数据访问接口设计

目录

1 需求分析

背景：

数据访问层接口众多且不统一，每次业务增加，需要增加新的接口，开发效率低下，需要对接口进行统一抽象，形成理想情况下的增删改查接口。

需求：

1．封装一套数据增删改查的接口，使数据访问层实现通过调用一系列接口完成相应的处理逻辑

2．通过配置文件指定处理请求需要调用的接口序列。尽量实现通过添加配置项，完成新响应请求的添加。

2 名词解释

3 功能设计

3.1 设计方案

3.1.1 使用方式

3.1.1.1 上层调用

需要与数据访问层交互的模块，在开发新的功能时，首先需要明确请求的类型(增删改查)，然后使用对应的请求组包接口完成数据包的组装，上层模块只需要根据请求的逻辑，填入正确的数据即可，无需了解数据包的结构。模块收到数据访问层的响应后同样调用解包

接口解析数据包，得到自己想要的数据。

3.1.1.2 数据访问层配置

数据访问层要响应新的请求，只需要更新请求响应配置文件，配置文件可以针对每个请求配置如何请求数据包、如何处理请求和如何组织响应数据包。配置完成后，只需要热重启数据访问层，使其重新装载配置文件，数据访问层就可以响应新的请求了。

3.1.2 数据结构

3.1.2.1 类型枚举

数据库支持的数据类型，上层调用时组包时需要填写数据类型，定义一下枚举方便组包时使用。

```
enum DB_TYPE
{
    DB_INT= 0,
    DB_LONG_LONG = 1，
    …
};
```

3.1.3 上层调用接口

数据访问层向上层提供 4 个接口，上层通过调用接口完成对数据库的增删改查操作，实际应用中，上层首先要明确要对数据库做的操作，然后根据接口填充值，发送请求，并通过解析返回包得到想要的数据，或得知请求结果，完成一次请求的调用。

3.1.3.1 Query 接口

Query 描述一个查询请求，支持批量查询(多个用户 uid)。查询结构为变长结构，可以包含多个查询条件，一个条件对应表的一列（由配置文件指定列），查询条件由接口体 queryValue 描述。

//单个查询条件数据结构

//查询结果数据结构

uid 为发送请求的用户 id，主要用于记录日志，方便定位问题，后面插入，更新，删除结构中的 uid 字段都是这个作用。conditionCount 和 condition 分别指查询条件个数和查询条件。通过这两个字段可以表示查询条件，各条件之间的关系由配置文件指定。id 字段可以使 uid 等，id 的每个值分别与 condition 按照“与”的关系组成查询条件，如果不是批量请求，可以只填一个 id，或将 id 直接写到 condition 中。

//请求包

查询请求的回包，主要包含 rowCount 和 colCount 和 data。上层调用可以通过计算得知数据数量（rowCount*colCount）。通过对 dataValue 的解析可以得到每个 data 的数据。

//回包

3.1.3.2 Insert 接口

Insert 描述一个插入请求，insert 结构为变长结构，可以包含多个要插入的值，一个值对应表的一列（由配置文件指定列），值由接口体 newValue 描述。

//单个列数据结构

插入请求主要有 valueCount 和 vaule 组成，valueCount 表名请求中值的个数，value 为值的数组，配置文件中插入请求列的配置顺序必须也 value 的顺序一致。

//请求包

插入请求的回包主要向调用返回处理的结果

//回包

;

3.1.3.3 Update 接口

更新请求的结构需要包含查询条件和新的值两项相当于查询接口和插入接口的组合。

更新请求的回包，同样要向上层返回请求处理结果

//回包

3.1.3.4 Delete 接口

删除接口，与查询接口类似，只是不支持批量操作。

//请求包

返回删除操作的处理结果

//回包

3.1.4 组包和解包接口

3.1.2 中定义的数据包大部分为变长，组包和解包的代码比较复杂，对上层调用来说编码比较麻烦。为了简化上层编码，提高开发效率，本设计封装了多个类，分别负责组装发送包和解析数据访问层返回的数据包。针对增删该查接口定义分别定义 QueryBuildPacket，QueryParsePacket ， InsertBuildPacket ， InsertParsePacket ， UpdateBuildPacket ，UpdateParsePacket、DeleteBuildPacket 和 DeleteParsePacket 几个类负责组包和解包工作。

3.1.4.1 QueryBuildPacket—查询请求组包类

接口功功能说明

序号	接口	说明
1	QueryBuildPacket	构造函数，根据默认值分配内存
2	~ QueryBuildPacket	析构函数，释放内存
3	SetRequestUid	设置请求 uid
4	getRequestUid	得到请求 id
5	SetRequestCode	设置请求命令码
6	GetRequestCode	得到请求命令码
7	AddId	添加 id
8	AddCondition	添加条件
9	GetPacket	构造并返回发送包，同时返回包的长度
10	reset	清空数据，但不释放内存，使内存可以重用

3.1.4.2 QueryParsePacket—查询返回包解析类

接口功功能说明

序号	接口	说明
1	QueryParsePacket	构造函数
2	QueryParsePacket(void* packet, uint32_t size)	参数指定数据包和大小，构造对象的同时解析包
3	~ QueryParsePacket	析构函数，释放内存
4	getValue(uint32_t row, uint32_t col, uint32_t& outsize, uint32_t outType)	根据行 ,列号得到指定位置的数据并返回数据大小和类型
5	getRowCount	得到行数
6	getColCount	得到列数
7	getReslut	得到处理结果

3.1.4.3 InsertBuildPacket—插入请求组包类

接口功功能说明

序号	接口	说明
1	InsertBuildPacket	构造函数，根据默认值分配内存
2	~ InsertBuildPacket	析构函数，释放内存
3	SetRequestUid	设置请求 uid
4	getRequestUid	得到请求 id
5	SetRequestCode	设置请求命令码

6	GetRequestCode	得到请求命令码
7	AddInsertValue	添加插入值
8	GetPacket	构造并返回发送包，同时返回包的长度
9	reset	清空数据，但不释放内存

3.1.4.4 InsertParsePacket—插入返回包解析类

接口功能说明

序号	接口	说明
1	InsertParsePacket	构造函数
2	InsertParsePacket (void* packet, uint32_t size)	参数指定数据包和大小，构造对象的同时解析包
3	getResult	得到处理结果

3.1.4.5 UpdateBuildPacket—更新请求组包类

public 接口功功能说明

序号	接口	说明
1	UpdatetBuildPacket	构造函数，根据默认值分配内存
2	~UpdateBuildPacket	析构函数，释放内存
3	SetRequestUid	设置请求 uid
4	getRequestUid	得到请求 id
5	SetRequestCode	设置请求命令码
6	GetRequestCode	得到请求命令码

7	AddUpdateValue	添加更新值
8	AddCondition	添加条件
9	GetPacket	构造并返回发送包，同时返回包的长度
10	reset	清空数据，但不释放内存

3.1.4.6 UpdateParsePacket—更新返回包解析类

接口功能说明

序号	接口	说明
1	UpdateParsePacket	构造函数
2	UpdateParsePacket (void* packet, uint32_t size)	参数指定数据包和大小，构造对象的同时解析包
3	getAffectedRowCount	得到受影响的行数
4	getReslut()	得到处理结果

3.1.4.7 DeleteBuildPacket—删除请求组包类

public 接口功功能说明

序号	接口	说明
1	DeleteBuildPacket	构造函数，根据默认值分配内存
2	~ DeleteBuildPacket	析构函数，释放内存
3	SetRequestUid	设置请求 uid
4	getRequestUid	得到请求 id
5	SetRequestCode	设置请求命令码
6	GetRequestCode	得到请求命令码

7	AddCondition	添加条件
8	GetPacket	构造并返回发送包，同时返回包的长度
9	reset	清空数据，但不释放内存，使内存可以重用

3.1.4.8 DeleteParsePacket—更新返回包解析类

接口功能说明

序号	接口	说明
1	DeleteParsePacket	构造函数
2	DeleteParsePacket (void* packet, uint32_t size)	参数指定数据包和大小，构造对象的同时解析包
3	getAffectedRowCount	得到受影响的行数
4	getReslut()	得到处理结果

3.1.5 数据库操作接口

数据访问层接到上层请求后，通过请求的内容，以及配置文件的配置，调用一个或多个数据库操作接口，完成上层请求的处理，并返回。数据库操作接口属于数据访问层内部接口，对上层调用不可见。

3.1.5.1 查询

函数功能：执行查询，返回查询结果。

参数

db：数据库名

table：表名

condition：查询条件

3.1.5.2 位查询

函数功能：执行查询，并返回保存开关量列指定位的开关状态

参数

db：数据库名

table：表名

condition：查询条件

columnName：保存开关量的列

bitNo：要返回的开关量所在列的位。

说明：位查询接口实现对于开关量的操作，例如查询用户是否绑定手机。接口通过 columnName，和 bitNo 指定开关量所在的字段以及在字段中的位置，如果开关打开，返回 true，否则返回 false。

3.1.5.3 插入

函数功能：执行插入操作，通过参数指定数据库和表。

参数

db：数据库

tableName：表名

insertVaule：插入值

注意：insertVaule 是通过解析请求 struct 和配置文件内容，构造的，所以配置文件中“newValue”配置项中的列顺序一定要与请求 struct 中的值的意义对应，否则会导致数据库中数据错误。

3.1.5.4 更新

函数功能：更新数据库中符合条件的记录。

参数

db：数据库

tableName：表名

condition：查询条件

insertVaule：新值

说明：更新操作需要包含查询条件以及要更新的值，所以配置文件中的查询条件和值两项都需要配置

3.1.5.5 位更新

函数功能：更新数据库中符合条件的记录的指定开关量状态

参数：

db：数据库

tableName：表名

condition：查询条件

columnName：列名

bitNo：开关量位号

value：要更新的值 true/false

3.1.5.6 删除

函数功能：删除符合条件的记录

参数

db：数据库

tableName：表名

condition：查询条件

3.1.6 配置文件

3.1.6.1 作用及设计思路

通过配置项指定某个请求要做的操作和操作的对象，另外还要考虑一个请求包含多个操作的情况。

需要配置的项包括：

- 库：要连接的数据库
- 表：要操作的表
- 操作：对表进行的操作

- 查询条件：将查询 struct 中的值与表的列对应。插入操作可以没有该项
- 查询条件的关系：and 或者 or，插入操作可以没有该项
- 新值：将更新或插入 struct 中的值与表的列对应，查询操作可以没有该项
- 分库分表支持：

- 排序规则支持：
- 分页支持：limit/offset 放在请求包里，不配置
- 返回内容：对于查询，配置结果中的那些列需要返回，对于只需要返回成功或失败的操作，可以没有该项

3.1.6.2 配置文件内容

```
<requests>
 <request ID=6666>
  <reqNo value="0X20190112"/>
  <instance value= "login_MySQL ' />
  <db value= “db_user" />
  <table value= “t_user” />
  <splitcolumn value=” uid” />     <!—支持分库分表-->
  <splitvalue value= "1024"/>        <!—splitcolumn value % splitvalue -->

  <op value=” 1” >   <!—Query -->
    <condition>
      <column>
        <name "XXXX"/>
        <rel X/>
      </column >
```

```
    <column>
      <name "XXXX"/>
      <rel X/>
    </column >
      <column>
        <name "XXXX"/>
    </ column >
</condition >
<newvalue>
    <column "XXXX"/>
    <column "XXXX"/>
</newvalue >
<queryresult>
    <MaxCount XXX/>
    <column "XXXX"/>
    <column "XXXX"/>
</queryresult >
 <sort>   <!—支持分页和排序，分页放在请求包里 -->
      <column>
          <name value= "login_time" />
           <type value= “DESC” />
      </column>
```

```
        </sort>
      </op>
      <op>
      </op>
    </request>
  </requests>
```

4 示例代码

4.1 查询请求

以拉取 IM 离线消息为例，介绍上层如何使用数据访问层提供的查询接口完成离线消息拉取请求包的构造以及返回数据的解析。

4.2 插入请求

下面以添加好友为例，介绍上层如何使用数据访问层提供的插入接口完成添加好友请求包的构造以及返回数据的解析。