**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**Môn: Mẫu thiết kế hướng đối tượng và ứng dụng**

**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

DATABASE ACCESS MANAGEMENT (DAM) FRAMEWORK

**Giáo viên:** Nguyễn Đức Huy

Hồ Tuấn Thanh

**Sinh viên:** Trang Trung Hoàng Phúc 1612521

Tô Hiếu Minh 1612377

Lâm Cương Đạt 1612098

Mục lục

[1 Sơ đồ lớp 3](#_Toc29153432)

[2 Giải thích các lớp 3](#_Toc29153433)

[2.1 Các lớp thao tác trên cơ sở dữ liệu 3](#_Toc29153434)

[2.1.1 Class DbFactory 3](#_Toc29153435)

[2.1.2 Class Parser 4](#_Toc29153436)

[2.1.3 Class DBManager 4](#_Toc29153437)

[2.2 Lớp dành cho người dùng 5](#_Toc29153438)

[Class ORMManager 5](#_Toc29153439)

[2.3 Các lớp hỗ trợ truy vấn 6](#_Toc29153440)

[2.3.1 Class ExecutableQuery 6](#_Toc29153441)

[2.3.2 Class SelectQuery<T>: 6](#_Toc29153442)

[2.3.3 Class Include 7](#_Toc29153443)

[2.3.4 Class Aggregate 8](#_Toc29153444)

[2.4 Các lớp điều kiện 8](#_Toc29153445)

[2.4.1 Class Condition 8](#_Toc29153446)

[2.4.2 Class Not 8](#_Toc29153447)

[2.4.3 Class Connector 8](#_Toc29153448)

[2.4.4 Class Restriciton 8](#_Toc29153449)

[3 Các mẫu thiết kế hướng đối tương được áp dụng 9](#_Toc29153450)

[3.1 Mẫu Abstract Factory 9](#_Toc29153451)

[3.1.1 Sơ đồ lớp 9](#_Toc29153452)

[3.1.2 Ý nghĩa 9](#_Toc29153453)

[3.1.3 Sử dụng 9](#_Toc29153454)

[3.2 Mẫu Composite 10](#_Toc29153455)

[3.2.1 Sơ đồ lớp 10](#_Toc29153456)

[3.2.2 Ý nghĩa 10](#_Toc29153457)

[3.2.3 Sử dụng 10](#_Toc29153458)

[3.3 Mẫu Template Method 11](#_Toc29153459)

[3.3.1 Sơ đồ lớp 11](#_Toc29153460)

[3.4 Mẫu Decorator 13](#_Toc29153461)

[3.4.1 Sơ đồ lớp 13](#_Toc29153462)

[3.4.2 Ý nghĩa 13](#_Toc29153463)

[3.4.3 Sử dụng 13](#_Toc29153464)

[3.5 Mẫu Strategy 14](#_Toc29153465)

[3.5.1 Sơ đồ lớp 14](#_Toc29153466)

[3.5.2 Ý nghĩa 14](#_Toc29153467)

[3.5.3 Sử dụng 14](#_Toc29153468)

[4 Mức độ hoàn thành 15](#_Toc29153469)

# Sơ đồ lớp

Nhóm có chuẩn bị fle đính kèm sơ đồ lớp dưới định dạng SVG để thầy cô tiện theo dõi vì sơ đồ lớp khá lớn. Thầy cô có thể sử dụng trình duyệt web để mở file ClassDiagram.SVG.



# Giải thích các lớp

## Các lớp thao tác trên cơ sở dữ liệu

### Class DbFactory

Là lớp nhà máy ảo, mỗi loại nhà máy có 2 thuộc tính là DBManager và Parser.

Người dùng cần phải định nghĩa cơ sở dữ liệu mình sẽ sử dụng, từ đó nhà máy sẽ tạo DBManager và Parser tương ứng.

Phương thức

|  |  |
| --- | --- |
| **DBManager** CreateDBManager(string host, int port, string database, string username, string password) | Trả về DBManager tương ứng với cơ sở dữ liệu được tạo ra từ Factory. |
| **Parser** CreateParser() | Tạo Parser tương ứng với cơ sở dữ liệu được tạo ra từ Factory. |

Các lớp con

Class **MySQLFactory**: nhà máy quản lý cơ sở dữ liệu MySQL.

Class **PostgresFactory**: nhà máy quản lý cơ sở dữ liệu Postgres

### Class Parser

Là lớp ảo, quy định cách quy đổi thao tác của người dùng thành câu truy vấn tương ứng với loại cơ sở dữ liêu.

Các phương thức

|  |  |
| --- | --- |
| **string** ParseValue(Object obj, Type type) | Constructor |
| **string** ParseInsertQuery(string tableName, Dictionary<string, string> values) | Tạo ra câu chèn dòng mới để insert cho cơ sở dữ liệu. |
| **string** ParseDeleteQuery(string tableName, string whereCondition) | Tạo ra câu query để xóa cho cơ sở dữ liệu. |
| **string** ParseUpdateQuery(string tableName, Dictionary<string, string> setValues, string whereCondtion) | Tạo ra câu query để cập nhật các dòng cho cơ sở dữ liệu. |
| **string** ParseSelectQuery(string tableName, string projections, string whereCondition, string groupByCondition = "", string havingCondition = "", string orderBy = "") | Tạo ra câu query để chọn các dòng thỏa điều kiện, gom nhóm, điều kiện nhóm, sắp xếp, ... |

Các lớp con

Class **MySQLParser**: quy định cách quy đổi cho cơ sở dữ liệu MySQL.

Class **PostgresParser**: quy định cách quy đổi cho cơ sở dữ liệu Postgres.

### Class DBManager

Là lớp ảo, quy định các phương thức kết nối với cơ sở dữ liệu, cũng như gọi và nhận kết quả truy vấn.

Các phương thức

|  |  |
| --- | --- |
| **void** Connect(string host, int port, string database, string username, string password) | Hàm kết nối, kết nối thành công sẽ in ra "Connect Successfully", ngược lại in ra "Connect Failed" |
| **void** Disconnect() | Hàm đóng kết nối. |
| **List<List<string>>** Select(string query) | Trả về danh sách các thông tin lấy dược từ cơ sở dữ liệu |
| **int** Insert(string query) | Thực hiện thao tác chèn lên cơ sở dữ liệu, trả về số dòng chèn được. |
| **int** Update(string query) | Thực hiện thao tác cập nhật lên cơ sở dữ liệu, trả về số dòng được cập nhật. |
| **int** Delete(string query) | Thực hiện thao tác xóa lên cơ sở dữ liệu, trả về số dòng xóa được. |

Các lớp con

Class **MySQLManager**: dành cho cơ sở dữ liệu MySQL.

Class **PosgresManager**: dành cho cơ sở dữ liệu Postgres.

## Lớp dành cho người dùng

### Class ORMManager

Đảm nhận việc liên kết các class do người dùng định nghĩa với thông tin từ cơ sở dữ liệu, ngoài ra cung cấp các phương thức để người dùng thao tác trên thông tin đó.

Các thuộc tính

|  |  |
| --- | --- |
| Dictionary<string, string> featureMap | Dictionary map từ tên cột và thuộc tính của class. |
| **List**<**string**> primaryKeys | Danh sách các khóa chính. |
| **List**<**string**> autoincrementCols | Danh sách các cột tự tăng |
| **string** tableName | Tên bảng |
| **DBManager** dbManager | DBManager để kết nối với CSDL. |
| **Parser** parser | Parser tương ứng với CSDL. |

Các phương thức

|  |  |
| --- | --- |
| **Object** GetValue(Object src, string propName) | Trả về giá trị của thuộc tính propName của object src. |
| **InsertQuery** Insert(T obj) | Trả về InsertQuery để chèn object T. |
| **DeleteQuery** Delete(T obj) | Trả về DeleteQuery để xóa object T. |
| **DeleteQuery** Delete(Condition condition) | Trả về DeleteQuery để xóa các dòng thỏa condition. |
| **UpdateQuery** Update(T obj) | Trả về UpdateQuery, với dữ liệu mới là object T. |
| **UpdateQuery** Update() | Trả về UpdateQuery để lấy dữ liệu mới nhất từ cơ sở dữ liệu. |
| **SelectQuery<T>** Select() | Trả về SelectQuery. |

## Các lớp hỗ trợ truy vấn

### Class ExecutableQuery

Là lớp ảo, là lớp cha cho các câu query có thay đổi đến CSDL như Insert, Delete, Update.

Các thuộc tính

|  |  |
| --- | --- |
| **string** tableName | Tên bảng. |
| **DBManager** dbManager | DBManager để kết nối với CSDL. |
| **Parser** parser | Parser tương ứng với CSDL. |
| **Dictionary**<**string**, **string**> featureMap | Dictionary map từ tên cột và thuộc tính của class. |

Các phương thức

|  |  |
| --- | --- |
| **int** Execute() | Hàm dùng để chuyển ExecutableQuery sang string (nhờ parser) rồi truyền vào dbManager để thực thi. Kết quả trả về là số dòng thực hiện thành công. |

Các lớp con

class **UpdateQuery**

class **DeleteQuery**

class **InsertQuery**

### Class SelectQuery<T>:

Các thuộc tính

|  |  |
| --- | --- |
| **int** ASC = 0 |  |
| **int** DESC = 1 |  |
| **string** tableName | Tên bảng của T |
| **DBManager** dbManage | DBManger tương ứng với CSDL |
| **Parser** parser | Parser tương ứng với CSDL |
| **Dictionary**<**string**, **string**>  Projections | Dictionary các phép chiếu |
| **Dictionary**<**string**, **string**> aliasMap | Tên các thuộc tính map |
| **Dictionary**<**string**, **int**> orderBy | Dictionary các trường sắp xếp |
| **Dictionary**<**string**, **string**> featureMap | Dictionary tên thuộc tính và tên cột. |
| **Disjunction** condition | Điều kiện. |

Các phương thức

**new** **GroupByQuery**<**T**> Where(Condition condition)

**GroupByQuery**<**T**> AddProjection(string attr, string aggFunc = "", string alias = "")

new **GroupByQuery<T>** GroupBy(string attr)

new **GroupByQuery<T>** OrderBy(string attr, string order = "ASC")

**GroupByQuery<T>** Having(Condition condition)

### Class Include

Chọn các phần tử liên quan theo quan hệ 1-m. Ví dụ: 1 lớp có nhiều học sinh.

Class này cũng sử dụng mẫu decorator, mở rộng class SelectQuery như GroupByQuery.

Các thuộc tính

|  |  |
| --- | --- |
| **SelectQuery<T>** wrappee | SelectQuery gốc. |
| **Type** included | Kiểu của object được T sở hữu. |
| **string** forgeinKey | Khóa ngoại của T. |
| **string** otherTable | Tên bảng của included |
| **string** insideKey | Tên khóa của included được trỏ tới. |
| **PropertyInfo** hasManyAttr | thông tin quan hệ. |
| **Dictionary**<**string**, **string**> aliasMap | Dictionary từ cột sang tên biến. |
| **Dictionary**<**string**, **string**> featureMap | Dictionary từ tên biến sang cột. |
| **string**[] projection | Các thuộc tính muốn lấy. |

### Class Aggregate

Cass này chứa các chuỗi tên hàm như: MIN, MAX, COUNT, SUM, …

## Các lớp điều kiện

### Class Condition

Là lớp ảo, đóng vai trò như một nhà máy (thông qua các hàm static).

Các phương thức

toSQL(): trả về chuỗi điều kiện.

Các hàm tạo các class con như:

* GreaterThan(string a, value b): tạo điều kiện a > b.
* LessThan(string a, value b): tạo điều kiện a < b.
* And (Conditon a, Conditon b): tạo điều kiện a and b
* ….

### Class Not

Là lớp con của Condition. Thể hiện not của condition, sử dụng mẫu decorator.

### Class Connector

Là lớp con của Condition. Là lớp ảo, lưu danh sách các Condition được nối với nhau bằng AND hoặc OR.

Các thuộc tính

**List**<**Condition**> conditions: danh sách các condition.

Các phương thức

Add(Condition a): thêm điều kiện mới

Các lớp con

Class **Conjunction**

Class **Disjunction**

### Class Restriciton

Là lớp ảo, đóng nhiệm vụ thể hiện các điều kiện hai ngôi.

Các thuộc tính

**string** param: vế trái (biến).

**object** value: vế phải (giá trị).

Các lớp con

|  |  |
| --- | --- |
| Class **Equal** | Class **LessThan** |
| Class **GreaterThan** | Class **LessThanOrEqual** |
| Class **GreaterThanOrEqual** | Class **Like** |

# Các mẫu thiết kế hướng đối tương được áp dụng

## Mẫu Abstract Factory

### Sơ đồ lớp



### Ý nghĩa

Người dùng muốn thao tác được với cơ sở dữ liệu thì cần cung cấp 2 thành phần là DBManager và Parser. 2 thành phần này cần có cùng loại với nhau nên ta áp dụng mẫu Abstract Factory để đảm bảo người dùng sinh ra các thành phần cùng loại.

### Sử dụng

DBFactory factory = new MySQLFactory();

DBManager dbManager = factory.CreateDBManager("localhost", 3306, "school", "root", "1234");

Parser parser = factory.CreateParser();

## Mẫu Composite

### Sơ đồ lớp



### Ý nghĩa

Các lớp Connector (Conjunction và Disjunction) để liên kết các điều kiện cần phải chứa nhiều Condition nên ta có thể sử dụng mẫu Composite ở đây.

### Sử dụng

Disjunction c = new Disjunction();

c.Add(Condition.Equal("ten", "abc"));

c.Add(Condition.LessThan("lop", 3));

## Mẫu Template Method

### Sơ đồ lớp



Phương thức toSQL() của các lớp con của Connector có rất nhiều điểm chung do đó ta sẽ áp dụng mẫu Template Method để đưa phần chung đó lên lớp cha. Mẫu này cũng được áp dụng cho hàm toSQL() của lớp Restriction (các toán tử so sánh 2 ngôi).

Hàm template

public override string toSQL(Dictionary<string, string> featureMap)

{

    string opt = getLogic();

    if (conditions.Count == 0)

    {

        return "";

    }

    string res = conditions[0].toSQL(featureMap);

    for (int i = 1; i<conditions.Count; i++)

    {

        res += " " + opt + " " + conditions[i].toSQL(featureMap);

    }

    res = "(" + res + ")";

    return res;

}

Các lớp Conjunction và Disjunction sẽ override lại phương thứ getLogic():

class Conjunction:Connector

{

    public override string getLogic()

    {

        return "AND";

    }

}

## Mẫu Decorator

### Sơ đồ lớp



### Ý nghĩa

Ở đây GroupByQuery là một trường hợp đặc biệt của SelectQuery nên ta có thể sử dụng mẫu Decorator để thêm tính năng GroupBy cho câu lệnh Select lõi. Hơn nữa ta cũng giới hạn câu lệnh Select sẽ không có phương thức Having và không có các hàm Aggregate như câu lệnh GroupBy.

### Sử dụng

List<Object> res = orm.Select().AddProjection("ten") .GroupBy("lop").AddProjection("ten", Aggregate.COUNT, "demten").Having(Condition.LessThan("ten",41,Aggregate.COUNT)).ToList();

## Mẫu Strategy

### Sơ đồ lớp



### Ý nghĩa

Lớp ORMManager nhận vào đối tượng DBManager và Parser có thể thay đổi theo loại cơ sở dữ liệu mà người dùng sử dụng. Do đó ta sử dụng mẫu Strategy để định nghĩa một interface chung cho các đối tượng đó giúp việc thao tác được linh hoạt.

### Sử dụng

DBFactory factory = new MySQLFactory();

MySQLManager dbManager = factory.CreateDBManager("localhost", 3306, "school", "root", "palo1234") as MySQLManager;

MySQLParser parser = factory.CreateParser() as MySQLParser;

ORMManager<Student> orm = new ORMManager<Student>(dbManager, parser);