|  |  |  |
| --- | --- | --- |
| Từ/Cụm từ viết tắt | Nội dung đầy đủ | Ghi chú khác |
| EF | Entity Framework |  |
| CSDL | Cơ sở dữ liệu |  |
| MVP | Model-View-Presenter |  |
| ORM | Object Relational Mapping |  |
| DB | Database |  |
| LINQ | Language Integrated Query |  |

I. ADO.NET Entity Framework

\*Definition:

-Là một framework giúp ánh xạ cơ sở dữ liệu quan hệ lên một khung nhìn các đối tượng và tập hợp các đối tượng.

-Là một DB ORM (Database Object-Relational Mapping) mã nguồn mở được chính Microsoft phát triển dựa trên nền tảng .NET.

-EF giúp người lập trình tiết kiệm được rất nhiều thời gian và công sức trong việc truy xuất và làm việc với các CSDL quan hệ, bởi vì bản thân EF đã cung cấp đầy đủ các tính năng và công nghệ tiên tiến giúp độc lập hóa khối CSDL ra khỏi lớp truy xuất.

\*Version: 6.1.1

\*Các tính năng được vận dụng trong đồ án:

- LINQ to Entities: EF sử dụng LINQ để truy vấn trên CSDL. LINQ là một công nghệ truy vấn CSDL không phụ thuộc vào dữ liệu nguồn, tức là có thể dùng LINQ để truy vấn và thực thi các thao tác dữ liệu trên: tập hợp/danh sách các đối tượng, tập tin XML hay các hệ quản trị CSDL khác nhau.

-LINQ IEnumberable và biểu thức chính quy Lambda

+LINQ IEnumberable: là một lớp giao diện (interface) định nghĩa trong LINQ, cho phép tạo ra các câu truy vấn lồng nhau trên một tập hợp có thực thi lớp giao diện này. Kết quả của một loạt các thực thi sẽ được trả về ngay sau khi có lời gọi để chuyển từ tập hợp không chính quy (non-generic) sang tập hợp chính quy (generic). Đây là tính năng rất hay trong LINQ, mà nhờ đó tiết kiệm được chi phí cũng như thời gian thực thi dữ liệu, vì câu truy vấn cuối cùng chỉ được kết lại và gọi chạy khi hoàn tất một loạt các truy vấn lồng nhau thay vì phải tải dữ liệu lên sau mỗi bước nạp điều kiện.

+Lambda là một dạng biểu thức chính quy được sử dụng trong các truy vấn đến các lớp dữ liệu có đăng ký thực thi lớp IEnumberable.

[Hình minh họa IEnumberable và Lambda]



-Lazy loading: Trì hoãn load dữ liệu khi chưa cần thiết, dữ liệu chỉ được load khi có lời gọi tới lần đầu tiên (Giảm thời gian nạp dữ liệu ban đầu). Lazy loading rất hữu ích trong trường hợp một các đối tượng có mối quan hệ phức tạp qua lại với nhau và cây liên hệ có chiều cao lớn hơn 1. Mặc định EF được cấu hình bật Lazy loading.



-Code first: là một cách tiếp cận mới trong các ứng dụng hướng CSDL, thay vì thiết kế cấu trúc CSDL trước thì Code first cung cấp các đặc tả để người lập trình có thể định nghĩa cấu trúc CSDL bằng các class và các logic, ràng buộc trên class; tận dụng được các tính năng kế thừa trên class, sau đó Entity Framework sẽ tự động ánh xạ bản thiết kế xuống cấu trúc CSDL.

-Tích Đề-các (Descartes) trong biểu diễn quan hệ 1-n hoặc n-n:

Gọi tập hợp A = {Table A0, Table A1,...Table Am} chứa các thực thể ở quan hệ nhiều

Gọi tập hợp B = {Table B0, Table B1, ... Table Bn} chứa các thực thể ở quan hệ 1

Một tích Đề-các diễn tả mối quan hệ 1-n hoặc n-n từ Bi đến Aj (i < [B], j < [A])

B x A { (b,a) | b thuộc B, a thuộc A }

-Mối quan hệ 1-n trong CSDL quan hệ có thể được biểu diễn bằng 2 cách sau:

+Cách 1: Nâng cấp quan hệ 1-n thành quan hệ n-n:

Khi đó mỗi quan hệ (B x A)i ( i < [B x A] ) sẽ liên kết với (1+1)+1=3 Table vật lý

Tổng Table vật lý tối thiểu cần thiết để biểu diễn quan hệ trên (AxB) là:

[A] + [B] + [B x A]

+Cách 2: Truyền thống, đối tượng ở quan hệ nhiều sẽ có n khóa ngoại trỏ đến đối tượng ở quan hệ 1

Khi đó mỗi quan hệ (B x A)i (i < [B x A]) sẽ liên kết ứng với 1+1=2 Table vật lý

Tổng Table vật lý tối thiểu cần thiết để biểu diễn quan hệ trên (B x A) là:

[A] + [B]

\*EF làm việc được với cả 2 cách biểu diễn trên.

\*Dễ thấy rằng cách 1 sẽ tiêu tốn nhiều không gian lưu trữ hơn, bù lại tận dụng được nhiều tính năng ưu việt hơn do kỹ thuật ánh xạ của EF.

-Mối quan hệ n-n trong CSDL quan hệ có thể được biểu diễn bằng 2 cách:

+Cách 1: tương tự cách 1 trong biểu diễn quan hệ 1-n (nhưng vì đã là quan hệ n-n nên không cần nâng cấp).

+Cách 2: Sử dụng thuộc tính đa trị trong từng trường định nghĩa khóa ngoại, tuy nhiên sẽ vi phạm dạng chuẩn CSDL 1NF (dạng chuẩn thấp nhất) do chứa thuộc tính đa trị, và không được EF hỗ trợ, nên không khả thi khi triển khai.

\*Đồ án sử dụng cách 1 trong việc biểu diễn quan hệ n-n giữa: Cơ sở, Dãy, Tầng, Phòng,... và Hình Ảnh

+Sử dụng nhiều trong các ứng dụng dạng Domain-Driven Design, cho phép người lập trình ảo hóa CSDL vật lý, giảm sự phục thuộc vào CSDL vật lý (các vấn đề về tính tương thích trên các phiên bản Hệ quản trị).

+Code first to new Database

+Phiên bản CSDL (Database Version): (Upgrade/Downgrade/Rebase): EF xem mỗi sự thay đổi trong bản thiết kế lớp (class) sẽ tương ứng với một phiên bản CSDL mức vật lý, khi bản thiết kế có sự thay đổi, EF sẽ ghi nhận lại sự thay đổi đó và tạo ra các đoạn mã để điều chỉnh cấu trúc CSDL hiện tại lại cho khớp với bản thiết kế mới. Những đoạn mã này được gọi là một phiên bản của CSDL, và sẽ được định danh bằng tên mã nhằm phục vụ cho quá trình dịch chuyển phiên bản.

+Dịch chuyển phiên bản CSDL (Migration to database Version):

* Nâng cấp (Upgrade): là hình thức dịch chuyển từ phiên bản thấp Vi lên phiên bản cao hơn Vj (i<j)
* Hạ cấp (Downgrade): là hình thức di chuyển từ phiên bản cao Vj xuống phiên bản thấp hơn Vi (i<j)
* EF cho phép nhảy cốc giữa các phiên bản, ví dụ có thể nhảy từ phiên bản v3 xuống v1 mà không cần thông qua v2 và ngược lại, miễn là các v1, v2, v3 đã được định danh trước đó. Tính năng này rất hữu ích khi dự án được thực hiện ở quy mô nhóm, khi đó các lập trình viên sẽ tiết kiệm được rất nhiều thời gian cho khâu cập nhật CSDL trong suốt quá trình lập trình và kiểm thử.

+Sự tương thích giữa các phiên bản CSDL: đây cũng là vẫn đề được quan tâm tới trong khi thiết kế và vận hành ứng dụng. EF chỉ có thể làm việc trên phiên bản CSDL Vi nếu Vi tương thích với phiên bản CSDL hiện tại (ứng với thiết kế lớp (class) hiện tại). Tùy thuộc vào sự thay đổi giữa các phiên bản mà có thể tương thích hoặc không tương thích cụ thể như sau:

\*Xét 2 phiên bản CSDL Vi và Vj (i<j):

* Tương thích ngược: Vj được xem là tương thích ngược với Vi nếu CSDL Vj bao trùm CSDL Vi.



* Tương thích xuôi: Vi được xem là tương thích xuôi với Vj nếu CSDL Vi bao trùm CSDL Vj (điều này rất hiếm khi xảy ra vì đa phần bản thiết kế mới Vj luôn mở rộng hơn so với bản cũ Vi).



* Không tương thích: Trong các trường hợp còn lại.





\*Các ví dụ minh họa trên chỉ thể hiện ở mức đơn giản nhất, trên thực tế việc xét tính tương thích sẽ phức tạp hơn nhiều do sự kết hợp của nhiều bảng và định nghĩa các thuộc tính trên bảng.

+Data Seeding: Cung cấp các đặc tả để tạo dữ liệu mặc định ban đầu khi tạo mới CSDL, ví dụ: Tài khoản quản trị mặc định, các giá trị cài đặt mặc định,...

+Model cheking (Structure changed): Kiểm tra cấu trúc CSDL có bị thay đổi bởi các tác nhân ngoài hệ thống, đảm bảo hệ thống làm việc ổn định và dúng đắn. Model cheking là cực kỳ quan trọng khi làm việc với EF bởi vì các trường thuộc tính của dữ liệu vật lý gắn chặt với các class tương ứng của ứng dụng khi bộ máy ánh xạ hoạt động, nên chỉ với một thay đổi nhỏ về định nghĩa của CSDL vật lý cũng sẽ khiến EF không hoạt động.

+Trình khởi tạo CSDL tùy biến (Custom Database Initalizer): Chỉ định phương thức khởi tạo CSDL, EF cho phép lựa chọn và tùy biến nhiều chỗ trong quá trình tạo tự động cấu trúc CSDL khi ánh xạ bản thiết kế xuống CSDL mức vật lý, các tính năng bao gồm: tự động tạo CSDL nếu chưa có ?, tự động tạo bảng nếu chưa có ?, tự động tạo dữ liệu mẫu ban đầu ?,…)

a. CreateDatabaseIfNotExists: mặc định của EF. Tự động tạo CSDL nếu chưa có.

b. DropCreateDatabaseIfModelChanges: Tự động tạo lại CSDL khi cấu trúc bị thay đổi. Tuy nhiên không an toàn vì chỉ một sơ suất nhỏ trong khâu thiết kế cũng sẽ dẫn đến nguy cơ mất CSDL.

c. DropCreateDatabaseAlways: Luôn luôn xóa và tạo mới lại CSDL mỗi khi ứng dụng khởi chạy, phù hợp với các ứng dụng sử dụng CSDL như là một bộ nhớ tạm trong lúc làm việc, và muốn CSDL rỗng cho mỗi phiên mới.

d. Custom DB Initializer: Nếu các định nghĩa trên không đáp ứng được yêu cầu thì người lập trình có thể tự định nghĩa một trình khởi tạo dữ liệu riêng cho CSDL.

-BootStrapper (override some method for specific business process) vs Event Hooker: Cho phép đăng ký các sự kiện xảy ra trên Entity khi Entity Framework làm việc, sử dụng để thực hiện một vài hành động logic như: sau khi cập nhật thành công thì trường date\_modified sẽ có giá trị giờ hiện tại.

-Class proxy (override defined class): EF đòi hỏi thuộc tính khóa ngoại của các class phải được khai báo dạng virtual để EF override các kỹ thuật như Lazy loading lúc truy xuất các thuộc tính này.

-Transaction Manager (Rollback/commit): Đảm bảo tính đơn nguyên của giao dịch, hoặc là tất cả thao tác dữ liệu đều được thực thi hoặc là không. Nhằm đảm bảo tính nhất quán trên CSDL. EF cung cấp một cơ chế Transaction rất đơn giản và hiệu quả.

-Entity State tracking and "Filter Update": Để có thể theo dõi được các sự thay đổi dữ liệu trên các Entity trong môi trường runtime, EF đưa ra định nghĩa về trạng thái của các Entity, trong đó một Entity có thể thuộc một trong các trạng thái sau:

a. Attached (Entity mới khoải tạo và được đưa và hệ thống tracking, tuy nhiên chưa được lưu xuống CSDL)

b. Detached (Entity đã bị loại khi hệ thống tracking)

c. Added (Entity được đưa vào hàng đợi chờ thêm vào CSDL)

d. Modified (Enity được đánh dấu là đã bị thay đổi ít nhất 1 thuộc tính được định nghĩa trong cấu trúc CSDL, được đưa vào hàng đợi chờ cập nhật xuống CSDL)

e. Unchange (Entity được đánh dấu là sạch, có thể là mới được khởi tạo hoặc là mới được load lên từ CSDL)

f. Deleted (Entity được đánh dấu là bị xóa, được đưa vào hàng đợi chờ xóa khởi CSDL)

-Entity framework kết hợp mô hình 3 lớp

-Manual media cached (class HinhAnh):

Với các ứng dụng hướng đa phương tiện thì việc tổ chứa lưu trữ và truy xuất các tài nguyên là quan trọng nhằm tiết kiệm bộ nhớ và thời gian truy xuất.

-Entity vs DataFilter:

+Entity là một thực thể mang các thuộc tính trực tiếp của bản thân nó, trong lập trình giao diện, thông thường khi hiển thị thông tin một đối tượng của một thực thể nào đó, ta thường hiển thị các thuộc tính gián tiếp (thuộc tính của khóa ngoại), do đó phần xử lý giao diện sẽ không làm việc trực tiếp với Entity mà làm việc thông qua một lớp mặt nạ gọi là DataFilter, nhiệm vụ của DataFilter là kết các Entity có liên quan lại với nhau sau đó chọn ra các thuộc tính cần hiển thị.

-Database Context vs Singleton Database Instance Provider:

+ EF xem Database Context là một ngữ cảnh truy xuất đến CSDL, trên đó chứa các lớp mặt nạ giúp truy xuất tài nguyên CSDL.

+Singleton giúp các lớp Entity nhìn thấy cùng một DbContext trong suốt phiên làm việc, bởi vì EF đòi hỏi các đối tượng sinh ra từ các class Entity phải thống nhất về DbContext, một đối tượng không thể được tracking bởi 2 DbContext khác nhau.

+Phiên làm việc được đánh dấu từ lúc DbContext được khởi tạo cho đến khi bị hủy bỏ (Dispose), các đối tượng có runtime nằm ngoài phiên làm việc được xem là không hợp lệ và không có ý nghĩa về mặt dữ liệu, muốn làm việc lại trên các đối tượng này nhất thiết phải load lại trong một phiên làm việc khác của DbContext.

-TPC (Table per Concreted class) pattern: tận dụng code logic bằng cách thiết kế mô hình kế thừa, đa hình. Code logic chung sẽ được đặt ở lớp trừu tượng.

+Trong TPC, các lớp ảo (abstract) sẽ không được ánh xạ xuống CSDL vật lý, các lớp này chỉ dùng cho mục đích kế thừa và đa hình. Muốn ánh xạ một Entity xuống CSDL vậy lý thì lớp Entity đó bắt buộc không được khai báo là một lớp ảo

-OOP:

+Interface: Định nghĩa các lớp giao diện phục vụ cho kế thừa và Entity Bootstrap

+Kế thừa:

+Đa hình (override) ở các lớp dẫn xuất để thay đổi hành vi ứng với các Rntity khác nhau.

\*References source:

<http://www.entityframeworktutorial.net/>

<http://msdn.microsoft.com/en-us/data/aa937723>

<https://github.com/tidyui/EFBootstrapper>

II. Sync Framework

-Định nghĩa: Sync Framework là công nghệ được Microsoft phát triển với mục đích chính là đồng bộ dữ liệu qua lại giữa các nguồn dữ liệu, dữ liệu nguồn ở đây có thể là hệ thống tập tin hoặc là một CSDL của một hệ quản trị nào đó, hiện Sync Framework hỗ trợ các dữ liệu nguồn tương thích chuẩn ADO.NET mà trong đó hệ quản trị MSSQL Server hoàn toàn đáp ứng được các yêu cầu trên.

-Phiên bản mới nhất hiện tại: 2.1

-Phiên bản được áp dụng trong phần mềm: 2.1

-Trong phạm vi ứng dụng của đề tài này, sẽ chỉ xem xét đến nguồn dữ liệu là hệ quản trị CSDL, cụ thể là MSSQL Server.

-Các tính năng được vận dụng trong đồ án:

+Đồng bộ dữ liệu giữa các CSDL với mức đơn vị dữ liệu là bảng.

+Sử dụng kỹ thuật trigger trên từng bảng, trigger có nhiệm vụ thu thập và bắt các sự thay đổi về dữ liệu trên CSDL, sau đó lưu trữ lại trong các bảng theo dõi (tracking table, mỗi bảng được chỉ định trong Sync Scope sẽ phát sinh ra một bảng theo dõi tương ứng) mà Sync Framework tạo ra khi một Sync Scope được cài đặt.

+Nhờ các kỹ thuật lưu trữ và theo dõi dữ liệu như đã được giới thiệu ở trên mà Sync Framework sẽ chỉ đồng bộ những dữ liệu sai khác giữa các dữ liệu nguồn, do đó quá trình xử lý và truyền nhận dữ liệu sẽ tiết kiệm được thời gian và tài nguyên hệ thống, khác với các giải pháp truyền thống là phải tải mới toàn bộ dữ liệu.

-Không gian đồng bộ (Sync Scope)

+Định nghĩa: Sync Scope là được hiểu như là định nghĩa về một phiên đồng bộ trên một CSDL cụ thể, chứa các thông tin về tập hợp bảng cần đồng bộ. Một CSDL có thể có nhiều hơn một Sync Scope. Một CSDL có thể có nhiều Sync Scope được thiết lập sẵn thông qua việc mở rộng vùng lưu trữ trên CSDL bằng các bảng tạm và các procedure chức năng

+Cài đặt một Sync Scope lên CSDL có sẵn: là một loạt các thao tác cần thiết để khởi tạo và định danh một Sync Scope lên trên CSDL đã có sẵn dữ liệu hoặc CSDL mới hoàn toàn, trong đó việc chỉ định danh sách các bảng cần đồng bộ được xem là quan trọng nhất, khi các phương thức khởi tạo Sync Scope được gọi, Sync Framework sẽ thực hiện nhiệm vụ còn lại.

+Gõ bỏ một Sync Scope có sẵn ra khỏi CSDL: thao tác gỡ bỏ Sync Scope sẽ ngược lại với các bước khi cài đặt trước đó. Chỉ cần chỉ định dữ liệu nguồn và tên Sync Scope, sau đó gọi phương thức gỡ bỏ, Sync Framework sẽ thực hiện nhiệm vụ còn lại. Việc gõ bỏ Sync Scope sẽ đồng nghĩa với việc CSDL sẽ không thể cung cấp phiên đồng bộ cho các trình quản lý đồng bộ, và do đó sẽ không tham gia vào hệ thống.

+Áp đặt Sync Scope giữa các CSDL (Fetching scope among databases): là sao chép các thông tin về các định nghĩa phiên, các cấu hình liên quan,... từ một Sync Scope trên CSDL A có sẵn sang một Sync Scope mới trên CSDL mới B, khai báo rằng CSDL A và B có thể "bắt tay" được với nhau, lúc này trình quản lý đồng bộ mới có thể nhìn thấy và làm việc được trên cả 2 CSDL này, cũng có thể nói đây là bước thiết lập một cầu nối (pipeline) dữ liệu. Việc áp đặt Sync Scope từ một CSDL này lên một CSDL là rất quan trọng. Trong một tiến trình đồng bộ cụ thể thì thao tác này được thiết lập sớm nhất ngay khi có thể. Một CSDL có thể bắt tay với nhiều CSDL khác, đây là tính năng sẽ được ứng dụng trong các mô hình triển khai máy trạm khi đưa vào vận hành.

+Hướng đồng bộ lên/xuống/2 chiều, tính thông nhau giữa các cầu nối (Sync Direction (Up/Down/Bidirectional link)): khi một cầu nối được thiết lập giữa 2 CSDL thì việc quy định hướng của luồng đồng bộ được xem xét và quyết định tùy thuộc vào yêu cầu về chức năng và quyền hạn của các máy trạm. Ví dụ: trong hệ thống có những máy trạm chỉ muốn sao chép các thay đổi từ máy chủ tập trung xuống để xem và thống kê mà không có các thao tác thay đổi (read only) thì giải pháp Download only được áp dụng. Việc chọn giải pháp phù hợp sẽ làm tăng tính an toàn dữ liệu vì việc cập nhật sửa đổi đã được kiểm soát ở mức CSDL thấp hơn so với mức ứng dụng (Application), tính bảo mật dữ liệu cũng được tăng lên.

\*Challenge:

-Đụng độ dữ liệu:

+Đụng độ vật lý trên khóa chính: Xét 2 CSDL độc lập có cùng cấu trúc Table và dữ liệu, nếu khóa chính được thiết lập dạng Auto\_Increasement thì khi gọi Insert trên 2 CSDL, hệ quản trị CSDL địa phương ở cả 2 CSDL trên sẽ có khả năng tạo ra khóa chính trùng nhau (khả năng rất cao). Nên khi đồng bộ dữ liệu sẽ bị đụng độ khóa chính, 1 trong 2 dữ liệu mới Insert đó phải bị loại bỏ.

=> Sử dụng khóa chính Guid: Khóa chính Guid là khóa chính dạng "tự nhiên" (Natural), được tạo ra dựa trên các giá trị định danh (địa chỉ MAC của card mạng) và ngẫu nhiên (thời gian hiện tại trên Hệ quản trị CSDL), Hệ quản trị sẽ đảm bảo mỗi Guid được cấp phát sẽ là duy nhất trên toàn cầu (mặc dù khả năng trùng là có thể xảy ra trên lý thuyết nhưng có thể chấp nhận được).

+Đụng độ logic do sự trễ (Delay) dữ liệu: một sự trễ dữ liệu được định nghĩa khi mà sự thay đổi về mặt CSDL ở một máy trạm khác (dù đã đẩy hay chưa đẩy lên máy chủ tập trung) chưa kịp cập nhật cho máy trạm địa phương (local machine) mà máy trạm địa phương cũng đã tạo nên một sự sửa đổi. Khi mà đụng độ về khóa chính (đụng độ vật lý) đã được giải quyết thì đụng độ logic do sự chậm trễ trong việc cập nhật dữ liệu lên máy chủ tập trung là vấn đề không thể tránh khỏi do đặc thù làm việc Offline, và Sync Framework chỉ có thể can thiệp và xử lý tự động khi hạng mục dữ liệu bị thay đổi ở cả 2 nguồn là không giao nhau hoặc hạng mục dữ liệu giao nhau là đơn nhất. Đối với các trường hợp sự thay đổi ở một hạng mục kéo theo sự thay đổi ở các hạng mục liên quan thì Sync Framework không thể can thiệp được - đây cũng là cách làm việc chung cho các hệ đồng bộ hiện nay. Việc tránh đụng độ kiểu này phải do lập trình viên tự quy định các chính sách về đồng bộ dữ liệu giữa các máy trạm và máy chủ tập trung.

- Database Schema "dbo" cho User (SQL Server):

+Database Schema có ảnh hưởng đến phân quyền trên CSDL, các Table thuộc các Database Schema khác nhau sẽ rất khó để giao tiếp qua lại bằng các trigger cầu nối. Vì thế để tránh rắc rối phát sinh trong quá trình sử dụng, nhất là tác vụ đồng bộ (đã được ghi nhận không thể hoạt động ổn định trên tập CSDL được thiết lập bởi nhiều Database Schema), chúng tôi khuyến các người quản trị CSDL phải cấu hình sao cho tất cả các bảng trong CSDL đều được gắn kết bởi cùng một Database Schema thống nhất là "dbo"

-Đồ thị tiến trình đồng bộ: như đã đề cập ở phần trên, Sync Framework đồng bộ ở mức đơn vị dữ liệu là bảng, nên thứ tự các bảng trong một tiến trình đồng bộ là vô cùng quan trọng vì đặc thù ràng buộc khóa ngoại của CSDL quan. Nếu bảng B có chứa khóa ngoại tham chiếu đến bảng A thì bảng A phải được xử lý trước bảng B.

=> Cần phải xây dựng đồ thị tiến trình cho phiên đồng bộ. Do thứ tự bảng trước sau được xem xét nên đồ thị là đồ thị có hướng. Do trong CSDL quan hệ không được phép chứa có liên hệ vòng nên đồ thị là đơn đồ thị, không chứa chu trình (vì chu trình sẽ gây chết tiến trình đồng bộ (deadlock)) và không bao giờ là một đồ thị liên thông mạnh. Đồ thị có thể liên thông hoặc không liên thông tùy thuộc vào sự giao nhau giữa các tập quan hệ bảng. Đồ thị có thể có nhiều đồ thị con (các thành phần liên thông). Các thành phần liên thông có thể là các cây, trường hợp này cây có thể được tách ra thành nhiều cây con để xử lý song song bằng các tiến trình song song hoặc cũng có thể được xử lý tuần tự bằng một tiến trình duy nhất, tùy thuộc thiết kế đơn luồng hay đa luồng lúc thực thi.

Xem các bảng là các đỉnh (V - vertexes)

Xem các liên hệ khóa ngoại từ bảng này đến bảng kia là các cung (E - edges), trong đó một cung có nút con là bảng chứa khóa ngoại và nút cha là bảng tham chiếu đến

Ta được đồ thị có hướng G = {V, E}, trong đó:

Tập hợp các nút: V = {Table 0, Table 1, ...., Table n}

Tập hợp các cung: E = {e 0 = (V i, V j), e 1 = (V p, V q),...e m = (V t, V v)} (i,j,p,q,t,v < n)

Nguyên tắc xây dựng tiến trình đồng bộ song song:

+Mỗi đồ thị con Hi trong đồ thị G sẽ được xử lý bởi một tiến trình Pi. Nếu Hi là cây thì Hi sẽ được tách ra thành n cây khác nhau tại vị trí nút nhánh. và kéo theo sự phát sinh n tiến trình con W tương ứng (trong đó n là bậc phát của nút nhánh). Các cây con này sẽ tiếp tực được chia tách nếu vẫn còn nhánh.

+Thời gian sống của tiến trình con W phụ thuộc vào vòng đời sống của tiến trình cha P.

+Trong mỗi tiến trình Pi, ta dùng giải thuật sau để thiết lập hàng đợi đồng bộ Qi:

Bước 1: Khởi tạo hàng đợi Q có kích thước n phần tử, n là số đỉnh của đồ thị Hi tương ứng.

Bước 2: Thiết lập vị trí bắt đầu BD=0 và kết thúc KT=n-1

Bước 3: Lập danh sách các đỉnh phát, đỉnh thu của đồ thị Hi.

Bước 4: Đưa danh sách đỉnh phát vào đầu hàng đợi kể từ vị trí BD, không quan trọng thứ tự trước sau. Đưa danh sách đỉnh thu vào cuối hàng đợi từ vị trí KT trở về trước, không quan trọng thứ tự trước sau.

Bước 5: Xóa các đỉnh phát, đỉnh thu và các cung liên quan ra khỏi đồ thị Hi.

Bước 6: Thiết lập lại hàng đợi Q với:

BD = BD + [tập đỉnh phát]

KT = KT - [tập đỉnh thu]

Bước 7: Nếu hàng đợi Q rỗng (BD>=KT) thì kết thúc ngược lại thì quay lại bước 3.

+Nếu cây Ti có nhánh thì tiến hành cắt cây tại nút nhánh đó, lúc này tiến trình Pi tương ứng sẽ được chia ra thành w tiến trình Q song song (trong đó: w là bậc ngoài (số cung đi ra từ nút nhánh)), lặp lại cho các nhánh tiếp theo cho đến khi duyệt hết cây.

-Việc xây dựng tiến trình đồng bộ song song chỉ thực sự phát huy hiệu quả đối với những đồ thị có số lượng lớn các nút (cỡ khoảng trên 100 nút) và các cây (nếu có) có nhiều nhánh. Trong phạm vi đề tài này đồ thị có số lượng nút rơi vào khoảng 30-40 nút và các cây (nếu có) có ít nhánh nên việc xây dựng cây tiến trình song song sẽ không mang lại hiệu quả nhiều và rất phức tạp. Do đó phương pháp sử dụng một tiến trình duy nhất được xem xét tới.

Nguyên tắc xây dựng tiến trình đồng bộ đơn nhất:

+Chỉ có một tiến trình P duy nhất cho toàn đồ thị.

+Các nút trong từng đồ thị sẽ lần lượt được đưa vào danh sách hàng đợi Q sao cho mệnh đề sau luôn đúng:

"Với mọi Qi, Qj thuộc Q nếu cung e=(Qi, Qj) thuộc E thì j phải nhỏ hơn i"

(tức là nếu bảng A có chứa khóa ngoại đến bảng B thì bảng B phải được xử lý đồng bộ trước bảng A)

+Phương pháp này có thể được thực hiện bằng giải thuật sắp xếp trong đó điều kiện so sánh là xét cung tạo thành thuộc đồ thị hay không.

\*References source:

<http://msdn.microsoft.com/en-us/library/bb902854(v=sql.110).aspx>

III. DevExpress

\*Definition:

\*Version:

\*Features:

-MVP (Model-View-Presenter):

+Tận dụng các giao diện sử dụng chung

+Tận dụng các logic về giao diện chung

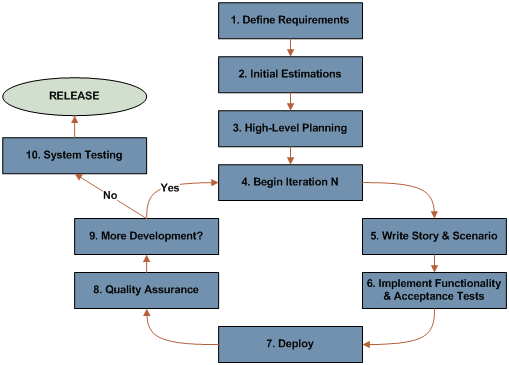
+Tạo sự thống nhất trong giao diện giữa các project GUI

-Custom User control:

+Coi như là một module MVP nhỏ sử dụng chung cho nhiều khu vực

IV. Mô hình phát triển phần mềm

-Nhóm áp dụng mô hình Agile trong toàn bộ quy trình phát triển phần mềm



[http://www.varsys.com/knowledgecenter\_WaterfallDownfall.html]

Duoc thay huong dan nhu vay, co tung giai doan nhu vay,

Ket luan co phai la agile hay khong ?

* Agile Trinh bay o phan gioi thieu
* Tao table trong Word go rieng thuoc tinh cho tung table vat ly
* Mo hinh Physical DB ban dau chi de ten Table va moi quan he giua chung thoi