Trường Đại học Công nghệ - ĐHQGHN
Khoa Công nghệ thông tin

BÀI TẬP LỚN: PHÂN TÍCH & THIẾT KẾ HƯỚNG ĐỐI TƯỢNG

Giảng viên: PGS.TS. Đặng Đức Hạnh
ThS. Trần Mạnh Cường

IDENTIFY DESIGN MECHANISM

ỨNG DỤNG CHĂM SÓC SỨC KHỎE TRỰC TUYẾN

Ngày: 10/05/2024

Chuẩn bị bởi: Nhóm 5

Mục lục

Lịch sử sửa đổi

Họ tên	Thời gian	Lý do sửa đổi	Phiên bản
Nguyễn Thị Ngọc Ánh	10/05/2024	Khởi tạo tài liệu	1.0
Nguyễn Thị Ngọc Ánh	11/05/2024	Liệt kê các cơ chế kiến trúc	1.1
Lê Trọng Minh	12/05/2024	Viết hoàn thiện nội dung cơ chế cài đặt Security	1.2
Hoàng Văn Nguyên Phạm Hoàng Hải	13/05/2024	Viết hoàn thiện nội dung cơ chế cài đặt Persistency và Distribution	1.3

1. Các cơ chế kiến trúc

1.1. Cơ chế phân tích

1.1.1. Persistence

Các lớp có vai trò lưu trữ dữ liệu, cần xác định:

- **Volume:** số lượng đối tượng cần lưu trữ là bao nhiêu?
- **Duration:** các đối tượng thường được lưu trong bao lâu?
- **Retrieval mechanism:** một đối tượng cụ thể được xác định và lấy về như thế nào?
- **Reliability:** các đối tượng cần sống sót nếu như một tiến trình bị crash, hay cả hệ thống crash?
- **Update frequency:** các đối tượng được tạo một lần và gần như không bị thay đổi, hay các đối tượng thường xuyên được cập nhật?

1.1.2. Communication

Đối với tất cả các thành phần mô hình cần giao tiếp với các thành phần hoặc dịch vụ đang chạy trong các tiến trình hoặc luồng khác, cần xác định:

- **Latency**: Các tiến trình phải giao tiếp với nhau nhanh đến mức nào?
- **Synchronicity**: Giao tiếp không đồng bộ
- **Size of message**: Một chuỗi rộng có thể phù hợp hơn là một số
- **Protocol**: Kiểm soát các luồng, đệm, v.v.

1.1.3. Security

Các lớp có yêu cầu bảo mật cao, như HealthRecord, MedicalImage, Appointment, ExaminationResult, User cần xác định:

- **User granularity:** Người dùng hệ thống có các role gì?
- **Security rules:** Đề ra các chuẩn mực về bảo mật để bảo vệ dữ liệu người dùng
- **User privilege:** Các role của người dùng hệ thống có thể làm được những gì trên hệ thống?

1.1.4. Các cơ chế khác

- **Error detection / handling / reporting**: Các lỗi nên được phát hiện, xử lý và báo cáo như thế nào?
- **Distribution**: Dữ liệu nên được lưu vào máy chủ nào?
- **Transaction management**: Một giao dịch nên được hoàn tất như thế nào?

1.2. Analysis-to-Design-to-Implementation Mechanisms Map

Cơ chế phân tích	Cơ chế thiết kế	Cơ chế cài đặt
Volume	RDBMS	JDBC + MySQL
Duration	RDBMS	JDBC + MySQL
Retrieval mechanism	RDBMS	JDBC + MySQL
Reliability	RDBMS	JDBC + MySQL
Latency	Communication	
Synchronicity	Communication	
Size of message	Communication	
Protocol	Communication	
Update frequency	Security	Secure User Set-up

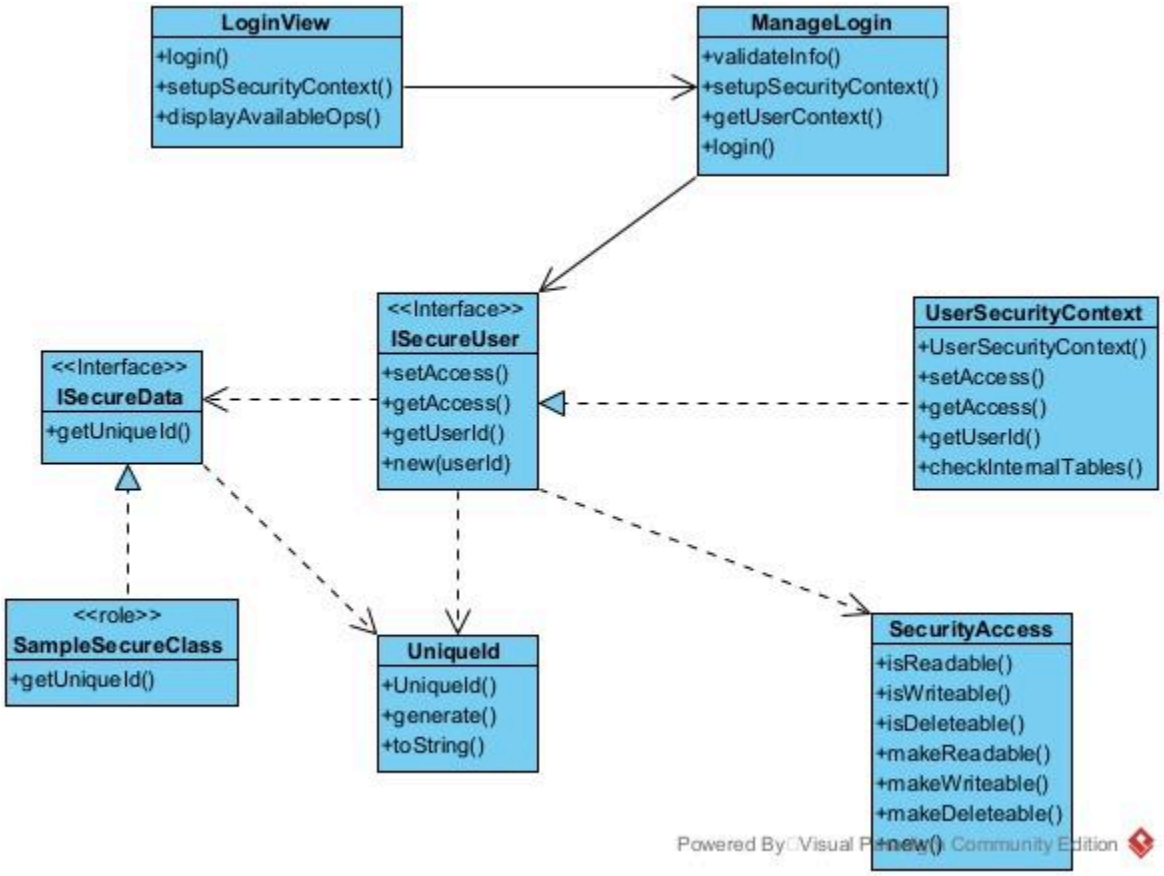
User granularity	Security	Secure User Setup
Security rules	Security	Secure Data Access
User privileges	Security	Secure Data Access
Privilege types	Security	Secure Data Access
Error detection / handling / report		
Distribution		RMI
Transaction management		

1.3. Cơ chế cài đặt

1.3.1. Security

1.3.1.1. Static view

1.3.1.1.1. Biểu đồ lớp



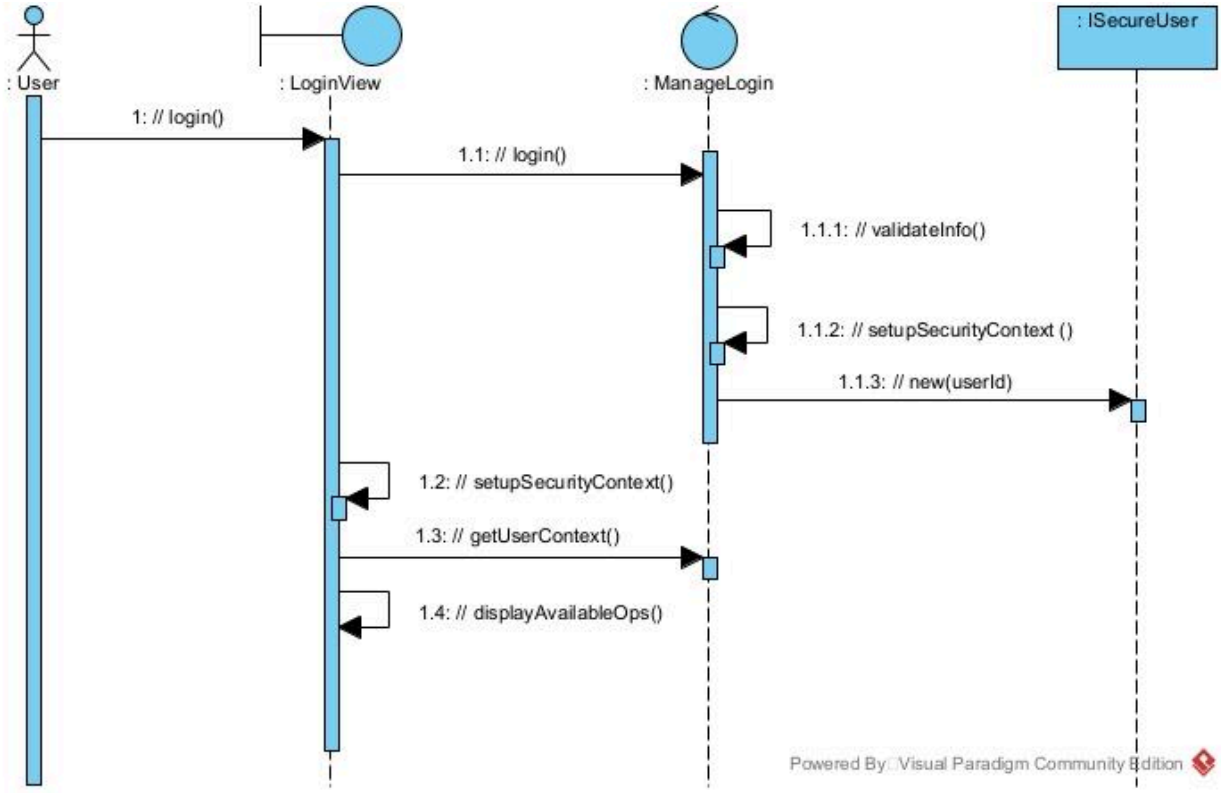
Ảnh 1.1. Biểu đồ lớp cơ chế security - static view.

1.3.1.1.2. Mô tả biểu đồ lớp

- **ISecureData**: Cơ chế phân tích: security
- **Security Access**: Cơ chế phân tích: security
- **UserSercurityContext**: Cơ chế phân tích: security
- **UniqueId**: Cơ chế phân tích: security

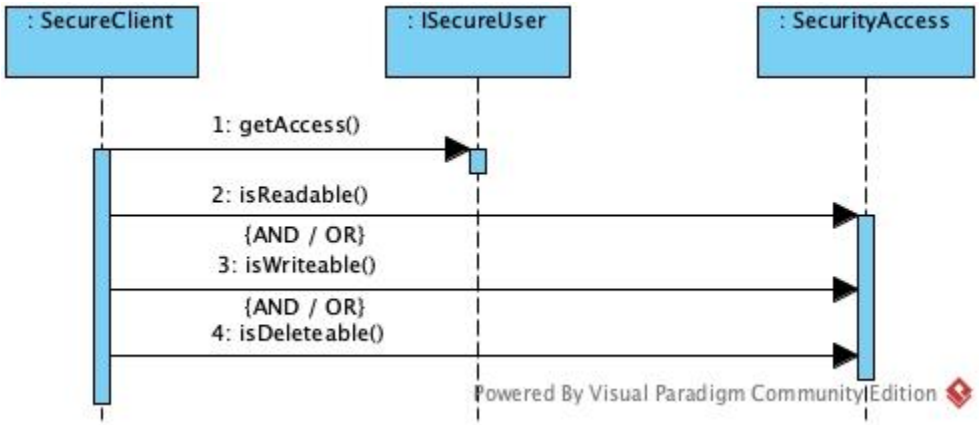
- **ISecureUser**: Cơ chế phân tích: security
- **ManageLogin**: Cơ chế phân tích: security

1.3.1.2. Dynamic view: Secure User Setup



Ảnh 1.2. Biểu đồ trình tự Secure User Setup.

1.3.1.3. Dynamic view: Secure Data Setup

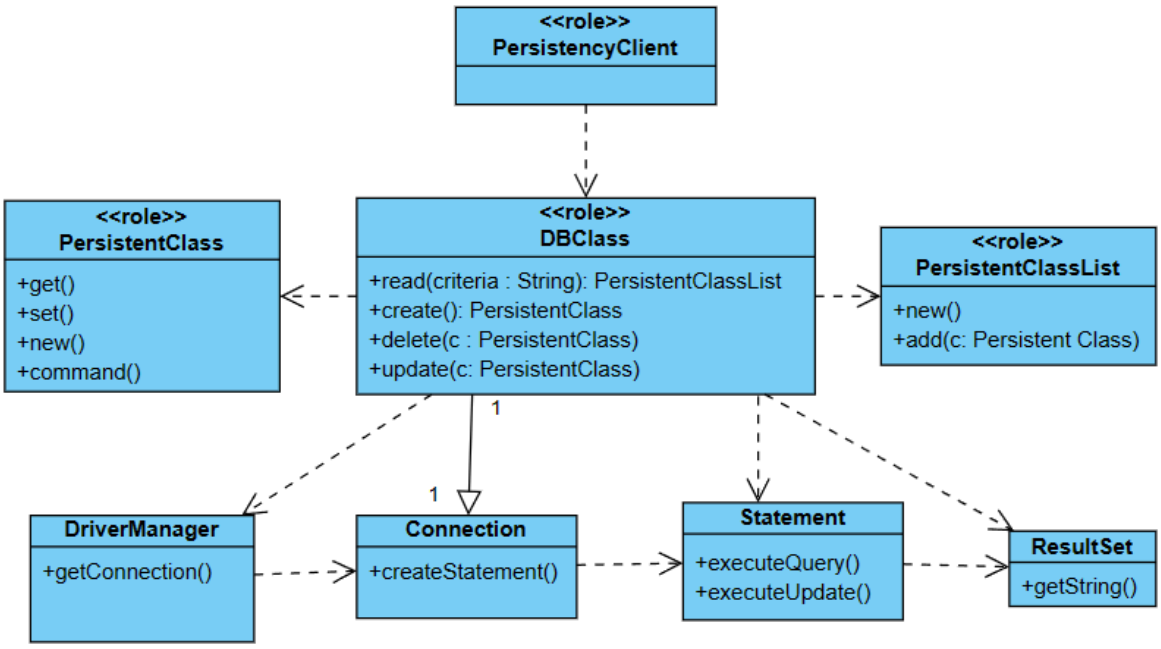


Ảnh 1.3. Biểu đồ trình tự Secure Data Access.

1.3.2. Persistency - RDBMS - JDBC

1.3.2.1. Static view

1.3.2.1.1. Biểu đồ lớp



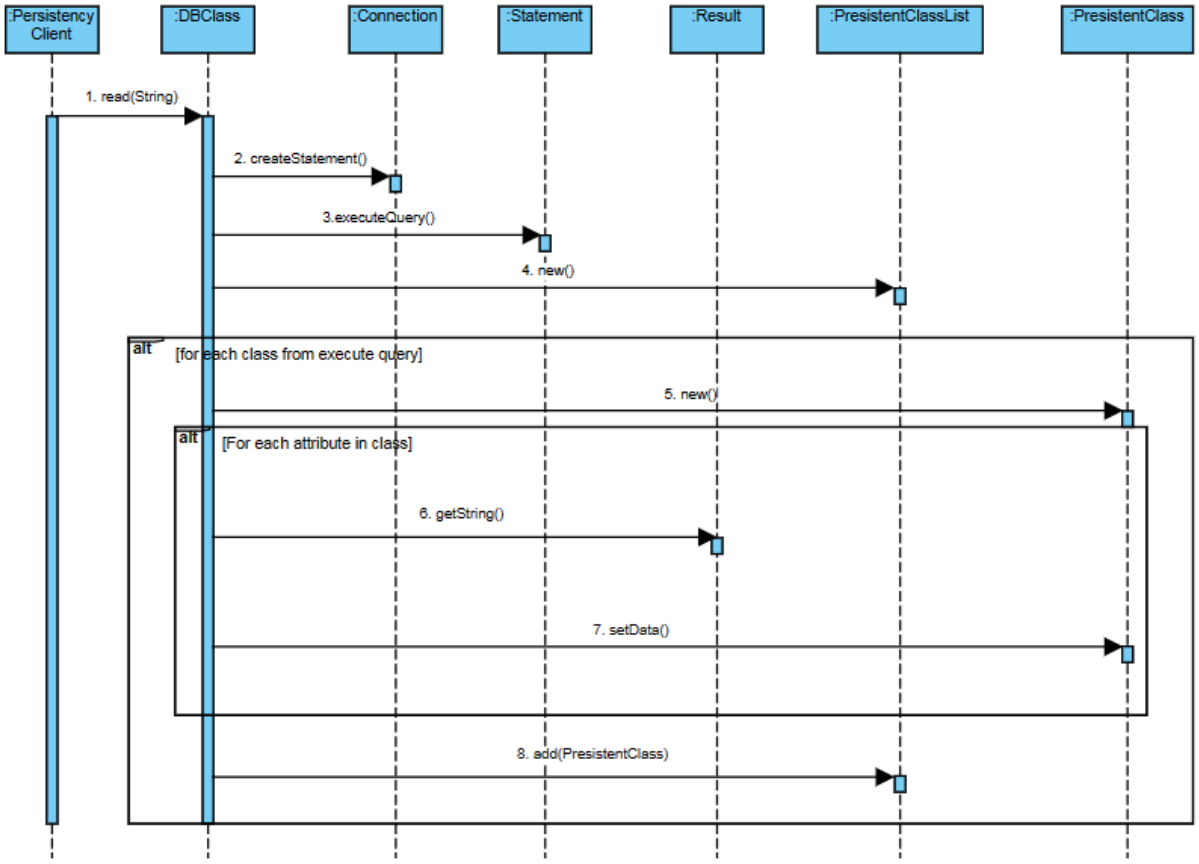
Ảnh 1.4. Biểu đồ lớp cơ chế persistency - static view.

1.3.2.1.2. Mô tả biểu đồ lớp

- **PersistencyClient**: Một ví dụ về một client của một lớp persistent.
- **PersistentClass**: Một ví dụ về một lớp có tồn tại trong ứng dụng.
- **PersistentClassList**: Một danh sách các đối tượng PersistentClass.
- **Statement**: Lớp được sử dụng để thực thi một câu lệnh SQL tĩnh và thu được kết quả do nó tạo ra. Các câu lệnh SQL không có tham số thường được thực thi bằng cách sử dụng các đối tượng Statement.
- **DBClass**: Một mẫu cho một lớp chịu trách nhiệm làm cho một lớp khác persistent. Mỗi lớp persistent sẽ có một DBClass tương ứng.

- **Connection**: Một kết nối (phiên) với cơ sở dữ liệu cụ thể. Trong ngữ cảnh là một kết nối, các câu lệnh SQL được thực thi và kết quả được trả về.
- **ResultSet**: Bộ kết quả cung cấp quyền truy cập vào một bảng dữ liệu. Đối tượng ResultSet thường được tạo bằng cách thực thi một Statement.
- **DriverManager**: Dịch vụ cơ bản để quản lý một bộ trình điều khiển JDBC.

1.3.2.2. Dynamic view: RDBMS Read



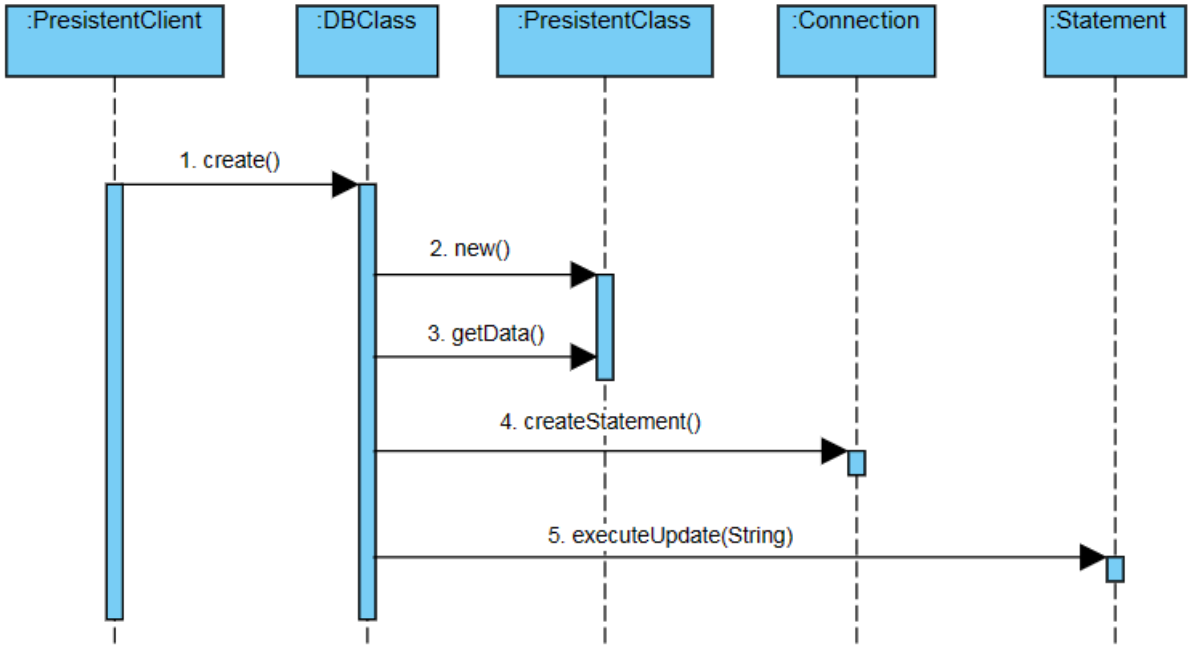
Ảnh 1.5. Biểu đồ trình tự JDBC RDBMS Read.

Biểu đồ này cho thấy việc đọc một đối tượng như thế nào.

Đầu tiên, SampleDBManager tạo một transaction chỉ đọc mới, sau đó tra cứu đối tượng bằng thao tác Map "get()". Khi đối tượng đã được tìm thấy, nó có thể được đọc bằng thao tác "getData()", transaction đã được thực hiện. RETAIN_HOLLOW được chỉ định, do đó, các tham chiếu đến đối tượng và dữ liệu được truy xuất có thể được sử dụng bên

ngoài transaction truy xuất. Sau khi transaction được thực hiện, đối tượng có thể được cập nhật.

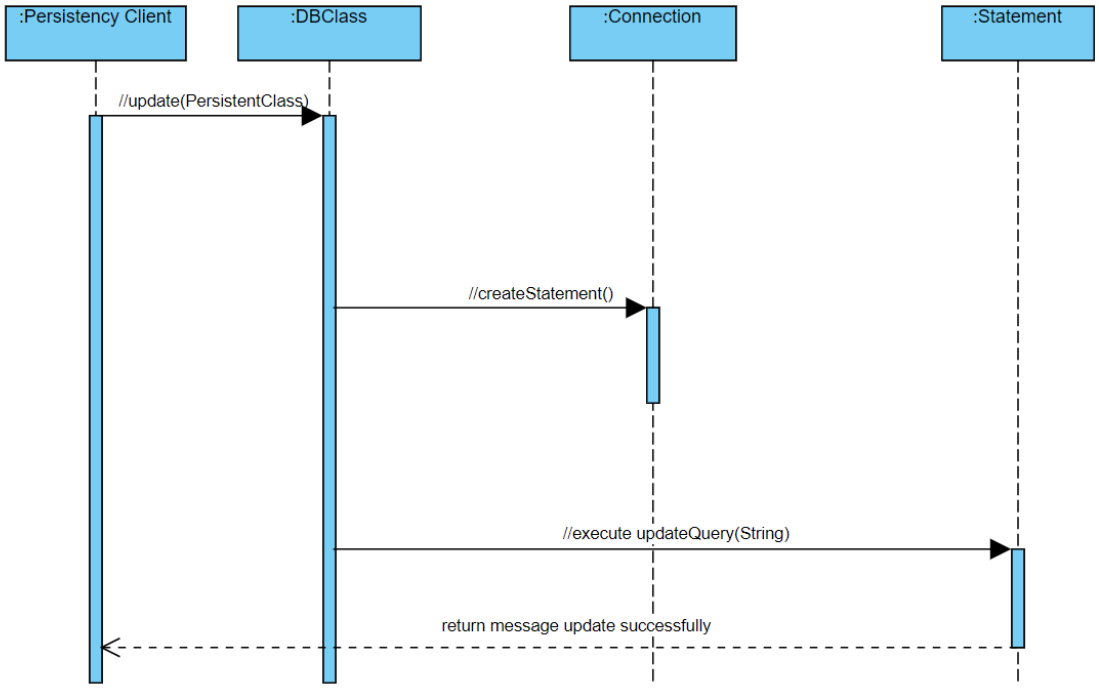
1.3.2.3. Dynamic view: RDBMS Create



Ảnh 1.6. Biểu đồ trình tự JDBC RDBMS Create

Biểu đồ này mô tả tạo một PersistentClass mới trong cơ sở dữ liệu như thế nào. Đầu tiên, SampleDBManager tạo một transaction và gọi constructor cho PersistentClass. Khi đã được tạo, lớp được thêm vào cơ sở dữ liệu thông qua "put()". Sau đó, transaction được thực hiện.

1.3.2.4. Dynamic view: RDBMS Update

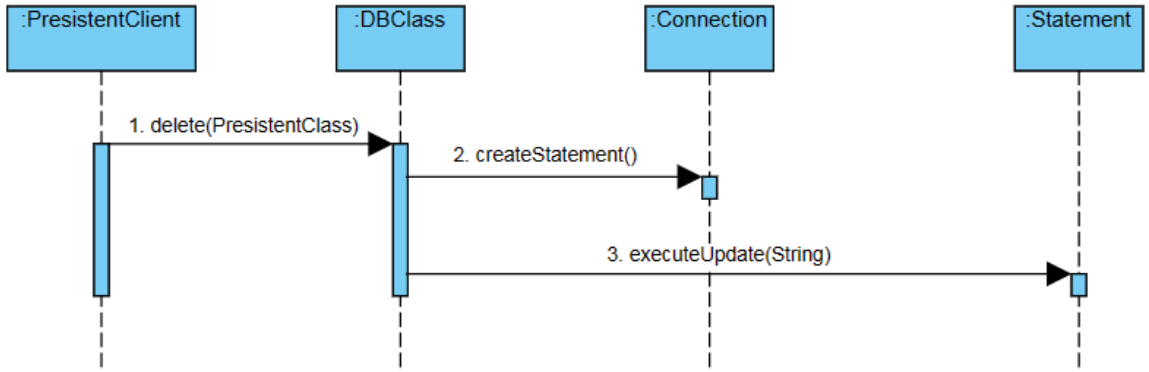


Ảnh 1.7. Biểu đồ trình tự JDBC RDBMS Update

Sơ đồ này cho thấy cách cập nhật một đối tượng.

Để cập nhật thông tin về một đối tượng cụ thể, client yêu cầu DBClass hỗ trợ. Sau đó, DBClass tạo một tập hợp các hướng dẫn bằng cách dùng createStatement() từ lớp Connection, chỉ định cách cập nhật thông tin trong cơ sở dữ liệu cho đối tượng đã cho. Sau đó, các hướng dẫn này được thực thi và dẫn đến việc cập nhật dữ liệu trong cơ sở dữ liệu.

1.3.2.5. Dynamic view: RDBMS Delete



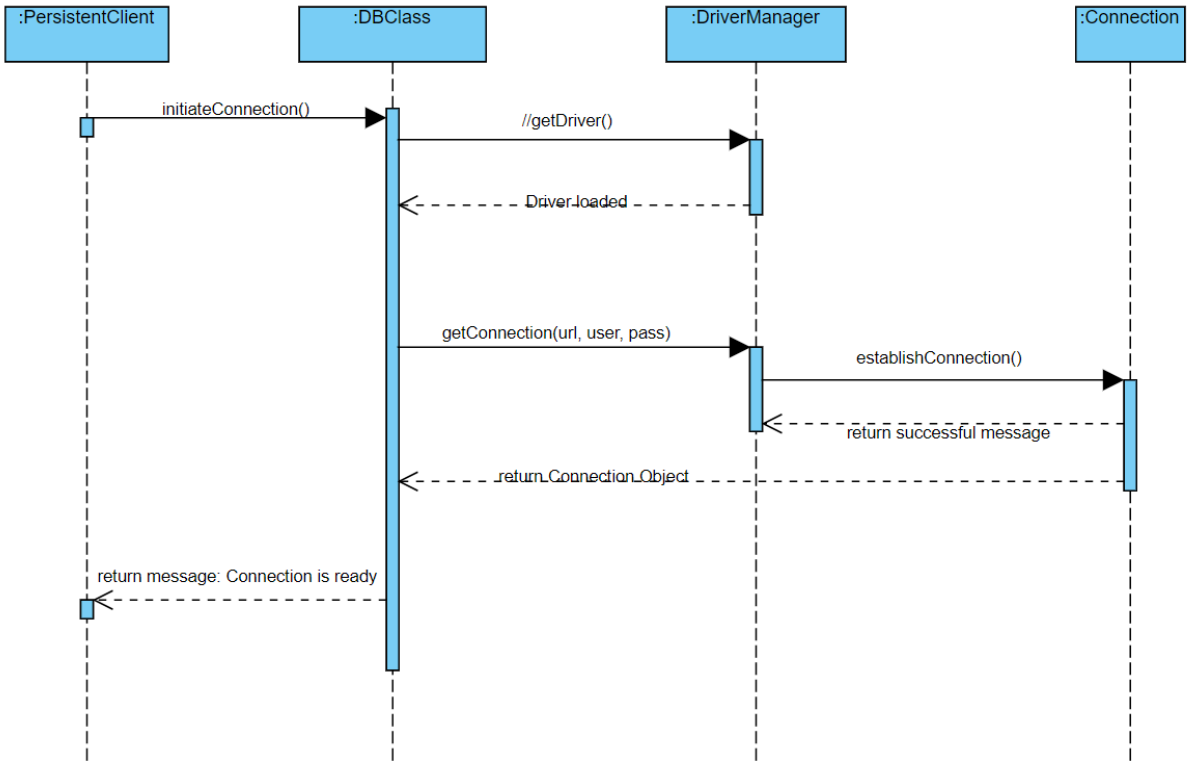
Ảnh 1.8. Biểu đồ trình tự JDBC RDBMS Delete

Biểu đồ này mô tả xóa một đối tượng khỏi cơ sở dữ liệu như thế nào.

Đầu tiên, SampleDBManager tạo một transaction mới, loại bỏ mọi phần cấu thành và sau đó loại bỏ đối tượng bằng cách sử dụng "remove()". Sau đó, đối tượng sẽ bị xóa hoàn toàn khỏi cơ sở dữ liệu ObjectStore ngay lập tức thông qua ObjectStore.destroy(). Khi đối tượng đã bị xóa, transaction được thực hiện.

Do đó, trong ObjectStore, việc xóa có hai bước - xóa khỏi lớp container là cơ sở dữ liệu trong bộ nhớ và xóa khỏi cơ sở dữ liệu vật lý. Đó là vì muốn việc xóa diễn ra ngay lập tức, trái ngược với bộ nhớ đệm.

1.3.2.6. Dynamic view: RDBMS Initialize



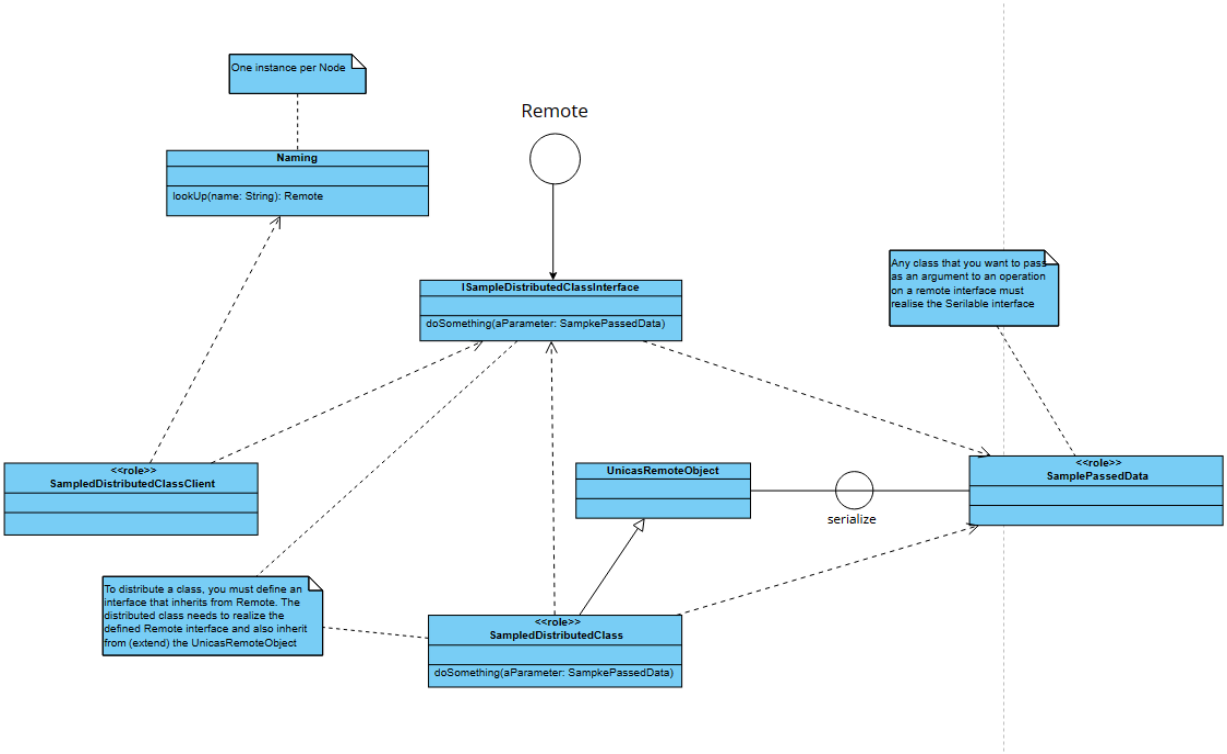
Ảnh 1.9. Biểu đồ trình tự JDBC RDBMS Initialize

Trình tự bắt đầu bằng việc PersistentClient yêu cầu DBClass thiết lập kết nối. DBClass tải JDBC driver cần thiết thông qua DriverManager.getDriver(). Tiếp theo, DBClass khởi tạo DriverManager để kết nối với cơ sở dữ liệu. Khi kết nối thành công, kích hoạt phần Connection. Kết nối đã tạo sau đó sẽ được đưa trở lại DBClass. Cuối cùng, DBClass thông báo cho PersistentClient rằng kết nối cơ sở dữ liệu đã sẵn sàng để sử dụng, hoàn thành trình tự và giải thích quy trình từng bước khởi tạo kết nối JDBC.

1.3.3. Distribution – RMI

1.3.3.1. Static view

1.3.3.1.1. Biểu đồ lớp



1.3.3.1.2. Mô tả biểu đồ lớp

Naming: Đây là cơ chế bootstrap để lấy tham chiếu tới các đối tượng từ xa dựa trên cú pháp Uniform Resource Locator (URL). URL cho một đối tượng từ xa được chỉ định bằng cách sử dụng tên máy chủ, cổng và tên thông thường:

- rmi://host:port/name
- host = tên host của registry (defaults to current host)
- port = port của registry (defaults to the registry port number)
- name = tên của đối tượng từ xa

SampleDistributedClass: Ví dụ về một lớp được phân tán.

Remote: Giao diện Remote dùng để xác định tất cả các đối tượng từ xa. Bất kỳ đối tượng nào là đối tượng từ xa đều phải trực tiếp hoặc gián tiếp

implements giao diện này. Chỉ những phương thức được chỉ định trong giao diện từ xa mới có sẵn từ xa.

Các lớp triển khai có thể implements bất kỳ số lượng giao diện từ xa nào và có thể mở rộng các lớp triển khai từ xa khác.

Đối với tất cả các lớp implements giao diện Remote, một stub từ xa và một skeleton từ xa được tạo. Các lớp này xử lý giao tiếp cần thiết để hỗ trợ phân tán.

SampleDistributedClassClient: Ví dụ về client của một lớp phân tán.

SamplePassedData: Ví dụ về dữ liệu được truyền đến / từ một lớp phân tán.

UnicastRemoteObject

ISampleDistributedClassInterface: Ví dụ về một giao diện được định nghĩa cho một lớp phân tán.

Serializable: Bất kỳ lớp nào bạn muốn truyền làm đối số cho một hoạt động trên giao diện từ xa đều phải implements giao diện Serializable.

1.3.3.2. Dynamic view