động là rất hiệu quả trong trường hợp:

-Sự chính xác của dữ liệu là cốt yếu, và

-Có sự cạnh tranh mức thấp của dữ liệu, hoặc

-Có chỉ dẫn địa phương (vị trí đã cập nhật dữ liệu gần đây nhất là vị trí phù hợp làm cập nhật dữ liệu tiếp theo).

Sử dụng phương pháp giải quyết xung đột nhãn thời gian:

Quyền sở hữu động không cần thiết hạn định cho nhiều kiểu dữ liệu. Dữ liệu như là ngày sinh hoặc địa chỉ là ít cốt lõi cho việc điều khiển chinha xác của một ứng dụng. Do đó có thể một xung đột rất thấp. Xa hơn nữa, trên thực tế thường có kiểm tra và cân bằng bù cho thông tin lỗi thời. Thường có thể giải quyết xung đột với kiểu dữ liệu này bằng việc sử dụng phương pháp nhãn thời gian chậm nhất. (Chỉ định một phương pháp backup, như ưu tiên vị trí, trong trường hợp nhãn thời gian đồng nhất.)

Phương pháp nhãn thời gian là đặc biệt hiệu quả bởi vì dữ liệu hội tụ không quan tâm tới số vị trí, nhưng phải làm:

-Các khoá đồng bộ.

-Sử dụng vùng thời gian nhất quán.

-Chắc chắn việc tăng nhãn thời gian cho cập nhật địa phương.

Các phương pháp người dùng định nghĩa:

Phương pháp nhãn thời gian là không thích hợp cho toàn bộ dữ liệu với quyền sở hữu chia sẻ. Các thủ tục giải quyết xung đột người dùng cung cấp có thể được sử dụng khi các ngữ nghĩa của dữ liệu nào không tương xứng các thủ tục giải quyết xung đột được định nghĩa trước của Oracle đã cung cấp. Các thủ tục người dùng định nghĩa cũng có thể được sử dụng cho việc giám sát và nhận biết trong trường hợp xung đột.

Ghi chú: các phương pháp giải quyết xung đột cần đảm bảo sự hội tụ dữ liệu và cung cấp các kết quả thích hợp cho nghiệp vụ sử dụng dữ liệu của bạn.

Tránh xoá:

ứng dụng sao bản không nên lạm dụng việc xoá. Các xung đột liên quan đến việc xoá là khó giải quyết bởi vì chúng đòi hỏi một lịch sử về các dòng đã xoá. Sự sao bản đối xứng Oracle không duy trì lịch sử này. Thay thế, ứng dụng đánh dấu một dòng như được xoá. Một cách định kỳ, các dòng được đánh dấu như xoá có thể được loại bỏ.

Đặt thời gian lan truyền:

Nếu các xung đột là có thể, định nghĩa một thời gian lan truyền- khoảng thời gian tối thiểu trung bình giữa các cập nhật cùng một dòng.

Việc sử dụng các nhóm cột:

Các thủ tục sẵn sàng trong gói DBMS_REPCAT cho phép tạo và xoá các nhóm cột, và cộng thêm các thành phần, xoá các thành phần, từ một nhóm cột đang tồn tại.

Tạo một nhóm cột với các thành phần:

Để tạo một nhóm cột mới với một hoặc nhiều thành phần, gọi thủ tục MAKE_COLUMN_GROUP trong gói DBMS_REPCAT.

Cộng thêm các thành phần vào một nhóm cột đang tồn tại:

Gọi thủ tục ADD_GROUPED_COLUMN.

Xoá Các thành phần từ một nhóm cột:

Gọi thủ tục DROP_GROUPED_COLUMN.

Xoá một nhóm cột:

Gọi thủ tục DROP_COLUMN_GROUP.

Tao một nhóm cột rỗng:

Gọi thủ tục DEFINE_COLUMN_GROUP.

Chỉ định một thủ tục giải quyết xung đột cho một bảng:

Có các thủ tục riêng lẻ trong gói DBMS_REPCAT cho việc chỉ định các phương pháp để giải quyết xung đột, Sử dụng thủ tục ADD_UPDATE_RESOLUTION để chỉ định một phương pháp cho giải

quyết xung đột cập nhật cho một nhóm cột. Sử dụng thủ tục ADD_DELETE_RESOLUTION để chỉ định một phương pháp cho giải quyết xung đột xoá cho một bảng. Sử dụng thủ tục ADD_UNIQUE_RESOLUTION để chỉ định một phương pháp cho giải quyết xung đột không duy nhất liên quan đến một ràng buộc duy nhất. Bạn phải gọi thủ tục này ở vị trí định nghĩa chủ. Phương pháp giải quyết xung đột này không thực sự được cộng đến sau thời gian bạn GENERATE_REPLICATION_SUPPORT cho bảng. Bạn có thể chỉ ra nhiều phương pháp giải quyết xung đột cho một nhóm cột, một bảng, một ràng buộc. Nếu bạn cung cấp nhiều phương pháp, chúng sẽ được áp dụng trong một thứ tự đến khi xung đột dược giải quyết hoặc không còn phương pháp nào sẵn sàng. Bạn phải cung cấp một số thứ tự cho mỗi phương pháp bạn thêm vào. Bạn cũng có thể chỉ ra một trông các phương pháp chuẩn được cung cấp với symmetric replication facility, hoặc bạn có thể cung cấp tên của một chức năng bạn tự viết. Nếu bạn viết một thủ tục giải quyết xung đột bạn phải gọi DBMS_MASTER_REPOBJECT để chức năng này đảm bảo rằng nó tồn tại tại từng vị trí chủ.

Thay đổi một thủ tục giải quyết xung đột:

Để thay đổi phương pháp giải quyết xung đột cho một bảng phải làm những bước sau:

1. Đình chỉ hiệu lực sao bản đối với các nhóm đối tượng của bảng bằng việc gọi DBMS_REPCAT.SUSPEND.MASTER_ACTIVITY.
2. Gọi thủ tục DBMS_REPCAT.ADD_Conflicttype_RESOLUTION với phương pháp giải quyết xung đột mới.
3. Sinh lại hỗ trợ sao bản cho đối tượng bằng việc gọi DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT. Hoặc DBMS_REPCAT.GENERATE_REPLICATION_PACKAGE. Vì nó không cần thiết tạo lại các trigger sao bản và các gói liên kết của chúng.
4. Lấy lại hiệu lực sao bản bằng việc gọi DBMS_REPCAT.RESUME_MASTER_ACTIVITY.

Sinh các thủ tục sao bản:

Để sinh các gói hỗ trợ cho một đối tượng sao bản tại toàn bộ các vị trí chủ, như kiểm tra các bảng và các gói giải quyết xung đột, gọi thủ tục DBMS_REPCAT.GENERATE_REPLICATION_PACKAGE. Bạn phải gọi thủ tục này tại vị trí định nghĩa chủ cho đối tượng sao bản. Oracle phải tạo thành công các gói cần thiết tại các vị trí định nghĩa chủ cho thủ tục này. Các đối tượng này được tạo đồng bộ tại các vị trí chủ khác.

Bỏ một thủ tục giải quyết xung đột:

Các thủ tục riêng lẻ trong gói DBMS_REPCAT cho việc loại bỏ các thủ tục giải quyết xung đột. Sử dụng thủ tục

DROP_UPDATE.RESOLUTION để loại bỏ một thủ tục giải quyết xung đột cập nhật cho một nhóm cột. Sử dụng thủ tục DROP_DELETE.RESOLUTION để loại bỏ một thủ tục giải quyết xung đột xoá cho một bảng. Sử dụng thủ tục DROP_UNIQUE.RESOLUTION để loại bỏ một thủ tục giải quyết xung đột không duy nhất liên quan tới một ràng buộc duy nhất.

Các thủ tục này phải được gọi từ vị trí định nghĩa chủ. Thủ tục bạn chỉ ra không thực sự loại bỏ đến sau thời gian bạn GENERATE_REPLICATION_SUPPORT cho bảng.

Sử dụng các nhóm ưu tiên:

Để sử dụng phương pháp nhóm ưu tiên để giải quyết các xung đột cập nhật, đầu tiên tạo một nhóm ưu tiên, sau đó thêm các phương pháp giải quyết xung đột này cho một nhóm cột. Để tạo một nhóm ưu tiên, làm như sau:

1. Định nghĩa tên của một nhóm ưu tiên và kiểu dữ liệu của các giá trị trong nhóm.
2. Định nghĩa mức ưu tiên cho từng giá trị có thể của cột ưu tiên. Những thông tin này lưu giữ trong RepPriority view.

Một nhóm ưu tiên đơn có thể được sử dụng bởi nhiều bảng. Do đó tên bận chọn cho cột ưu tiên phải duy nhất trong một nhóm đối tượng sao bản. Cột giao tiếp với nhóm ưu tiên này có thể có các tên khác nhau trong các bảng khác nhau.

Tạo một nhóm ưu tiên:

Sử dụng thủ tục DEFINE_PRIORITY_GROUP trong gói DBMS_REPCAT.

Cộng thêm các thành phần vào một nhóm ưu tiên:

Sử dụng các thủ tục:

- ADD_PRIORITY_CHAR.
- ADD_PRIORITY_VARCHAR2.
- ADD_PRIORITY_NUMBER.
- ADD_PRIORITY_DATE.
- ADD_PRIORITY_RAW.

Phải gọi thủ tục này từ vị trí định nghĩa chủ. Và có hiệu lực sau khi chạy GENERATE_REPLICATION_SUPPORT. Nếu sửa đổi một nhóm ưu tiên gọi các thủ tục trong thứ tự sau để chắc chắn việc giải quyết xung đột thích hợp:

1. DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY.
2. DBMS_REPCAT.ADD_PRIORITY_type.
3. DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT.

4. DBMS_REPCAT.RESUME_MASTER_ACTIVITY.

Thay đổi các giá trị của một thành phần:

có một vài thủ tục cho thay đổi giá trị của các thành phần của một nhóm ưu tiên. Cú pháp ALTER_PRIORITY_type:

- ALTER_PRIORITY_CHAR.
- ALTER_PRIORITY_VARCHAR2.
- ALTER_PRIORITY_NUMBER.
- ALTER_PRIORITY_DATE.
- ALTER_PRIORITY_RAW.

Nếu sửa đổi một nhóm ưu tiên gọi các thủ tục trong thứ tự sau để chắc chắn việc giải quyết xung đột thích hợp:

1. DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY.
2. DBMS_REPCAT.ALTER_PRIORITY_type.
3. DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT.
4. DBMS_REPCAT.RESUME_MASTER_ACTIVITY.

Thay đổi mức ưu tiên của một thành phần:

Sử dụng thủ tục ALTER_PRIORITY để thay đổi mức ưu tiên liên kết với một thành phần nhóm ưu tiên.

Loại bỏ một thành phần qua giá trị:

Cú pháp DROP_PRIORITY_type:

- DROP_PRIORITY_CHAR.
- DROP_PRIORITY_VARCHAR2.
- DROP_PRIORITY_NUMBER.
- DROP_PRIORITY_DATE.
- DROP_PRIORITY_RAW.

Loại bỏ một thành phần qua ưu tiên:

Sử dụng thủ tục DROP_PRIORITY. Ví dụ:

```
DBMS_REPCAT. DROP_PRIORITY(
        gname ⇒ 'acct';
        pgroup ⇒ 'status';
        priority ⇒ 4);
```

Loại bỏ một nhóm ưu tiên:

Sử dụnh thủ tục DROP_PRIORITY_GROUP. Trước khi sử dụng thủ tục này phải gọi thủ tục DROP_UPDATE_RESOLUTION cho bất kỳ nhóm cột nào trong nhóm đối tượng sao bản sử dụng phương pháp giải quyết xung đột nhóm ưu tiên với nhóm ưu tiên này.

Sử dụng ưu tiên vị trí:

Ưu tiên vị trí là dạng đặc biệt của nhóm ưu tiên. Có nhiều thủ tục được liên kết với ưu tiên vị trí chạy như các thủ tục được liên kết với nhóm ưu tiên.

Nếu bạn chọn sử dụng phương pháp ưu tiên vị trí để giải quyết xung đột cập nhật, đầu tiên bạn phải tạo một nhóm vị trí ưu tiên trước khi cộng phương pháp giải quyết xung đột này cho một nhóm cột. Việc tạo một nhóm vị trí ưu tiên bao gồm hai bước:

1. Định nghĩa tên của nhóm vị trí ưu tiên.
2. Cộng từng vị trí vào nhóm và định nghĩa mức ưu tiên của nó. Thông tin này được cất giữ trong RepPriority view.

Nói chung bạn chỉ có một nhóm ưu tiên vị trí trong một nhóm đối tượng sao bản. Nhóm ưu tiên vị trí này có thể được sử dụng bất kỳ số lượng bảng sao bản.

Tạo một nhóm ưu tiên vi trí:

Sử dụng thủ tục DEFINE_SITE_PRIORITY.

Cộng thêm một vi trí vào nhóm:

Sử dụng thủ tục ADD_SITE_PRIORITY_SITE.

Thay đổi mức ưu tiên của một vi trí:

Sử dụng thủ tục ALTER_SITE_PRIORITY.

Thay đổi vi trí liên kết với một mức ưu tiên:

Sử dụng thủ tục ALTER_SITE_PRIORITY_SITE.

Loại bỏ một vị trí qua tên vị trí:

Sử dụng thủ tục DROP_SITE_PRIORITY_SITE.

Loại bỏ một vi trí qua mức ưu tiên:

Sử dụng thủ tục DROP_PRIORITY.

Loại bỏ một nhóm ưu tiên vị trí:

Sử dụng thủ tục DROP_SITE_PRIORITY. Chú ý trước khi gọi thủ tục này phải gọi thủ tục DROP_UPDATE_RESOLUTION cho bất kỳ nhóm cột trong nhóm đối tượng sao bản đang sử dụng phương pháp giải quyết xung đột ưu tiên vị trí với nhóm ưu tiên vị trí này.

Xem thông tin giải quyết xung đột:

Symmetric replication facility cung cấp một vài view để người dùng có thể xác định phương pháp giải quyết xung đột nào đã được sử dụng bởi từng bảng và các nhóm cột trong môi trường sao bản. Mỗi view có ba phiên bản USER_*, ALL_*, SYS.DBA_*:

- RepResolution_Method: Liệt kê toàn bộ phương pháp giải quyết xung đột sẵn sàng.
- RepColumn_Group: Liệt kê toàn bộ các nhóm cột đã định nghĩa cho CSDL.

- RepGgouped_Column: Liệt kê toàn bộ các cột trong từng nhóm cột trong CSDL.
- RepPrioity_Group: Liệt kê toàn bộ các nhóm ưu tiên và các nhóm ưu tiên vị tríđã định nghĩa trong CSDL.
- RepPrioity: Liệt kê các giá trị và việc tương ứng các mức ưu tiên cho từng ưu tiên hoặc nhóm ưu tiên vị trí.
- RepConflict: Liệt kê các kiểu xung đột (cập nhật, xoá, không duy nhất) cho một phương pháp giải quyết xung đột được chỉ ra, cho các bảng, các nhóm cột, và ràng buộc duy nhất trong CSDL.
- RepResolution: Trình bầy thêm các thông tin về phương pháp giải quyết xung đột đã sử dụng để giải quyết xung đột cho từng đối tượng.

RepParameter_Column: Trình bầy các cột được sử dụng bởi các thủ tục giải quyết xung đột để giải quyết một xung đột.

V/Công việc và hàng đợi công việc (Job, Job queue):

Trong chương này thảo luận:

- Khởi tạo các tham số cần để đặt các hàng đợi công việc sử dụng.
- Gói DBMS_JOB và các thủ tục của nó cho phép quản lý các công việc trong hàng đợi công việc.
- Các vấn đề lưu tâm với việc thực hiện công việc.
- Các view từ điển dữ liệu có thể sử dụng xem các thông tin hàng đợi công việc.

Các hàng đợi công việc:

Sử dụng hàng đợi công việc, có các thủ tục lập lịch để thực hiện một cách định kỳ. Một thủ tục chỉ có thể là mã PL/SQL. Để lập lịch một công việc (job), phải đưa nó tới một hàng đợi công việc và xác định tần số tại công việc được chạy.

Các tiến trình SNP Backgroud:

Các tiến trình background, được gọi tiến trình SNP background, thực hiện các hàng đợi công việc. Tiến trình SNP được đánh thức một cách định kỳ và thực hiện bất kỳ công việc được chạy. Phải có tối thiểu một tiến trình SNP chạy để thực hiện các công việc trong background. Tiến trình SNP background không giống các tiến trình background khác của Oracle là lỗi của một tiến trình SNP không dẫn đến instance bị lỗi. Nếu một tiến trình SNP lỗi Oracle khởi động lại chúng.

Đa tiến trình SNP:

Một instance có thể có trên 36 tiến trình SNP, được đặt tên từ SNP0 tới SNP19 và SNPA tới SNPZ. Nếu một instance có nhiều tiến trình

SNP, nhiệm vụ thực hiện các công việc có thể được chia sẻgiữa các tiến trình này. Lưu ý bao giờ mỗi công việc chạy bởi một tiến trình trong một thời điểm. Các công việc đơn không thể được chia sẻ đồng thời bởi các tiến trình SNP.

Khởi động các tiến trình SNP:

Các tham số khởi tạo hàng đợi công việc cho phép điều khiển các thao tác của các tiến trình SNP background. Đặt các tham số này trong file tham số khởi tạo cho một instance.

Tên tham số	Mô tả
JOB_QUEUE_PROCESSES	Mặc định: 0 Miền giá trị: 0..36 Đa instance: Có thể có các giá trị khác nhau Đặt số của tiến trình SNP background cho mỗi instance.
JOB_QUEUE_INTERVAL	Mặc định: 0 (giây) Miền giá trị: 1..3600(giây) Đa instance: Có thể có các giá trị khác nhau Đặt khoảng giữa các lần khởi động các tiến trình SNP background của instance.

Sử dụng các hàng đợi công việc:

Để lập lịch và quản lý các công việc trong hàng đợi công việc, sử dụng các thủ tục trong gói DBMS_JOB.

Thủ tục	Mô tả
SUBMIT	Đưa một công việc vào hàng đợi công việc.
REMOVE	Loại bỏ một công việc được chỉ định trong hàng đợi công việc.
CHANGE	Thay đổi công việc được chỉ định.Có thể thay đổi mô tả công việc, vào lúc công việc sắp được chạy, hoặc trong khoảng giữa các lần thực hiện công việc.
WHAT	Thay đổi mô tả công việc cho một công việc được chỉ định.
NEXT_DATE	Thay đổi lần thực hiện tiếp theo cho một công việc được chỉ định.
INTERVAL	Thay đổi khoảng giữa các lần thực hiện cho một công việc được chỉ định.
BROKEN	Làm mất khả năng hoặc có khả năng thực hiện công việc. Nếu một công việc được đánh dấu như công việc bị phá huỷ, Oracle không cố gắng thực hiện chúng.
RUN	Hiệu lực công việc được chỉ định để chạy.

Chia quyền và các hàng đợi công việc:

Không có quyền CSDL kết nối với việc sử dụng các hàng đợi công việc. Bất kỳ người sử dụng nào cũng có thể thực hiện các thủ tục hàng đợi công việc có thể sử dụng hàng đợi công việc.

Môi trường của một công việc:

Khi bạn đưa một công việc vào hàng đợi công việc hoặc thay đổi định nghĩa của công việc, Oracle ghi các đặc điểm của môi trường hiện tại. Các đặc điểm này như sau:

- Người sử dụng hiện tại.
- Schema hiện tại.
- Các phân quyền MAC (nếu thích hợp).
- Các tham số NLS như sau:

-NLS_LANGUAGE.
-NSL_TERRYITORY.
-NSL_CURRENCY.
-NSL_ISO_CURRENCY.
-NSL_NUMERIC_CHARACTERS.
-NSL_DATE_FORMAT.
-NSL_DATE_LANGUAGE.
-NSL_SORT.

Mỗi lần mội công việc được thực hiện, Oracle lưu giữ môi trường của công việc. Một công việc có thể thay đổi môi trường của nóbằng việc sử dụng gói DBMS_SQL và các lệnh SQL ALTER SESSION.

Nhập/Xuất (Import/Export) và các hàng đợi công việc:

Các công việc có thể được nhập và xuất. Nếu định nghĩa một công việc trong một CSDL, có thể chuyển nó sang một CSDL khác. Khi export và import các công việc, số, môi trường, và định nghĩa của công việc vẫn không được thay đổi.

Đưa một công việc vào hàng đợi công việc:

Để đưa một công việc mới vào hàng đợi công việc, sử dụng thủ tục SUBMIT trong gói DBMS_JOB. Thủ tục này sẽ trả lại số các công việc được đưa vào hàng đợi.

Người chủ công một việc:

Khi một người sử dụng đưa một công việc vào hàng đợi công việc, Oracle xác định người sử dụng như người chủ của công việc. Chỉ người chủ của công việc có thể thay đổi công việc, hiệu lực công việc để chạy, hoặc loại bỏ công việc trong hàng đợi công việc. Tuy nhiên người sử dụng đưa công việc phải có các quyền thích hợp tại vị trí xa.

Các số công việc:

Một hàng đợi công việc được định nghĩa bởi số công việc của nó. Khi đưa một công việc, số công việc được tự động sinh ra từ

SYS.JOBSEQ. Đầu tiên công việc được gán một số công việc, số này không thể thay đổi. Nếu công việc được xuất và nhập số công việc vẫn giữ nguyên.

Các định nghĩa công việc:

Định nghĩa công việc là mã PL/SQL xác định trong tham số WHAT của thủ tục SUBMIT.

Khoảng thực hiện công việc:

Thực hiện các công việc như thế nào:

Các tiến trình SNP background thực hiện các công việc. Để thực hiện một công việc, tiến trình tạo một phiên để chạy công việc. Khi một tiến trình SNP chạy một công việc, công việc được chạy.

-Trong môi trường giống môi trường công việc được đưa vào.

-Với các quyền mặc định của người chủ công việc.

Khi hiệu lực một công việc để chạy bằng việc sử dụng thủ tục DBMS_JOB.RUN, công việc được chạy bởi một tiến trình người sử dụng. Khi tiến trình người sử dụng chạy một công việc, nó được chạy với chỉ các quyền được chia trực tiếp của người sử dụng, Các quyền được chia thông qua các vai là không hiệu lực.

Các khoá hàng đợi công việc:

Oracle sử dụng các khoá hàng đợi công việc để chắc chắn rằng một công việc được thực hiện chỉ bởi một phiên tại một thời điểm. Khi một công việc chuẩn bị được chạy, phiên của nó giành được một khoá hàng đợi công việc cho công việc đó.

Các database link và các công việc:

Nếu công việc bạn đưa sử dụng một database link, database link phải kèm theo username và password. Một database link giấu tên sẽ không thành công.

Loại bỏ một công việc từ hàng đợi công việc:

Để loại bỏ một công việc từ hàng đợi công việc, sử dung thủ tục REMOVE trong gói DBMS_JOB. Các công việc đang được thực hiện sẽ không bị ngắt, có nghĩa là không loại bỏ được các công việc đang được thực hiện. Chỉ có thể loại bỏ các công việc bạn làm chủ.

Thay đổi một công việc:

Để thay đổi một công việc đã được đưa vào hàng đợi công việc, sử dụng các thủ tục CHANGE, WHAT, NEXT_DATE, hoặc INTERVAL trong gói DBMS_JOB.

Các công việc bị phá vỡ:

Một công việc được gán nhãn bị phá vỡ hoặc không bị phá vỡ. Oracle sẽ không cố gắng chạy các công việc bị phá vỡ. Có hai khả năng một công việc bị phá vỡ:

- Oracle bị lỗi để thực hiện thành công sau 16 lần cố gắng.
- Đánh giấu một công việc như một công việc bị huỷ bỏ, sử dụng thủ tục DBMS_JOB.BROKEN.

Hiệu lực một công việc sẽ được thực hiện:

Sử dụng thủ tục DBMS_JOB.RUN. Oracle sẽ tính lại ngày thực hiện tiếp theo, và đặt lại bộ đếm của số lần thực hiện lỗi của công việc trong trường hợp công việc bị phá vỡ.

Huỷ bỏ (killing) một công việc:

Có thể huỷ bỏ một công việc đang chạybằng việc đánh dấu công việc như công việc bị huỷ bỏ, xác định phiên đang chạy công việc, và huỷ bỏ kết nối với phiên này. có thể huỷ bỏ kết nối với phiên đang chạy công việc bằng việc sử dụng câu lệnh SQL ALTER SYSTEM.

Xem thông tin hàng đợi công việc:

Các view từ điển dữ liệu sau hiển thị thông tin về các công việc trong hàng đợi công việc:

- DBA_JOBS.
- USER_JOBS.
- DBA_JOBS_RUNNING.

Chương 4: Cơ Sở Dữ Liệu Phân Tán trong bài toán Wsc

I/Giới thiệu khái quát về hệ thống và các vấn đề liên quan đến hệ thống:

1/Mô hình tổ chức và mô hình mạng của Công ty cấp nước thành phố Hồ Chí Minh (WSC):

WSC là cơ quan đã có nhiều năm ứng dụng máy tính trong sản xuất và quản lý kinh doanh. Từ trước năm 1997 WSC đã sử dụng hệ máy tính IBM sau đó là các chương trình viết bằng FoxBase và FoxPro để quản lý và tính hoá đơn tiền nước.Đến năm 1997 WSC được trang bị một hệ thống mạng máy tính hiện đại đòi hỏi một hệ thống phần mềm mới, ứng dụng công nghệ hiện dại, có khả năng kết nối diện rộng, quản lý lượng khách hàng lớn và đáp ứng yêu cầu nghiệp vụ là:

-Đáp ứng 142 yêu cầu do các chuyên gia tư vấn nước ngoài đưa ra bao trùm lên các lĩnh vực chính:

1.Khách hàng.

2.Yêu cầu và khiếu nại của khách hàng.

3.Đồng hồ vật tư, thiết bị và các vị trí lắp đặt đồng hồ.

4.Biểu giá tiền nước và tiền phụ thu.

5.Chỉ số đồng hồ và xử lý hoá đơn tiền nước.

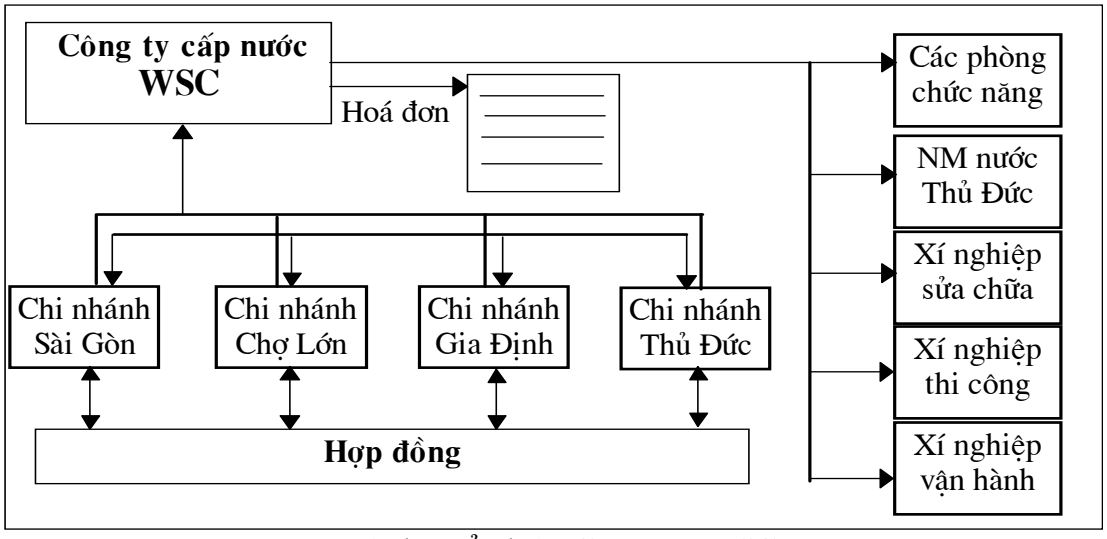
6.Thu tiền.

7.Thưởng phạt khách hàng.

8.Phiếu công tác, thi công và nhân sự

-Đáp ứng yêu cầu nghiệp vụ hiện tại.

a.Tổ chức công ty: Thể hiện qua sơ đồ:

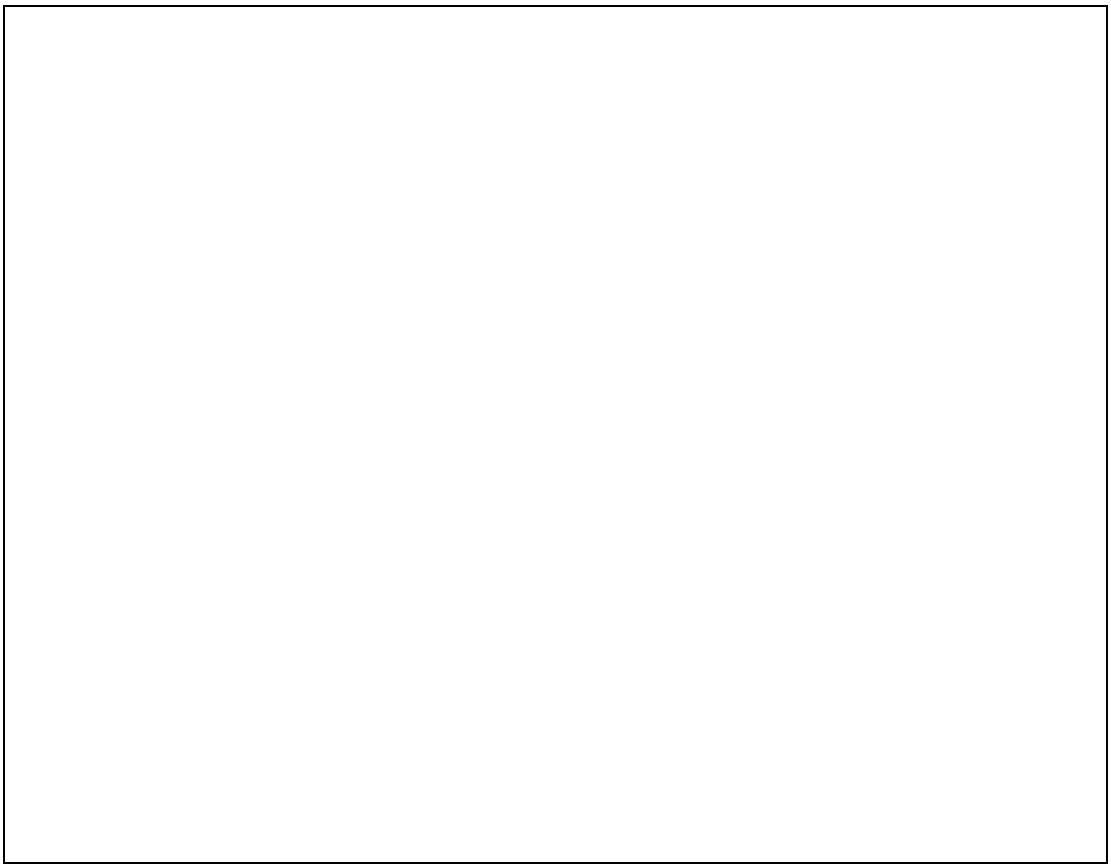


Hình: Tổ chức Công ty WSC

b.Mô hình mạng của Công ty WSC:

Tại trung tâm có hai máy chủ chính là Billing và Account được nối với nhau và chạy theo chế độ dụ phòng. Khi máy thứ nhất có sự cố, thì máy thứ hai sẽ đảm nhận nhiệm vụ của máy chủ thứ nhất. Trong trường hợp máy chủ ở chi nhánh có sự cố thì có thể khôi phục đầy đủ dữ liệu từ trung tâm.

Toàn bộ mạng máy tính của công ty WSC được thể hiện qua hình sau:



Trên cơ sở tổ chức mạng như trên, Hệ quản lý khách hàng và xử lý hoá đơn tiền nước được thiết kế theo mô hình CSDL phân tán trên môi trường Oracle.

2/Phạm vi của hệ thống:

Hệ thống đáp ứng 142 yêu cầu do chuyên gia tư vấn nước ngoài đưa ra và các yêu cầu nghiệp vụ hiện tại của WSC. Hệ được chia thành 4 phân hệ chính:

-Hệ quản lý khách hàng.
-Hệ xử lý hoá đơn và thu tiền.
-Hệ tổng hợp và phân tích thông tin.
-Hệ quản trị.

Bao gồm hơn 120 module chương trình, 70 module làm báo cáo, 30 database triggers, 105 thực thể, 83 thủ tục và hàm. Số bản ghi hệ thống phải lưu khoảng 35 000 000 bản ghi.

II.Các mô hình phân tán dữ liệu có thể áp dụng cho bài toán:

Có hai vấn đề được được đề cập trong khái niệm phân tán đó là:

+ Xử lý phân tán:
+ Dữ liệu phân tán:

Trong phạm vi của luận văn này sẽ trình bày các vấn đề liên quan đến phân tán dữ liệu. Mục đích chính là đưa ra được các giải pháp phân tán dữ liệu hiệu quả nhất cho từng ứng dụng cụ thể trong thực tế.

Dựa vào các phương pháp thiết kế phân đoạn CSDL người ta tiến hành phân tán dữ liệu theo nhiều cách khác nhau, trong đó có ba phương pháp chính thường được sử dụng là:

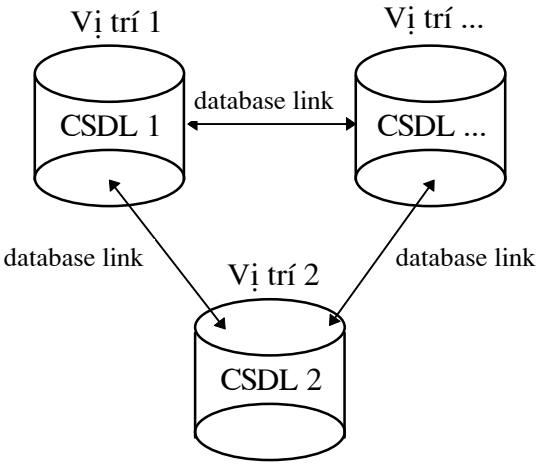
+ Phân tán dữ liệu hoàn toàn.
+ Phương pháp phân tán partition.
+ Phương pháp phân tán sử dụng replication.

1. Phân tán dữ liệu hoàn toàn.

a. Định nghĩa:

Là phương pháp thực hiện phân chia bảng dữ liệu của quan hệ tổng thể thành các phần hoàn toàn độc lập với nhau, sau đó định vị chúng vào các vị trí thích hợp theo các ứng dụng và yêu cầu thực tế.

b. Mô hình phân tán dữ liệu hoàn toàn:



Phương pháp phân tán dữ liệu hoàn toàn thường sử dụng kỹ thuật phân đoạn dọc. Các CSDL từ xa được kết nối với nhau thông qua database link.

Mỗi khi vị trí 1 muốn truy nhập tới CSDL của vị trí 2 thì thông qua database link vị trí 1 sẽ được đáp ứng qua đường truyền trực tiếp hoặc qua đường điện thoại. Tuy nhiên khối lượng dữ liệu mỗi khi cần truyền là tương đối lớn cho nên muốn áp dụng được phương pháp này thì trước hết là yêu cầu đường truyền phải đủ tốt phục vụ được nhu cầu truyền dữ liệu trong thực tế.

Không có khái niệm về các vị trí chủ trong phương pháp phân tán này, cũng như vậy sự tồn tại của vị trí trung tâm để lưu trữ toàn bộ CSDL là không cần thiết vì khi cần tổng hợp dữ liệu có thể thực hiện tại bất kỳ vị trí nào

trong hệ thống mạng của ứng dụng, dữ liệu sẽ hoàn toàn được truyền trực tiếp. Giải pháp để giữ cho dữ liệu được an toàn thì tại mỗi vị trí cần có tối thiểu hai Server trong đó có một Server hoạt động theo chế độ dự phòng hoặc chỉ cần một máy có khả năng lưu trữ toàn bộ dữ liệu của hệ thống.

Với các đặc điểm như trên phương pháp phân tán dữ liệu kiểu này tránh được dư thừa dữ liệu cao nhất, dữ liệu được phân tán thực sự tại các vị trí.

Ví dụ: Hệ thống quản lý vật tư của Công ty TNHH ABC

Thực trạng của Công ty là: Công ty TNHH ABC chuyên kinh doanh các loại vật tư. Công ty có ba chi nhánh đảm nhận công việc kinh doanh của một số loại vật tư như sau:

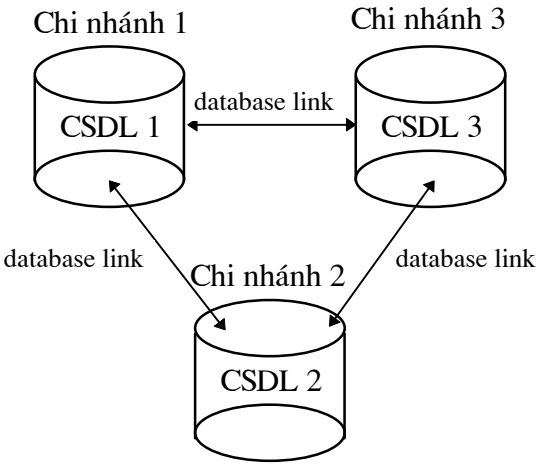
- Chi nhánh 1: Chuyên kinh doanh xi măng, sắt, thép.
- Chi nhánh 2: Chuyên kinh doanh các đồ trang trí nội thất.
- Chi nhánh 3: Chuyên kinh doanh các đồ điện gia dụng.

Các chi nhánh của Công ty nằm trong cùng một Quận của thành phố và Công đã trang bị được một hệ thống mạng nội bộ hiện đại.

Nhiệm vụ của hệ thống: Quản lý thông tin (Số lượng tồn, số lượng xuất, ...) về các loại mặt hàng của Công ty.

Giải pháp phân tán dữ liệu cho bài toán: Dựa trên thực trạng là các chi nhánh của Công ty kinh doanh các loại mặt hàng là độc lập với nhau, các chi nhánh được phân bố khá gọn đồng thời Công ty cũng đã có một hệ thống mạng cục bộ tương đối tốt. Giải pháp phân tán dữ liệu phù hợp cho bài toán này là dùng phương pháp phân tán dữ liệu hoàn toàn.

Mô hình phân tán dữ liệu của Công ty ABC:



CSDL 1: Các thông tin về mặt hàng Xi măng, Sắt, Thép.

CSDL 2: Các thông tin về mặt hàng Trang trí nội thất.

CSDL 3: Các thông tin về mặt hàng Đồ điện gia dụng.

c. Các ưu điểm của phương pháp phân tán dữ liệu hoàn toàn:

+ Xây dựng CSDL và các ứng dụng đơn giản.
+ Giảm mức độ dư thừa dư liệu.
+ CSDL thường được truyền qua đường truyền trực tiếp nên an toàn dữ liệu cao, tốc độ truyền lớn và ít xảy ra lỗi đường truyền.

d. Các nhược điểm của phương pháp phân tán dữ liệu hoàn toàn:

+ Giá thành đầu tư cho các trang thiết bị lớn: Vì giải pháp tốt nhất là phải có hệ thống mạng cục bộ với đường truyền tốt. Tuy nhiên cũng có thể truyền dữ liệu qua đường điện thoại trong trường hợp cần thiết.
+ Phạm vi phân tán hạn chế.

e. Các ứng dụng phù hợp:

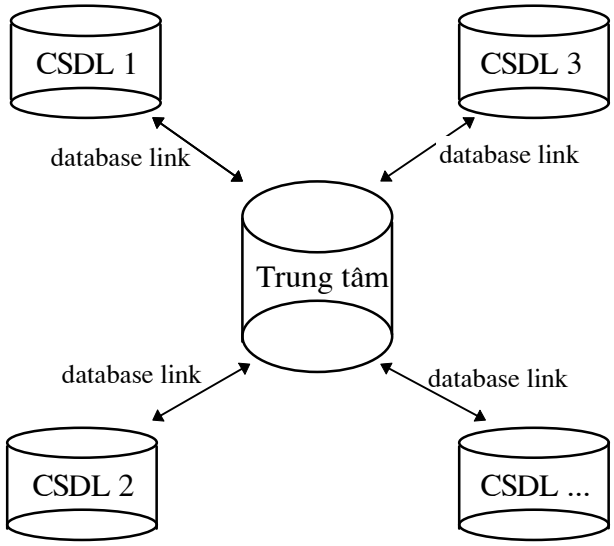
+ Các ứng dụng có CSDL nhỏ và vừa.
+ CSDL tự nó đã có sự phân chia thành các phần độc lập.
+ Nơi sử dụng các ứng dụng này phải có đường truyền tốt.

2. Phương pháp phân tán Partition.

a. Định nghĩa:

Phương pháp phân tán Partition thực hiện phân chia bảng dữ liệu của quan hệ tổng thể thành các bảng dữ liệu độc lập nhưng có cấu trúc giống hệt nhau, sau đó định vị chúng vào các vị trí thích hợp.

b. Mô hình phân tán dữ liệu của phương pháp Partition:



Như vậy phương pháp Partiton sử dụng kỹ thuật phân đoạn ngang cơ sở trong quá trình phân tán dữ liệu. Các CSDL từ xa kết nối với nhau thông qua database link.

Các khái niệm về vị trí chủ và vị trí ảnh trong phương pháp này được đề cập đến: Thông thường các CSDL được định vị tại các vị trí (trong thực tế thường là các chi nhánh), trung tâm sẽ tổng hợp CSDL tại các chi nhánh qua các Snapshot. Như vậy, các chi nhánh thường đóng vai trò là vị trí chủ và trung tâm là vị trí ảnh. Cũng có dữ liệu chỉ được cập nhật tại vị trí trung tâm, các chi nhánh muốn tra cứu sẽ qua Snapshot. Khi đó trung tâm đóng vai trò là vị trí chủ còn các chi nhánh đóng vai trò là vị trí ảnh.

Mỗi vị trí có một CSDL độc lập nhưng không giống như ở phương pháp phân tán hoàn toàn. Trong phương pháp này mỗi khi cần tổng hợp báo cáo thông tin về một loại dữ liệu nào đó thì tại vị trí trung tâm, theo định kỳ dữ liệu sẽ được làm tươi toàn bộ, phản ánh đúng tình trạng dữ liệu tại các vị trí. Sau đó mới bắt đầu công việc tổng hợp báo cáo các thông tin theo yêu cầu.

Quá trình làm tươi dữ liệu thường sử dụng phương pháp làm tươi nhanh (Chỉ cập nhật các thay đổi) do đó lượng dữ liệu truyền đi hạn chế hơn nên có thể truyền trực tiếp hoặc qua đường điện thoại.

Để đảm bảo cho các dữ liệu được an toàn, tại trung tâm phải có ít nhất hai máy chủ trong đó một máy sẽ hoạt động theo chế độ dự phòng.

Ví dụ: CSDL về Khách hàng trong WSC.

Thực trạng của Công ty WSC:

WSC có 4 chi nhánh (Sài Gòn, Gia Định, Thủ Đức, Chợ Lớn) được phân bố trên phạm vi rộng. Mỗi chi nhánh đều có nhiệm vụ quản lý Khách hàng trong khu vực của chi nhánh:

+ Chi nhánh Sài Gòn: Quản lý Khách hàng trong khu vực Sài Gòn.

+ Chi nhánh Gia Định: Quản lý Khách hàng trong khu vực Gia Định.

+ Chi nhánh Thủ Đức: Quản lý Khách hàng trong khu vực Thủ Đức.

+ Chi nhánh Chợ Lớn: Quản lý Khách hàng trong khu vực Chợ Lớn.

Ngoài ra Công ty WCA còn có một hệ thống mạng tương đối hiện đại.

Phân tích các đặc điểm dữ liệu về Khách hàng:

Công ty WCA phải quản lý một lượng Khách Hàng lớn trên diện rộng. Như vậy để tạo ra các điều kiện thuận lợi trong công tác quản lý thì ngoài giải pháp phân vùng chắc chắn không còn giải pháp nào khác.

Lựa chọn giải pháp phân tán dữ liệu:

+ Chọn phương pháp phân tán hoàn toàn: Dữ liệu về Khách Hàng tập chung ở một chi nhánh là không thể phù hợp cho công tác quản lý gây khó khăn không những cho Công ty mà còn cho cả Khách Hàng vì khoảng cách quá xa. Mặt khác nó làm ảnh hưởng đến các ứng dụng khác (tính hoá đơn ...

) của toàn bộ hệ thống vì những ứng dụng đó cũng cần có các thông tin chính xác về Khách Hàng. Và còn rất nhiều các khó khăn khác nếu dữ liệu được phân tán theo phương pháp hoàn toàn.

+ Chọn phương pháp phân tán sử dụng các replication: Chỉ trung tâm mới được cập nhật trực tiếp vào CSDL còn các chi nhánh chỉ được " tra cứu " CSDL qua các Snapshot. Như vậy các chi nhánh không thực hiện một thao tác nào đối với CSDL, quá trình xử lý đều tập chung ở trung tâm. Như vậy giải pháp này cũng sẽ gặp phải những khó khăn tương tự như giải pháp phân tán hoàn toàn.

+ Chọn phương pháp phân tán Partition: Đây chính là giải pháp phù hợp cho bài toán này, các Khách Hàng sẽ được quản lý trực tiếp tại chi nhánh thuộc chính khu vực của Khách Hàng (Khách Hàng ở Sài Gòn, Gia Định, Chợ Lớn, Thủ Đức sẽ do các chi nhánh tương ứng Sài Gòn, Gia Định, Chợ Lớn, Thủ Đức quản lý), các ứng dụng khác như tính hoá đơn cũng được thực hiện tương ứng với từng Khách Hàng trong khu vực. Trung tâm là nơi lưu trữ các dữ liệu của riêng nó và ảnh dữ liệu (Snapshot) của tất cả các vị trí phục vụ công tác quản lý và tổng hợp báo cáo...

Như vậy dữ liệu về Khách Hàng được các chi nhánh lưu trữ trong các bảng có cấu trúc tương tự như nhau (cùng có các thuộc tính: Mã Khách Hàng, tên Khách Hàng, địa chỉ, ...) chỉ các thông tin được cập nhật thật sự vào các bảng tại các chi nhánh là khác nhau.

Trong các phần sau sẽ trình bày cụ thể cách thực hiện giải pháp trên trong ứng dụng của WSC.

c. Các ưu điểm của phương pháp Partition:

+ Tránh insert một hàng sai vị trí.

+ Cho phép thực hiện nhanh hơn các thao tác: Lấy DL, sửa, tạo index... tại từng Partition do đó giảm được thời gian xử lý dữ liệu.

d. Các nhược điểm của phương pháp Partition:

+ Thực hiện phân chia dữ liệu tương đối phức tạp.

e. Các ứng dụng phù hợp:

+ Ứng dụng có lượng dữ liệu lớn.

+ Các ứng dụng có phạm vi địa lý tương đối rộng.

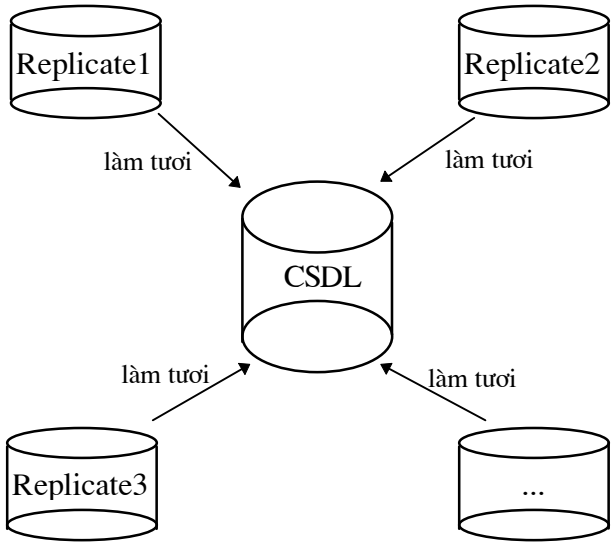
+ Các dữ liệu bị ràng buộc bởi một số điều kiện khách quan.

3. Phương pháp phân tán sử dụng các Replication.

a. Định nghĩa:

Là phương pháp sử dụng các bảng copy (còn gọi là các bảng ảnh) của một hay nhiều phần dữ liệu từ bảng chủ.

b. Mô hình phân tán dữ liệu của phương pháp phân tán dữ liệu sử dụng các Replication:



Dữ liệu được copy về tuỳ theo yêu cầu và mục đích của người sử dụng cần tra cứu như thế nào, cho nên tại các vị trí khác nhau có thể có nhiều các bản sao dữ liệu trùng lặp. Tuy nhiên cần nhấn mạnh rằng dữ liệu ảnh được tạo ra từ phương pháp này chỉ tra cứu mà không cập nhật được.

Vì yêu cầu và mục đích của người sử dụng tương đối đa dạng cho nên phương pháp phân tán sử dụng các Replication sử dụng kết hợp tất cả các kỹ thuật phân đoạn (ảnh) CSDL: Phân đoạn ngang, phân đoạn dọc và phân đoạn hỗn hợp.

Dữ liệu thường được truyền qua đường điện thoại.

Ví dụ: CSDL về Văn Bản Pháp Quy của Văn phòng Chính Phủ.

CSDL về Văn Bản Pháp Quy có đặc điểm là các thao tác làm thay đổi CSDL chỉ được thực hiện tại Văn phòng Chính Phủ nhưng được tra cứu bởi tất cả các Tỉnh, Thành Phố trong cả nước.

Dựa trên đặc điểm như trên của CSDL, nếu sử dụng hai phương pháp phân tán dữ liệu: Hoàn toàn và Partition là không hợp lệ cả về chuyên môn và tính chất kinh tế của ứng dụng.

Vậy giải pháp thích hợp cho CSDL này là sử dụng Replication.

c. Các ưu điểm của phương pháp phân tán sử dụng các Replication:

+ Dễ xây dựng CSDL cũng như các chương trình ứng dụng.
+ Truy nhập nhanh, vì thời gian truyền thông tin trên mạng giảm.
+ Có thể sử dụng đường điện thoại để truyền dữ liệu đi xa.
+ Mỗi vị trí đều có thể sử dụng toàn bộ dữ liệu của CSDL.

d. Các nhược điểm của phương pháp phân tán sử dụng các Replication:

+ Mức độ dư thừa dữ liệu cao.
+ Tăng thời gian truy nhập dữ liệu cục bộ, vì phải truy nhập trên một CSDL lớn.

e. Các ứng dụng phù hợp:

+ Các CSDL không quá lớn nhưng phạm vi địa lý ứng dụng rộng.

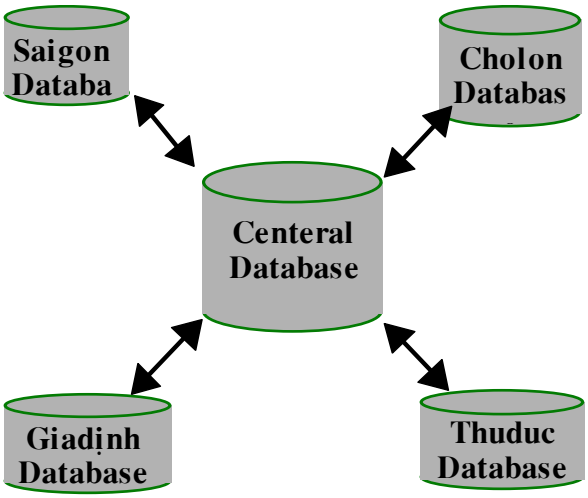
III/Mô hình phân tán dữ liệu tại WSC.

1/Phân tán chức năng hoạt động giữa trung tâm và chi nhánh tại WSC:

-Trung tâm có các chức năng sau:
+Quản lý các danh mục của hệ thống.
+Các thông số hệ thống.
+Các đơn vị trực thuộc: các chi nhánh, các nhà máy nước.
+Tạo các báo cáo phục vụ cho công việc hoạt động trên toàn công ty.

-Các chi nhánh có các chức năng sau:
+Quản lý khách hàng.
+Quản lý các dịch vụ đối với khách hàng.
+Quản lý việc đọc đồng hồ của khách hàng.
+Các báo cáo phục vụ cho công việc quản lý tại chi nhánh.

2/Mô hình dữ liệu chung tại WSC:



Sơ đồ dữ liệu

-Trung tâm: Dữ liệu tại trung tâm phải là hình ảnh đầy đủ về hoạt động của công ty.

-Chi nhánh: Là một phần con của dữ liệu tại trung tâm, sao cho dữ liệu đó đủ để chi nhánh có thể thực hiện các chức năng của mình.

3/Mô hình dữ liệu phân tán tại WSC:

Hệ thống của chúng ta sử dụng mô hình Basic Replication: Primary Site Replication và Advanced Primary Site Replication.

-Primary Site Replication: Một số bảng được quản lý tại trung tâm và có các bản sao (read-only snapshot) của chúng tại các chi nhánh. Như vậy, quyền làm chủ của bảng đó là thuộc trung tâm.

-Advanced Primary Site Replication: Các bảng còn lại được chia thành nhiều phần riêng biệt, mỗi phần được quản lý bởi một chi nhánh. Tương ứng với mỗi phần dữ liệu đó sẽ có một bản sao tại trung tâm, các bản sao của các phần riêng biệt của cung một bảng tại trung tâm sẽ được gộp lại thành một đối tượng duy nhất (bản sao tổng hợp của các chi nhánh) để phục vụ cho quá trình xử lý dữ liệu.

4/Các chú ý khi tạo các bảng tại các chi nhánh trong cấu hình CSDL phân tán:

-Đảm bảo các ràng buộc: Các ràng buộc của các bảng tại các chi nhánh (tức là tại một CSDL) đương nhiên phải được đảm bảo. Ngoài ra các ràng buộc này phải được đảm bảo trên toàn bộ hệ thống dữ liệu (bao gồm nhiều CSDL). Oracle Server tại một CSDL không thể đảm bảo được điều này, do đó ta phải thiết kế một cách hợp lý để đảm bảo được điều này.

-Phải xác định được nơi phát sinh dữ liệu cho một bảng nhất định (tại các chi nhánh, tại trung tâm hoặc cả hai).

-Tên của các bảng: Các module tại các chi nhánh khi thực hiện thao tác lên CSDL có một yêu cầu là tên các đối tượng được sử dụng phải ổn định mặc dù đối tượng đó có thể có kiểu khác nhau (table, view, synonym, snpashot...). Trong khi đó các bảng được tạo ra chưa chắc đã là các đối tượng truy nhập trực tiếp của các module, mà nó phải kết hợp với các đối tượng mới, do đó các bảng có thể phải đổi tên để cho các đối tượng có thể sử dụng tên đó. Như vậy các module mới có thể truy nhập trực tiếp các đối tượng.

Để đảm bảo cho mô hình dữ liệu phân tán đáp ứng được các chú ý trên ta phải thực hiện việc phân tích từng bảng trong hệ thống để có được những thông tin cần thiết trong bảng sau:

Table name	Owner	Constraints	Sequences

Chú thích:

-Table name: Là tên bảng theo thiết kế hiện tại.

-Owner: Xác định xem quyền làm chủ bảng sẽ là trung tâm hay chi nhánh. Giá trị cho cột này có thể là một trong hai giá trị 'Trung tâm' hoặc 'Chi nhánh'. Quyền làm chủ ở đây có nghĩa là các thao tác Update lên bảng sẽ do phía nào thực hiện. Nếu một bảng nào đó có thể Update được từ cả trung tâm và chi nhánh thì mô hình dữ liệu phân tán ở trên sẽ không áp dụng được.

-Constrains: Liệt kê tất cả các ràng buộc trên bảng theo thứ tự và khuôn dạng sau:

Primary key: PK_NAME(column1,column2,...)
Unique key: UK_NAME(column1,column2,...)
Foreign key: FK_NAME(col1,col2,...) => REF_TABLE(ref_col1,ref_col2,...)

-Sequences: Liệt kê tất cả các sequence mà bảng có sử dụng đến theo khuôn dạng sau: SEQ_NAME(column_use_this_sequence)

Ví dụ 1:

ABC: Assign billing cycle to customers.

Owner: Branch.

Constraints:

+ ASS_BCY_PK(CUST_ID, STT)
+ ASS_BCY_BLL_CYS_FK (BRANCH_ID,BC_CODE)=> BILLING_CYC (BARANCH_ID, CODE)
+ ASS_BCY_CUST_FK (CUST_ID) => CUSTOMERS(ID)

Sequences:

Notes:

Names:

+ Branch: Table ABC
+ Center: Snapsshot ABCTD, ABCSG ...; View ABC

Ví dụ 2: ACTIVITIES_CUST_PARAMETERS

Owner: Center

Constraints:

+ ACTIVITY_C_PK(ID,CPRS_ID)
+ ACTIVITY_C_ ACTIVITY_FK(ID) => ACTIVITY(ID)
+ ACTIVITY_C_CPRS_FK (CPRS_ID)
  => CUST_ PARAMETERS(CPRS_ID)

Sequences:

Notes: Định nghĩa tại trung tâm và được sử dụng tại các chi nhánh trong các giao dịch với khách hàng.

a.Cách đặt tên:

-Nếu một bảng có nguồn gốc từ chi nhánh thì cách đặt tên nó như sau:

Tại các chi nhánh có các bảng cùng tên.

Tại trung tâm có các Snapshot có tên tạo bởi tên bảng ghép với dấu $ và mã chi nhánh, và một View có cùng tên với bảng tổng hợp dữ liệu từ các Snapshot.

Ví dụ : Bảng ABC có nguồn gốc từ các chi nhánh do đó nó có cách tổ chức như sau:

Tại các chi nhánh Sài Gòn, Chợ Lớn, Thủ Đức, Gia Định có các bảng tên là ABC. *Tại trung tâm* có 4 snapshot là ABCSG, ABCCL, ABCTD, ABCGD lấy dữ liệu từ các bảng ở các chi nhánh theo câu lệnh sau:

```
CREATE SNAPSHOT ABC$SG AS
    SELECT * FROM ABC@saigon;
```

và một View tổng hợp dữ liệu từ 4 chi nhánh:

```
CREATE VIEW ABC AS
    SELECT *FROM ABC$SG
    UNION
    SELECT *FROM ABC$CL
    UNION
    SELECT *FROM ABC$TD
    UNION
    SELECT *FROM ABC$GD;
```

-Nếu một bảng có nguồn gốc từ trung tâm và dữ liệu của nó không phụ thuộc vào từng chi nhánh thì nó có cách đặt tên như sau:

Tại trung tâm có một bảng như tên đã định.

Tại mỗi chi nhánh có một Snapshot có cùng tên.

Ví dụ: Bảng EXCHANGES có nguồn gốc từ trung tâm.

Tại trung tâm có một bảng có tên là EXCHANGES.

Tại mỗi chi nhánh có một Snapshot có tên EXCHANGES và được tạo ra như sau:

```
CREATE SNAPSHOT EXCHANGES
  AS SELECT * FROM EXCHANGES@wsc;
```

-Nếu bảng có nguồn gốc trung tâm nhưng bản thân dữ liệu lại phụ thuộc từng chi nhánh thì nó cùng có cách đặt tên như trên nhưng khác ở câu lệnh tạo Snapshot.

Ví dụ: Tạo Snapshot BILLS tại chi nhánh Chợ Lớn

```
CREATE SNAPSHOT BILLS AS
   SELECT * FROM BILLS@wsc
    WHERE BRANCH_CODE = 'CL';
```

b.Cách tạo các FK:

Tại các chi nhánh các FK có dính dáng đến các Snapshot (đến hoặc đi từ) vẫn được giữ nguyên, chỉ có tên bảng chứa Snapshot được sử dụng thay vì tên Snapshot.

Tại trung tâm các FK có dính dáng đến các Snapshot được tách thành 4 FK.

c.Tiến hành:

Sử dụng Designer/2000 để thực hiện việc tạo các DDL scripts.

-Tạo hai Application đại diện cho trung tâm và chi nhánh (CENTER và BRANCH), và tạo một Application tên là ALL_OBJECTS phục vụ cho mục đích tạo ra các FK của tất cả các bảng trong hệ thống một cách tổng thể.

-Vào các ứng dụng WBL, WCA, WMA và WMI để share tất cả các bảng cho hai ứng dụng này một cách tương ứng (bảng nào có nguồn gốc trung tâm thì đưa vào ứng dụng CENTER và ngược lại bảng nào có nguồn gốc chi nhánh thì đưa vào ứng dụng BRANCH.

Share tất cả các bảng và sequence của 4 ứng dụng WBL, WCA, WMA, WMI cho ứng dụng ALL_OBJECTS.

-Tại CENTER tạo tất cả các snapshot cần thiết tham chiếu đến các bảng của các chi nhánh theo cách đặt tên ở trên, sau đó tạo các Views tương ứng tổng hợp thông tin từ các Snapshot.

-Tại BRANCH tạo tất cả các Snapshot tham chiếu đến các bảng của CENTER, cần để ý đến điều kiện chọn lọc trong một số bảng có liên quan đến chi nhánh.

Để tạo điều kiện chọn lọc khả thi, có một số cách như sau:

Bảng BRANCH tại chi nhánh chỉ chứa thông tin của riêng chi nhánh đó. Tuy nhiên một số module (mặc dù hoạt động tại chi nhánh) cần biết đến tất cả các chi nhánh khác.

Bảng BRANCH chứa thông tin về tất cả các chi nhánh và có một bảngriêngcùngcấu trúc với BRANCH chứa thông tin về chi nhánh hiện tại..

Bảng BRANCH chứa thông tin về tất cả các chi nhánh và trong bảng WSC_PARAMETERS có chứa một bản ghi chỉ ra mã của chi nhánh hiện thời..

Ta sẽ chọn cách thứ 3 để thực hiện.

Ví dụ với chi nhánh Chợ Lớn, ta thực hiện lệnh sau:

```
INSERT INTO WSC_PARAMETERS (NAME, VALUE, DESCRIPTION)
VALUES('BRANCH_CODE', 'CL', 'Mã của chi nhánh hiện tại');
```

Khi đó điều kiện để lấy bảng BNR như sau:

```
CREATE SNAPSHOT BNR
  AS SELECT * FROM BNR@wsc
  WHERE BRANCH_ID IN SELECT BRANCH.ID
    FROM BRANCH, WSC_PARAMETERS
    WHERE BRANCH.BRANCH_CODE = WSC_PARAMETERS.VALUE
      AND WSC_PARAMETERS.NAME = 'BRANCH_CODE' ;
```

Lần lượt vào 3 ứng dụng CENTER, BRANCH, và ALL_OBJECTS để thực hiện việc tạo ra các DDL script.

IV/Phân tán dữ liệu về khách hàng trong WSC:

Vì WSC là một hệ thống khá lớn, nên trong phạm vi của luận án xin chỉ trình bày chi tiết cách thức thực hiện phân tán một phần dữ liệu đầy đủ về Khách Hàng trong hệ thống WSC.

1/Giới thiệu các thực thể trong ứng dụng quản lí Khách Hàng:

+Khách Hàng (*CUSTOMER*) :

Hệ thống lưu các thông tin về khách hàng:

CUSTOMER_N0: Là mã số duy nhất ứng với mỗi khách hàng gọi là mã danh bộ, mã này được thiết lập dựa trên địa danh phố *(STREET)*, phường *(WARD)*, tiểu khu *(SUBAREA)* và quận *(DISTRICT)*.

*CONTRACT_N0:*Mã của hợp đồng được kí kết giữa khách hàng và công ty về sử dụng nước.

CONTRACT_DATE: Ngày kí hợp đồng.

NAME:Tên khách hàng.

ADDRESS: Địa chỉ của khách hàng.

INST_ADDRESS: Địa chỉ cài đặt đồng hồ.

PAYMENT_TYPE: Kiểu thanh toán tiền của khách hàng.

Một số các thuộc tính khác là: *TELFAX, IDCARD_N0, SORT_NAME, STATUS,HOUSE_NUM, PERSON_NUM, BANK_ACCOUNT, BANK_OWNER, BANK_NAME, TOTAL_QUOTA, USAGE_LIST, WSC_SIGNED* .

Ngoài ra ứng với mỗi khách hàng chính còn có thể có các khách hàng phụ (*SUB_CUSTOMER*) được phân biệt bởi *SUB_CUST_ID* (nếu là khách hàng chính thì thuộc tính này nhận giá trị *NULL*). Với mỗi loại khách hàng thuộc một trong các loại: Tư gia,Tập thể,Cơ quan, Người nước ngoài. Khách hàng có thể sử dụng nước cho nhiều mục đích khác nhau như dùng cho sinh hoạt và sản xuất và mỗi một đích có một giá biểu riêng. Một khách hàng thuộc một đợt tính hoá đơn/thu tiền/đọc số nhất định. Đồng hồ lắp đặt được lưu đầy đủ các thông số kỹ thuật và vị trí vật lý.

Một số các thực thể khác của ứng dụng quản lý Khách Hàng:

+STREET: Phố

+WARD: Phường

+SUBREA: Tiểu khu

+DISTRICT: Quận

+CUSTOMER CATEGORY: Loại khách hàng

+ACTIVITY: Hoạt động

+BRANCH: Chi nhánh

+ACTIVITY_ASSIGN

+CURRENCY: Tiền tệ

+CUST NOTE: Các chú thích về khách hàng

+AUDITTRAIL

+SUB_CUSTOMER: Khách hàng phụ

+USAGE ASSIGN: Kiểu sử dụng

+METER_INSTALLATION: Cài đặt đồng hồ

+SERVICE ORDER: Danh sách các dịch vụ

+SUPPORT_CALL:Đăng kí khách hàng

+COMPAINT : Yêu cầu khiếu nại của khách hàng

Hiện nay tổng số khách hàng của công ty là 500,000 khách hàng.

2/Thực hiện phân tán CSDL.

Tại trung tâm (Center) có các bảng dữ liệu như trong kết quả của câu lệnh SELECT dưới đây:

SQL> select * from tab where tabtype='TABLE';

TNAME	TABTYPE	CLUSTERID
ACTIVITY	TABLE	
BRANCH	TABLE	
CCATS	TABLE	
COMPLAINTS	TABLE	
CURRENCY	TABLE	
WSC_PARAMETERS	TABLE	
WSC_USERS	TABLE	

Các chi nhánh Sài Gòn, Gia Định, Chợ Lớn, Thủ Đức có các bảng dữ liệu: ACTIVITY_ASSIGN, AUDIT_TRAILS ...

SQL> select * from tab where tabtype='TABLE';

TNAME	TABTYPE	CLUSTERID
ACTIVITY_ASSIGN	TABLE	
AUDIT_TRAILS	TABLE	
CUSTOMERS	TABLE	
CUST_NOTE	TABLE	
DISTRICTS	TABLE	
STREETS	TABLE	
SUBAREA	TABLE	
SUB_CUSTOMERS	TABLE	
SUPPORT_CALL	TABLE	
USAGE_ASSIGN	TABLE	
WARDS	TABLE	
WSC_PARAMETERS	TABLE	
WSC_USERS	TABLE	

Tại trung tâm có các Snapshot của các table tại các chi nhánh, và tại các chi nhánh cũng có Snapshot của dữ liệu trung tâm.

Script thực hiện việc tạo Database link để kết nối với CSDL từ xa DBlink.sql. Kết quả là 8 Database link (4 Database link là Center tại 4 chi nhánh và các Database link: SaiGon, GiaĐinh, ThuĐưc, ChoLon tại Center) được tạo ra:

```
REM                Create database links
REM      CENTER
REM      CENTER
REM      CENTER
REM      CENTER
REM
REM      SAIGON
REM      GIADINH
REM      CHOLON
REM      THUDUC
REM

PROMPT Create database link CENTER for SAIGON branch

        CONNECT SAIGON/SAIGON@STU;
        Create database link CENTER connect to CENTER identified by
CENTER
        using 'VPCP'
    ;
    PROMPT Create database link CENTER for GIADINH branch

        CONNECT GIADINH/GIADINH@STU;
        Create database link CENTER connect to CENTER identified by
CENTER
        using 'VPCP'
    ;
    PROMPT Create database link CENTER for CHOLON branch

        CONNECT CHOLON/CHOLON@STU;
        Create database link CENTER connect to CENTER identified by
CENTER
        using 'VPCP'
    ;
    PROMPT Create database link CENTER for THUDUC branch

        CONNECT THUDUC/THUDUC@STU;
```

```
        Create database link CENTER connect to CENTER identified by
CENTER
        using 'VPCP'
    ;
    PROMPT Create database link SAIGON for CENTER site

        CONNECT CENTER/CENTER@VPCP;
        Create database link SAIGON connect to SAIGON identified by
SAIGON
        using 'STU'
    ;

    PROMPT Create database link GIADINH for CENTER site

        CONNECT CENTER/CENTER@VPCP;
        Create database link GIADINH connect to GIADINH identified
by GIADINH
        using 'STU'
    ;
    PROMPT Create database link CHOLON for CENTER site

        CONNECT CENTER/CENTER@VPCP;
        Create database link CHOLON connect to CHOLON identified
by CHOLON
        using 'STU'
    ;
    PROMPT Create database link THUDUC for CENTER site

        CONNECT CENTER/CENTER@VPCP;
        Create database link THUDUC connect to THUDUC identified
by THUDUC
        using 'STU'
    ;
    REM Database links created
```

Một phần Script thực hiện việc tạo các Snapshot tại vị trí trung tâm Sna_cen.sql như sau:

```
    REM                    Create snapshots from BRANCHS
    PROMPT              Create snapshots from BRANCHS
    REM
```

```
REM
REM                     Create snapshots from GIA DINH
PROMPT          Create snapshots from GIA DINH
REM GIADINH
REM ACTIVITY_ASSIGN$GD
REM AUDIT_TRAILS$GD
REM CUSTOMERS$GD
REM CUST_NOTE$GD
REM DISTRICTS$GD
REM STREETS$GD
REM SUBAREA$GD
REM SUB_CUSTOMERS$GD
REM SUPPORT_CALL$GD
REM USAGE_ASSIGN$GD
REM WARDS$GD
REM WSC_PARAMETERS$GD
REM WSC_USERS$GD
REM
REM
REM
REM
REM                 Create snapshots from THU DUC
PROMPT                  Create snapshots from THU DUC
REM THUDUC
REM ACTIVITY_ASSIGN$TD
REM AUDIT_TRAILS$TD
REM CUSTOMERS$TD
REM CUST_NOTE$TD
REM DISTRICTS$TD
REM STREETS$TD
REM SUBAREA$TD
REM SUB_CUSTOMERS$TD
REM SUPPORT_CALL$TD
REM USAGE_ASSIGN$TD
REM WARDS$TD
REM WSC_PARAMETERS$TD
REM WSC_USERS$TD
REM
                CONNECT CENTER/CENTER@VPCP;
REM
PROMPT Create snapshot ACTIVITY_ASSIGN$GD
```

```
        Create snapshot ACTIVITY_ASSIGN$GD
         as select * from ACTIVITY_ASSIGN@GIADINH
;
REM
PROMPT CREATE SNAPSHOT AUDIT_TRAILS$GD

        Create snapshot AUDIT_TRAILS$GD
         as select * from AUDIT_TRAILS@GIADINH
;
REM
PROMPT CREATE SNAPSHOT CUSTOMERS$GD

        Create snapshot CUSTOMERS$GD
         as select * from CUSTOMERS@GIADINH
;
REM
PROMPT CREATE SNAPSHOT CUST_NOTE$GD

        Create snapshot CUST_NOTE$GD
         as select * from CUST_NOTE@GIADINH
;
REM
PROMPT CREATE SNAPSHOT DISTRICTS$GD

        Create snapshot DISTRICTS$GD
         as select * from DISTRICTS@GIADINH
;
REM
PROMPT CREATE SNAPSHOT STREETS$GD

        Create snapshot STREETS$GD
         as select * from STREETS@GIADINH
;
REM
PROMPT CREATE SNAPSHOT SUBAREA$GD

        create snapshot SUBAREA$GD
         as select * from SUBAREA@GIADINH
;
REM
```

```
PROMPT CREATE SNAPSHOT SUB_CUSTOMERS$GD

      create snapshot SUB_CUSTOMERS$GD
       as select * from SUB_CUSTOMERS@GIADINH
;
REM
PROMPT CREATE SNAPSHOT SUPPORT_CALL$GD

      create snapshot SUPPORT_CALL$GD
       as select * from SUPPORT_CALL@GIADINH
;
REM
PROMPT CREATE SNAPSHOT USAGE_ASSIGN$GD

      create snapshot USAGE_ASSIGN$GD
       as select * from USAGE_ASSIGN@GIADINH
;
REM
PROMPT CREATE SNAPSHOT WARDS$GD

      create snapshot WARDS$GD
       as select * from WARDS$GD@GIADINH
;
REM
PROMPT CREATE SNAPSHOT WSC_PARAMETERS$GD

      create snapshot  WSC_PARAMETERS$GD
       as select * from  WSC_PARAMETERS@GIADINH
;
REM
PROMPT CREATE SNAPSHOT WSC_USERS$GD

      create snapshot WSC_USERS$GD
       as select * from WSC_USERS@GIADINH
;
REM
REM
REM
```

Sau khi Script Sna_cen.sql thực hiện tại Center ngoài các bảng dữ liệu cũ còn có thêm một tập các Snapshot từ các chi nhánh. Bảng TAB1 trong phần phụ lục mô tả toàn bộ CSDL của Center.

Tương tự như vậy các Script Sna_SG.sql, Sna_GD.sql, Sna_CL.sql, Sna_TD.sql sẽ tạo các Snapshot từ vị trí Center tới các chi nhánh tương ứng.

Trước khi trình bày các Script dùng để thực hiện làm tươi các Snapshot, xin giới thiệu Script: Log_cen.sql là một trong 5 Script thực hiện việc tạo các Snapshot log tại vị trí chủ.

```
REM                    This is log_cen.sql script
REM                    Create snapshot log at CENTER site
PROMPT          Create snapshot log at CENTER site
REM
REM
REM  ACTIVITY
REM  BRANCH
REM  CCATS
REM  COMPLAINTS
REM  CURRENCY
REM
REM
      CONNECT center/center@vpcp;
REM

      Create snapshot log on ACTIVITY;
      Create snapshot log on BRANCH;
      Create snapshot log on CCATS;
      Create snapshot log on COMPLAINTS;
      Create snapshot log on CURRENCY;

PROMPT snapshots created
```