ĐẠI HỌC CÔNG NGHIỆP TP.HCM
Khoa Công nghệ Thông tin

THIẾT KẾ THÀNH PHẦN DỮ LIỆU

(Designing Data Components)

TS. Huỳnh Hữu Nghĩa

huynhhuunghia@iuh.edu.vn

Nội dung:

- **Khái quát**
- **B1: Chọn công nghệ truy cập dữ liệu**
- **B2: Chọn cách truy hồi và duy trì các đối tượng nghiệp vụ từ kho dữ liệu.**
- **B3: Xác định cách kết nối đến nguồn dữ liệu**
- **B4: Xác định chiến lược cho xử lý lỗi nguồn dữ liệu**
- **B5: Thiết kế các đối tượng tác nhân dịch vụ** (tùy chọn)

Khái quát

- Các thành phần lớp dữ liệu cung cấp quyền truy cập dữ liệu được lưu trữ trong phạm vi của hệ thống, và dữ liệu được hiển thị bởi các hệ thống được nối mạng khác.

- Gồm các thành phần truy cập dữ liệu cung cấp chức năng để truy cập dữ liệu được lưu trữ bên trong phạm vi hệ thống, và các thành phần tác nhân dịch vụ cung cấp chức năng truy cập dữ liệu được hiển thị bởi các hệ thống phụ trợ khác thông qua các dịch vụ Web.

Khái quát

- ✓ **Đầu tiên** là xác định các ràng buộc liên quan với dữ liệu được truy cập.
- ✓ **Bước tiếp theo** là lựa chọn một chiến lược ánh xạ và sau đó xác định hướng tiếp cận truy cập dữ liệu, bao gồm xác định các thực thể nghiệp vụ được sử dụng và định dạng các thực thể.
- ✓ **Sau đó** xác định cách các thành phần truy cập dữ liệu sẽ kết nối đến nguồn dữ liệu.
- ✓ **Cuối cùng**, xác định chiến lược xử lý lỗi đến quản lý các ngoại lệ nguồn dữ liệu.

B1: Chọn công nghệ truy cập dữ liệu

- **ADO.NET Entity Framework (EF).**
  - Sử dụng khi muốn tạo mô hình dữ liệu và ánh xạ đến một cơ sở dữ liệu; ánh xạ một lớp duy nhất đến nhiều bảng sử dụng kế thừa; hay truy vấn các lưu trữ liên quan khác với các dòng sản phẩm Microsoft SQL Server.

  - Dùng EF phù hợp khi có một mô hình đối tượng cần phải ánh xạ đến một mô hình quan hệ sử dụng một lược đồ linh hoạt, và cần linh hoạt chia tách lược đồ ánh xạ từ mô hình đối tượng.

B1: Chọn công nghệ truy cập dữ liệu

- **ADO.NET Data Services Framework.**
  - ADO.NET Data Service được xây dựng trên EF và cho phép hiển thị các phần của mô hình thực thể thông qua giao diện REST.
  - Sử dụng ADO.NET Data Services Framework nếu đang phát triển một RIA hay một ứng dụng rich client n tầng, và muốn truy cập dữ liệu thông qua một giao diện dịch vụ tập trung nguồn tài nguyên.

B1: Chọn công nghệ truy cập dữ liệu

- **ADO.NET Core**
  - Dùng nếu cần sử dụng API cấp thấp để kiểm soát toàn quyền truy cập dữ liệu ứng dụng, muốn tận dụng khoản đầu tư hiện có vào các nhà cung cấp ADO.NET, hay đang sử dụng logic truy cập dữ liệu truyền thống đối với cơ sở dữ liệu.
  - Phù hợp nếu không cần chức năng bổ sung được cung cấp bởi các công nghệ truy cập dữ liệu khác, hay đang xây dựng một ứng dụng mà phải hỗ trợ một trải nghiệm truy cập dữ liệu không được kết nối.

B1: Chọn công nghệ truy cập dữ liệu

- **ADO.NET Sync Services**
  - Dùng khi đang thiết kế một ứng dụng mà phải hỗ trợ các kịch bản được kết nối thỉnh thoản, hay các yêu cầu cộng tác giữa các cơ sở dữ liệu.

- **LINQ to XML**
  - Dùng khi sử dụng dữ liệu XML trong ứng dụng, và muốn thực thi các truy vấn sử dụng cú pháp LINQ.

B2: dữ liệu

- Sau khi xác định yêu cầu nguồn dữ liệu, tiếp theo là chọn chiến lược gắn vào các đối tượng hoặc thực thể nghiệp vụ từ kho dữ liệu và duy trì chúng trở lại kho dữ liệu.

- Một cản trở không phù hợp thường tồn tại giữa một mô hình dữ liệu hướng đối tượng và kho dữ liệu quan hệ đôi khi gây khó cho chuyển đổi dữ liệu giữa chúng.

B2: dữ liệu

- Có một số hướng tiếp cận để xử lý cản trở này, nhưng nó khác nhau về kiểu dữ liệu, cấu trúc, kỹ thuật giao dịch, và cách thức xử lý dữ liệu.

- Hầu hết các hướng tiếp cận phổ biến sử dụng các công cụ và khung làm việc O/RM. Kiểu thực thể sử dụng trong ứng dụng là yếu tố chính trong quyết định cách thức ánh xạ các thực thể đó đến các cấu trúc nguồn dữ liệu.

B2: dữ liệu

- **Hướng dẫn giúp chọn cách truy hồi và duy trì các đối tượng nghiệp vụ từ kho dữ liệu:**
  - Nên dùng O/RM framework thông dịch giữa các thực thể lĩnh vực và cơ sở dữ liệu. Trong môi trường **greenfield** *(có toàn quyền kiểm soát lược đồ cơ sở dữ liệu)* có thể sử dụng công cụ O/RM để sinh ra một lược đồ hỗ trợ mô hình đối tượng và cung cấp một ánh xạ giữa cơ sở dữ liệu và các thực thể lĩnh vực. Trong môi trường **brownfield** *(phải làm việc với một lược đồ cơ sở dữ liệu có sẵn)* có thể sử dụng công cụ O/RM để giúp ánh xạ giữa mô hình lĩnh vực và mô hình quan hệ

B2: dữ liệu

- **Hướng dẫn giúp chọn cách truy hồi và duy trì các đối tượng nghiệp vụ từ kho dữ liệu:**
  - Một mẫu phổ biến có liên quan với thiết kế OO là mô hình lĩnh vực, nó dựa trên mô hình hóa các thực thể của đối tượng với một lĩnh vực.
  - Khi có yêu cầu các đối tượng bổ sung trong mô hình lĩnh vực, và các thực thể có liên quan được nhóm thành aggregate roots.

B2: dữ liệu

- **Hướng dẫn giúp chọn cách truy hồi và duy trì các đối tượng nghiệp vụ từ kho dữ liệu:**
  - Với các ứng dụng Web hay các dịch vụ, nhóm thực thể và cung cấp các lựa chọn để tải một phần các thực thể lĩnh vực chỉ với dữ liệu cần thiết. Điều này giảm thiểu sử dụng các nguồn tài nguyên bằng cách tránh giữ các mô hình lĩnh vực được khởi tạo cho mỗi người dùng trong bộ nhớ, và cho phép các ứng dụng xử lý tải người dùng cao hơn.

B3: Cách kết nối nguồn dữ liệu

- Chọn một hướng tiếp cận phù hợp:
  - Connections
  - Connection Pooling
  - Transaction and Concurrency

B3: Cách kết nối nguồn dữ liệu

- **Connections:**
  - Đảm bảo mở các kết nối đến nguồn dữ liệu càng sớm càng tốt và đóng lại càng sớm càng tốt. Điều này đảm bảo các nguồn tài nguyên bị khóa trong thời gian càng ngắn càng tốt, và có sẵn cho các quy trình khác.
  - Thực hiện giao dịch thông qua một kết nối duy nhất nếu có thể. Điều này cho phép sử dụng các đặc trưng giao dịch của ADO.NET mà không yêu cầu các dịch vụ điều phối giao dịch phân tán.

B3: Cách kết nối nguồn dữ liệu

- **Connections:**
  - Sử dụng kết nối tổng hợp và điều chỉnh hiệu suất dựa trên các kết quả thu được bằng cách chạy các kịch bản tải mô phỏng. Xem xét điều chỉnh các mức cô lập kết nối cho các truy vấn dữ liệu. Nếu đang xây dựng một ứng dụng với các yêu cầu thông lượng cao, các hoạt động dữ liệu đặc biệt có thể được thực hiện ở các mức độ cô lập thấp hơn so với phần còn lại của giao dịch. Kết hợp các mức cô lập có thể có một tác động tiêu cực đến tính nhất quán dữ liệu, cho nên phải phân tích cẩn thận tùy chọn này trên từng trường hợp.

B3: Cách kết nối nguồn dữ liệu

- **Connections:**
  - Đối với lý do bảo mật, tránh sử dụng một Hệ thống hay Tên nguồn dữ liệu người dùng (DSN) để lưu trữ thông tin kết nối.
  - Thiết kế logic thử lại để quản lý tình huống trong đó các kết nối đến nguồn dữ liệu bị mất hay hết thời gian.
  - Các dòng lệnh lô và thực thi chúng đối với cơ sở dữ liệu khi có thể để giảm các chuyến đi khứ hồi đến máy chủ cơ sở dữ liệu.

B3: Cách kết nối nguồn dữ liệu

- **Connections:**
  - Thích xác thực Windows hơn xác thực SQL Server. Nếu đang sử dụng Microsoft SQL Server, xem xét sử dụng xác thực Windows với hệ thống con đáng tin cậy.
  - Nếu sử dụng xác thực SQL, đảm bảo rằng sử dụng các tài khoản tùy chỉnh với mật khẩu mạnh, giới hạn các quyền cho mỗi tài khoản trong SQL Server sử dụng các vai trò cơ sở dữ liệu, thêm ACLs đến bất kỳ file nào được sử dụng để lưu trữ các chuỗi kết nối, và mã hóa các chuỗi kết nối trong các file cấu hình.

B3: Cách kết nối nguồn dữ liệu

- **Connections:**
  - Sử dụng các tài khoản với đặc quyền tối thiểu trong cơ sở dữ liệu, và yêu cầu người gọi gửi thông tin xác nhận đến lớp dữ liệu cho mục đích kiểm tra.
  - Không lưu trữ các mật khẩu xác thực người dùng trong một cơ sở dữ liệu; hoặc văn bản gốc hay được mã hóa. Thay vào đó, lưu trữ các băm mật khẩu sử dụng giá trị muối (các bít ngẫu nhiên được sử dụng như các đầu vào cho hàm băm).

B3: Cách kết nối nguồn dữ liệu

- **Connections:**
  - Nếu đang sử dụng các câu lệnh SQL để truy cập nguồn dữ liệu, hiểu các ranh giới tin cậy và sử dụng hướng tiếp cận tham số hóa để tạo ra các truy vấn thay vì nối chuỗi để bảo vệ chống lại các cuộc tấn công SQL injection.
  - Bảo vệ dữ liệu nhạy cảm gửi qua mạng đến và từ SQL Server. Xin lưu ý rằng xác thực Windows bảo vệ thông tin đăng nhập, nhưng không phải dữ liệu ứng dụng. Sử dụng IPSec hoặc SSL để bảo vệ dữ liệu trong kênh.

B3: Cách kết nối nguồn dữ liệu

- **Connection Pooling:**
  - Để tối đa hóa hiệu quả của kết nối pool, xem xét sử dụng mô hình bảo mật hệ thống con đáng tin cậy và tránh mạo danh nếu có thể. Bằng cách sử dụng tối thiểu số lượng tập ùy nhiệm, tăng khả năng kết nối pool có sẵn có thể được tái sử dụng và giảm sự thay đổi một kết nối pool tràn. Nếu mỗi cuộc gọi sử dụng thông tin đăng nhập khác nhau, ADO.NET phải tạo một kết nối mới mỗi lần.

B3: Cách kết nối nguồn dữ liệu

- **Connection Pooling:**
  - Các kết nối vẫn mở trong thời gian dài có thể giữ tài nguyên trên máy chủ. Một nguyên nhân cụ thể là mở kết nối sớm và đóng nó muộn.
  - Các kết nối có thể được giữ mở cho thời gian dài khi sử dụng các đối tượng DataReader, nó chỉ hợp lệ trong khi kết nối được mở.

B3: Cách kết nối nguồn dữ liệu

- **Transactions and Concurrency:**
  - Nếu đang truy cập một nguồn dữ liệu duy nhất, sử dụng các giao dịch dựa trên kết nối bất cứ khi nào có thể. Nếu đang sử dụng các giao dịch thủ công hay minh bạch, xem xét triển khai giao dịch bên trong một stored procedure. Nếu không thể sử dụng các giao dịch, triển khai các phương thức bù để hoàn nguyên kho dữ liệu về trạng thái trước đó.

B3: Cách kết nối nguồn dữ liệu

- **Transactions and Concurrency:**
  - Nếu đang sử dụng các giao dịch nguyên tử chạy lâu, tránh giữ các khóa trong thời gian dài. Trong kịch bản này, sử dụng các khóa bù thay thế. Nếu các giao dịch mất nhiều thời gian để hoàn thành, xem xét sử dụng giao dịch không đồng bộ gọi lại client khi hoàn thành. Ngoài ra, xem xét sử dụng nhiều bộ kết quả hoạt động trong các ứng dụng đồng thời chậm chạp để tránh các vấn đề bế tắt (deadlock) tiềm ẩn.

B3: Cách kết nối nguồn dữ liệu

- **Transactions and Concurrency:**
  - Nếu khả năng xung đột dữ liệu từ những người dùng đồng thời là thấp, nên sử dụng khóa tối ưu trong khi truy cập dữ liệu để áp dụng cập nhật sau cùng là hợp lệ. Nếu khả năng xung đột dữ liệu từ các người dùng đồng thời là cao, nên sử dụng khóa bi quan trong khi truy cập dữ liệu để những cập nhật chỉ có thể được áp dụng cho phiên bản mới nhất. Cũng xem các vấn đề đồng thời khi truy cập dữ liệu tĩnh trong các ứng dụng hay khi sử dụng các tiến trình để thực hiện các hoạt động không đồng bộ.

B3: Cách kết nối nguồn dữ liệu

- **Transactions and Concurrency:**
  - Giữ các giao dịch càng ngắn càng tốt có thể để giảm thiểu thời gian khóa và cải thiện đồng thời. Tuy nhiên, xem các giao dịch ngắn và duy nhất có thể dẫn đến giao diện trò chuyện nếu nó yêu cầu nhiều cuộc gọi để hoàn thành một hoạt động.
  - Sử dụng mức cách ly phù hợp. Sự đánh đổi là tính nhất quán dữ liệu so với sự tranh chấp. Một mức cô lập cao sẽ cung cấp tính nhất quán dữ liệu cao hơn với mức đồng thời. Mức cô lập thấp hơn cải thiện hiệu suất bằng cách hạ thấp sự tranh chấp với tính nhất quán.

B3: Cách kết nối nguồn dữ liệu

➢ Transactions and Concurrency

Có thể lựa chọn ba loại hỗ trợ giao dịch:

- System.Transactions.
- ADO.NET Transactions.
- T-SQL (Database) Transactions.

B4: Chiến lược xử lý lỗi

- **Exceptions:**
  - Xác định các ngoại lệ cần được bắt và xử lý trong lớp truy cập dữ liệu. Deadlocks, các vấn đề kết nối, và kiểm tra đồng thời lạc quan thường có thể được giải quyết trong lớp dữ liệu.
  - Xem xét triển khai một quá trình thử lại cho các hoạt động xảy ra các lỗi nguồn dữ liệu hay hết thời gian, nhưng chỉ khi nào an toàn mới thực hiện.
  - Xem xét sử dụng các khung làm việc hiện có như các mẫu và thực hành Enterprise Library để triển khai một chiến lược quản lý và xử lý ngoại lệ nhất quán.

B4: Chiến lược xử lý lỗi

- **Exceptions:**
  - Thiết kế một chiến lược tuyên truyền ngoại lệ phù hợp. Ví dụ, cho phép các ngoại lệ nổi lên trên các lớp ranh giới nơi chúng có thể được ghi lại và chuyển đổi khi cần thiết trước khi chuyển chúng đến lớp tiếp theo.
  - Thiết kế một chiến lược ghi chú và thông báo phù hợp cho các lỗi và ngoại lệ quan trọng không tiết lộ thông tin nhạy cảm.

B4: Chiến lược xử lý lỗi

- **Retry Logic:**
  - Thiết kế logic thử lại để xử lý các lỗi, như những lỗi có thể xảy ra chuyển đổi dự phòng máy chủ hay cơ sở dữ liệu. Logic thử lại cần bắt bất kỳ các lỗi xảy ra trong khi kết nối đến cơ sở dữ liệu hay thực thi các lệnh (truy vấn hay giao dịch) đối với cơ sở dữ liệu.

  - Khi một lỗi xảy ra, thành phần dữ liệu sẽ thiết lập lại kết nối bằng cách đóng bất kỳ các kết nối đang tồn tại và cố gắng tạo kết nối mới, và sau đó thực thi lại các lệnh bị lỗi nếu cần thiết.

B4: Chiến lược xử lý lỗi

- **Retry Logic:**
  - Nên thử lại quá trình chỉ một số lần nhất định, và cuối cùng sau đó từ bỏ và trả về một ngoại lệ thất bại. Đảm bảo rằng các truy vấn và yêu cầu, và bất kỳ thử lại đến sau, được thực thi không đồng bộ để chúng không khiến ứng dụng không phản hồi.

B5: Các đối tượng tác nhân dịch vụ

- **Thiết kế:**
  - Sử dụng công cụ phù hợp để bổ sung một tham chiếu dịch vụ. Điều này sẽ sinh ra một ủy thác và các lớp dữ liệu đại diện hợp đồng dữ liệu từ dịch vụ.
  - Xác định cách mà dịch vụ sẽ được sử dụng trong ứng dụng. Đối với hầu hết các ứng dụng, tác nhân dịch vụ hoạt động như một lớp trừu tượng giữa lớp nghiệp vụ và dịch vụ từ xa, và có thể cung cấp một giao diện nhất quán bất kể định dạng dữ liệu. Trong các ứng dụng nhỏ hơn, lớp trình bày, có thể truy cập tác nhân dịch vực trực tiếp


Q&A