# 摘　　要

ACM/ICPC 程序设计竞赛是世界上公认的规模最大、水平最高的国际大学生程序设计竞赛。可分布式部署的在线评测系统作为辅助竞赛者的日常培训的应用软件，必须不断的适应竞赛者的训练要求。只有一个能够不断适应竞赛训练的发展的在线评测系统才是一个成功的系统设计。

可分布式部署的在线评测系统是一个综合 B/S 与 C/S 模式，结合 Linux 内核编程的综合性可分布式系统应用。本系统不仅支持传统在线评测系统的动态 Web（B/S 模式）的访问，同时支持使用客户端（C/S 模式）访问。可分布式部署的在线评测系统分为 Web 服务模块、中心服务模块、评测模块、数据库服务模块、用户终端模块五个大的功能部件。根据各个部件所属的位置可视为三层结构：网络交互层，数据控制层和内核层。属于网络交互层的有 Web 服务模块和用户终端模块。属于数据控制层的有中心服务模块。属于内核层的有评测模块和数据库服务模块。数据接口模块作为中心服务模块的一个重要子系统，控制着中心服务模块的所有数据的存储与查询。

本文首先介绍了系统使用的一些技术与知识背景，数据接口的需求与分析和数据库的设计实现，再从整个系统到中心服务模块的框架出发，阐述了数据接口模块在中心服务模块的重要位置，然后给出了数据接口模块的设计到实现的整个过程描述，最后说明数据接口模块在设计实现中使用的一些技术和设计模式。

**关键字：** ACM/ICPC；程序设计竞赛；分布式；网上评测系统；数据接口

# Abstract

ACM/ICPC is recognized as the largest scale and highest level global college student programming contest. As a ACM/ICPC daily training supply system, the Distributed Online Judge System must constantly satisfy the requirements of the player's training. Only one system which can constantly satisfy the development requirements is a successful system.

The Distributed Online Judge System is a comprehensive distributed system, which use both B/S and C/S mode and combine the Linux kernel programming. Player not only can use the dynamic Web (B/S mode) page, but also can use the client (C/S mode), to access the system. The Distributed Online Judge System is composed of five components which is the Web Module, the Central Service Module, the Judge Module, the Database Module, and terminal. According to the function of the five components, the Distributed Online Judge System is divided into three layers: the Network Service Layer, the Data Control Layer, and the Core Layer. The Web Module and terminal is on the Network Service Layer. On the Data Control Layer is the Central Service Module and On the Core Layer is the Database Module and terminal. The Data Interface Module is an important subsystem of the Central Service Module. And it controls all the data operation for the Central Service Module.

At first, this thesis gives a brief introduction of technology and backgrounds this system used, the requirement analysis of the Data Interface Module and the database design. Then this thesis explains the framework for the whole system and the Central Service Module, expatiate the importance of the Data Interface Module for the Central Service Module. Afterward describe the design of the Data Interface Module. In the end, this thesis show some technology used in the design of Data Interface Module.

**Keywords:** ACM/ICPC; Programming Competition; Distributed; Online Judge System; Data Interface

# 目　录

# 第1章　　引言

## 1.1 课题背景及意义

ACM/ICPC（ACM International Collegiate Programming Contest，国际大学生程序设计竞赛）是由国际计算机界历史悠久、颇具权威性的组织 ACM（Association for Computing Machinery，国际计算机协会）主办的，世界上公认的规模最大、水平最高的国际大学生程序设计竞赛，其目的旨在使大学生运用计算机来充分展示自己分析问题和解决问题的能力。

在线评测系统（Online Judge System，OJ）作为国内外诸多参加程序设计竞赛选手的训练交流平台，对每个由评测系统展示的题目接收用户提交的源代码，在编译执行后将用户程序的输出与系统的标准输出进行比较，并给出用户程序运行和比较的结果，从而判断用户提交的源代码是否正确。目前国内外已有类似网上评测系统，其中世界最著名的网上评测系统是 UVA Online Judge，国内著名的网上评测系统有 PKU Online Judge（北京大学）和 ZJU Online Judge（浙江大学）。

武汉大学 ACM/ICPC 在线评测系统（Online Judge System）建设的目的正是为了培养学生的实际动手能力。其功能有在线测试（主要是用户通过网络直接检测自己编写的程序是否正确并提供在线比赛的平台）和论坛（提供用户在论坛上探讨问题，互相学习的机会）。通过此网站不但能让同学们对 ACM/ICPC 有所了解，而且还增添他们编程的兴趣。

然而随着参加 ACM/ICPC 的同学越来越多，原始的单服务器模式越来越不能承受多用户同时访问时的压力，表现在 Web 的响应速度无法跟上用户的大面积访问，以及判题服务的响应时间慢于用户提交新程序的时间，从而进入无法提供服务给新登录用户，且已登录用户的服务也会变得无法响应的恶性循环。近两年国内 ACM/ICPC 举行的亚洲区各赛点的网络预选赛充分暴露了这一问题。且在原有在线评测系统的运行中，暴露了很多其设计缺陷与不足，如一个题目只能对应一个比赛，Rejudge 反应不正常，讨论区不人性化等等。可分布式部署的在线评测系统将在武汉大学已有的在线评测系统上进行全面重构，将原始的单服务器模式转变成由一个底层数据库服务模块（本模块使用 MySQL），一个中心服务模块，多个 Web 服务模块，多个评测判题服务模块组成的服务集群，集群中的任何一个部件均可部署在不同的服务器上，且可以用普通 PC 临时替代，大大减轻服务器压力和提升服务性能。在提供服务集群的同时本课题还提供了评测系统的客户端软件，从而在 B/S 架构的同时也实现了 C/S 架构，让用户在网络不稳定的情况下也能一样进行练习。为了控制可分布式部署的在线评测系统数量众多，类型繁杂的数据，加快繁忙时系统的响应速度，一个独立的设计完善的数据接口模块和一个设计合理的数据库显得尤为重要。

## 1.2 课题需解决的问题

### 1.2.1 可分布式部署的实现

本课题所涉及的系统设计的原因之一就是解决单服务器模式在用户的大面积访问时的网络负载过重的问题。解决此问题一个可行的措施是设计一个可以分布式部署解决方案。多个 Web 服务入口以及额外的客户端访问入口，可以使在线评测服务的网络负载分散开了，

从而减轻单个服务器的负担，使在用户大面积访问时系统仍然能够正常相应用户的请求。

### 1.2.2 数据库与文件系统中的数据操作

本课题所涉及的系统的数据内容数量众多，数据的类型繁杂，因此不仅需要数据库的支持，同时需要管理文件系统中的数据文件。为了应付这些复杂数据管理操作，迫切需要创建一个统一的数据接口来支持数据的操作。通过设计一个统一的数据接口，不仅能使数据的管理变得方便，而且使得其他模块更改数据内容更加容易。

### 1.2.3 缓存加速数据读取速度的实现

在系统运行过程中，有时候会出现大量的用户都会连接本系统的情况，尤其是在系统举行比赛的期间，造成系统大量的数据查询，导致系统的性能持续下降。为了防止系统的性能下降，合理的选择是在系统中加入缓存对数据查询的加速。利用缓存进行加速，数据只有第一次需要访问数据库或者文件系统，并不需要每次查询都需要访问数据库或者文件系统，这样便节省了大量访问数据库或者文件系统的时间，从而使得整个系统的性能得到提升。

## 1.3 课题取得的成果

本课题所实现的可分布式部署在线评测系统目前已在实验室内网络临时服务集群上稳定运行超过两个星期，为整个武汉大学 ACM/ICPC 集训队提供了练习、比赛的平台。数据接口模块作为中心服务程序（可分布式部署的在线评测系统的核心控制部件）的一个重要子系统，能够正常的提供数据服务，其缓存加速功能也能够稳定的运行。

## 1.4 本文的组织

因为在线评测系统本身是一个比较复杂的系统，涉及的背景知识也会相对复杂，实现过程中所使用的方法和实现的内容也相对繁琐，加上本系统要求可分布式部署，致使本文覆盖的内容和知识体系不可避免的会比较复杂。为简化阅读，现将整篇论文的结构介绍如下：

整篇论文共分六章及附录两节，各章节的主要内容安排如下：

1．引言。介绍论文选题的课题背景及意义、需要解决的问题、实际上取得的成果及论文的结构安排。

2．背景知识。简单讨论一下所用的技术的背景知识，包括 Mysql 数据库系统及其编程，设计模式，Linux 与 Linux 下的多线程编程等。

3．数据接口模块设计与实现。对数据接口模块的需求，分析，设计。给出数据接口的设计图，并用流程图说明一些比较复杂的模块。

4．数据安全与防范措施。讨论数据模块可以防范的攻击，与数据的安全。

5．结果分析。讨论数据接口模块的所使用的一些特殊方法。

6．结束语。对数据接口模块的功能分析，以及不足的概述。

# 第2章　背景知识

## 2.1 MySQL 数据库系统

MySQL 是一个小型关系型数据库管理系统，开发者为瑞典 MySQL AB 公司。目前 MySQL 被广泛地应用在 Internet 上的中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，许多中小型网站为了降低网站总体拥有成本而选择了 MySQL 作为网站数据库。MySQL 具有以下特性：

1．使用 C 和 C++编写，并使用了多种编译器进行测试，保证源代码的可移植性

2．支持 AIX、FreeBSD、HP-UX、Linux、Mac OS、Novell Netware、OpenBSD、OS/2 Wrap、Solaris、Windows 等多种操作系统

3．为多种编程语言提供了 API。这些编程语言包括 C、C++、Eiffel、Java、Perl、PHP、Python、Ruby 和 Tcl 等。

4．支持多线程，充分利用 CPU 资源。

5．优化的 SQL 查询算法，有效地提高查询速度。

6．既能够作为一个单独的应用程序应用在客户端服务器网络环境中，也能够作为一个库而嵌入到其他的软件中提供多语言支持，常见的编码如中文的 GB 2312、BIG5，日文的 Shift_JIS 等都可以用作数据表名和数据列名。

7．提供 TCP/IP、ODBC 和 JDBC 等多种数据库连接途径。

8．提供用于管理、检查、优化数据库操作的管理工具。

9．可以处理拥有上千万条记录的大型数据库。

## 2.2 设计模式

设计模式是一套被反复使用，多数人知晓的，经过分类编目的，面对对象软件的设计经验的总结。设计模式使人物可以更加简单方便地复用成功的设计与体系结构，帮助你做出有利于系统复用的选择，避免设计损害了系统的复用性。

一般而言，一个设计模式有四个基本要素：

**1．模式名称。**一个助记名，它用一两个词来描述模式的问题，解决方案和效果。命名一个新的模式增加了设计词汇。设计模式允许在较高的抽象层次上进行设计。模式名可以帮助思考，便于与其他人交流设计思想及设计结果。找到恰当的模式名也是设计模式编名工作的难点之一。

**2．问题。**描述了应该在何时使用模式。它解释了设计问题和问题存在的前因后果。它可能描述了特定的设计问题，也可能描述了导致不灵活设计的类或对象结构。有时候，问题部分会包括使用模式必须满足的一系列先决条件。

**3．解决方案。**描述了设计的组成成分，他们之间的相互关系及各自的职责和协作方式。因为模式就像一个模板，可应用与多种不同场合，所以解决方案并不描述一个特定而具体的设计或实现，而是提供设计问题的抽象描述和怎样用一个具有一般意义的元素组合来解决问题。

**4．效果。**描述了模式应用的效果及使用模式应权衡的问题。尽管描述设计决策时，并不总提到模式效果，但他们对于评价设计选择和理解使用模式的代价及好处具有重要意义。

一般而言，效果包括它对系统的灵活性，扩充性或可移植性的影响。

常用的设计模式共有 23 种，按照其目的进行分类可以分为创建型（Creational）、结构型（Structural）、和行为型（Behavioral）三种。其中创建型包括抽象工厂（Abstract Factory）、生成器（Builder）、工厂方法（Factory Method）、原型（ProtoType）和单件（Singleton）共五种。结构型包括适配器（Adapter）、桥接（Bridge）、组成（Composite）、装饰（Decorator）、外观（Facade）、享元（Flyweight）和代理（Proxy）共七种。行为型包括职责链（Chain of Responsibility）、命令（Command）、解释器（Interpreter）、迭代器（Iterator）、中介者（Mediator）、备忘录（Memento）、观察者（Observer）、状态（State）、策略（Startegy）、模板方法（Template Method）和访问者（Vistor）共 11 种。

## 2.3 Linux 与 Linux 下的编程

### 2.3.1 Linux 简介

Linux 操作系统，是一种计算机操作系统。Linux 操作系统的内核的名字也是“Linux”。Linux 操作系统也是自由软件和开放源代码发展中最著名的例子。

严格来讲，Linux 这个词本身只表示 Linux 内核，但在实际上人们已经习惯了用 Linux 来形容整个基于 Linux 内核，并且搭配了程各种人机界面、应用和服务软件的操作系统（也被称为 GNU/Linux）。基于这些组件的 Linux 软件被称为 Linux 发行版。一般来讲，一个 Linux 发行套件包含大量的软件，比如软件开发工具，数据库，Web 服务器（例如 Apache），X Window，桌面环境（比如 GNOME 和 KDE），办公套件（比如 OpenOffice.org），等等。

Linux 内核最初是为英特尔 386 微处理器设计的。现在 Linux 内核支持从个人电脑到大型主机甚至包括嵌入式系统在内的各种硬件设备。

在开始的时候，Linux 只是个人狂热爱好的一种产物。但是现在，Linux 已经成为了一种受到广泛关注和支持的一种操作系统。包括 IBM 和惠普在内的一些计算机业巨头也开始支持 Linux。很多人认为，和其他的商用 Unix 系统以及微软 Windows 相比，作为自由软件的 Linux 具有低成本，安全性高，更加可信赖的优势。

Linux 是一套免费使用和自由传播的类 Unix 操作系统，这个系统是由全世界各地的成千上万的程序员设计和实现的。它以高效性和灵活性著称。并且能够在 PC 计算机上实现全部的 Unix 特性，具有多任务、多用户的能力。Linux 之所以受到广大计算机爱好者的喜爱，主要原因有两个，一是它属于自由软件，用户不用支付任何费用就可以获取它的源代码，并且可以根据自己的需要对它进行必要的修改。另一个原因是，它具有 Unix 的全部功能。Linux 是一个多用户多任务的分时操作系统，每个用户都可同时执行多个进程，系统中的进程数目在逻辑上不受限制。

### 2.3.2 Linux 环境下的编程

线程（thread）技术早在 60 年代就被提出，但真正应用多线程到操作系统中去，是在 80 年代中期，solaris 是这方面的佼佼者。传统的 Unix 也支持线程的概念，但是在一个进程（process）中只允许有一个线程，这样多线程就意味着多进程。

Linux 系统下的多线程遵循 POSIX 线程接口，称为 pthread。编写 Linux 下的多线程程序，需要使用头文件 pthread.h，连接时需要使用库 libpthread.a。pthread 提供了创建多线程程序的接口，与线程同步的解决方案（互斥锁）。

MySQL 提供了 Linux 下的开发库 mysqlclient。编写操作 MySQL 的程序需要使用头文件 mysql.h，连接时需要使用库 mysqlclient.a。mysqlclient 提供了连接 MySQL，执行 SQL 语句，获取 SQL 执行结果等接口。

# 第3章　数据接口模块设计与实现

## 3.1 需求分析

本系统需要保存的数据内容，即数据接口模块需要支持的数据内容，如下：

**1．题目**

每个题目需要有题号（ID），标题（Title），总时间限制（Time Limit），单个测试文件时间限制（Case Time Limit），总内存限制（Memory Limit），题目描述（Description），输入描述（Input），输出描述（Output），样例输入（Sample Input），样例输出（Sample Output），提示（Hint），来源（Source），相关比赛（Related Contest），输入输出文件（Input/Output File）相关文件（Related Files），是否 Special Judge，Special Judge 文件。其中题号由系统自动生成，题目描述中可以有显示图片，输入输出文件都是成对出现，每一个输入文件对应一个输出文件。题目还需要保存题目的提交数，通过的数目，提交的用户数，通过的用户数。提供根据题号（Problem ID），标题（Title），来源（Source），隶属比赛（Related Contest）等条件的搜索。

每个题目统计信息应包含本题的提交信息统计表和通过的提交表，提交信息统计表包括尝试的用户数（Users Tried），通过的用户数（Users Solved），提交总次数（Submit），通过次数（AC），格式错次数（PE），超内存次数（MLE），超时次数（TLE），答案错次数（WA），超输出次数（OLE），运行错次数（RE），编译错次数（CE）。

每个题目都需要能搜索到所有该题提交的实时状态（Status）的列表，详细见下面[3]中所说。

**2．比赛**

每个比赛需要有比赛号（ID），标题（Title），类型（Type），描述（Description），开始时间（Start Time），结束时间（End Time），关联题目（Related Problems），相关文件（Related Files）。其中比赛号为自动生成，无法修改，标题为字符串，类型分为公开比赛、私有比赛、有奖比赛以及虚拟比赛，开始时间和结束时间为 yyyy-mm-dd hh:mm:s。比赛还需要包含一个题目列表，表示比赛的题目集，并需要对题目列表中的每个题目生成一个比赛中的题目号（In Contest ID）。比赛相关文件类型应该支持图片和普通文件。

对于私有比赛，需要有一个表来保存能够参加该项比赛的用户的列表，只有这些有权限参加该比赛的用户才能参加比赛。

每次比赛都还要有一个比赛排名和一个比赛统计信息。比赛排名包含一个表格， 每行分别是一个用户的信息， 包含排名（Rank），昵称（Nickname），解决题数（Solved），罚时（Penalty），及各题的状态，罚时及各题状态中时间需要精确到秒来判断排名， 各题状态为 xx/y， 其中 xx 为解决该题的时间， y 为提交次数， 当该题没被解决时 xx 显示为 --。比赛统计信息包该场比赛所有题目的统计信息以及综合提交信息。题目的信息包括每个题的各种结果(AC, CE, WA, TLE, RE, MLE, OLE)数，各种语言提交数，总提交数。

**3．实时状态**

实时状态包含提交编号（Run_ID），提交者（User），提交题号（Problem），结果（Result），内存使用（Memory），耗时（Time），编程语言（Language），代码长度（Length），提交时间（Submit Time）。Result 有以下几种：

通过（Accepted ），结果错误（Wrong Answer），格式错误（Presentation Error），编译错误（Compile Error），超时（Time Limit Exceeded），超出内存限制（Memory Limit Exceeded），超出输出限制（Output Limit Exceeded），运行错误——缓冲区溢出或者堆栈溢出等（Runtime Error SigSegv），运行错误——除 0 错误（Runtime Error SigFpe），运行错误——硬件错误（Runtime Error Sigbus），运行错误——异常退出（Runtime Error Sigabrt），Java 运行错误（Runtime Error Java），受限制的函数调用（Restricted Function），系统错误（System Error）。

如果结果是 Compile Error 或者 Rumtime Error Java，需要保存这次提交产生的错误信息，并且能够关联到该实时状态的错误信息。Memory 栏显示单位为 KB，内容是整数值，Time 栏显示单位为 MS，内容是整数值，Length 栏显示单位为 B，内容是整数值，Submit Time 栏显示为 yyyy-mm-dd hh:mm:ss 格式的时间。每一个实时状态都对应一次代码提交，所有代码都需要一一对应到每一个实时状态。编程语言（Language）包含 GCC，G++，Pascal，Java 共四种。

提交的代码需要可以设置是否共享给他人看，提供用用户（User），题目（problem），结果（Result），编程语言（Language）等搜索条件搜索。实时状态需要分为普通用户提交实时状态和系统管理员提交实时状态。

**4．讨论区**

每一个讨论需要有主题，内容，提交时间，相关用户，相关题号和相关比赛号。讨论可以是重新开一个主题，也可以是回复他人的讨论。回复时，需要将该讨论关联上它回复的讨论。允许管理员按讨论的主题删除所有与该主题有关的讨论。提供相关用户（User），相关标题（Title），相关题号（Problem），相关比赛号（Contest）等搜索方式搜索讨论条目。

**5．用户**

每一个用户需要包括用户 ID（User ID），用户密码（Password），确认密码（Confirm Password），昵称（Nickname），学校（School），邮箱（E-mail），默认编程语言（Default Language），最近看过的题目卷号（Volume），注册时间（Register Time），上次登录的时间（Last_login_time），默认共享代码开关（Shared Code），通过的题目数（Solveds）和提交的次数（Submits）等信息。其中 User ID 限定为 16 字符内字母数字组合，判重时不区分大小写，但是显示时区分，密码为 6 字符以上 16 字符以内（含）的字母数字特殊字符组合，区分大小写，昵称为任意字符串，学校为任意字符串，邮箱地址必须为 user@domain。Domain 格式。还需要对每一个用户设置权限级别， 如可以对一个用户赋予管理员权限，也可以对一个赋予用户查看代码权限，还可以 2 种权限都赋予一个用户。每一个用户都能够修改他的基本信息。此外能够提供一个用户的排行榜，要按照解决题目数，提交数等信息进行排序产生用户的排名（Rank）。

**6．邮件**

邮件需要包含发件人 ID（Sender ID），收件人 ID（Receiver ID），邮件标题（Title），邮件正文（Content）以及发出邮件的时间。每个邮件可以别发件人或者收件人删除，所有邮件都有读与未读两种状态。

**7．即时信息（Notice）**

即时信息需要包含信息的内容。

**8．新闻（News）**

新闻需要包含标题，内容，以及发布时间等信息。

**9．链接（Links）**

链接需要包含链接的地址和链接的名字等信息。

## 3.2 数据库设计与实现

本系统使用的数据库系统为 MySQL，可分布式部署的在线评测系统的数据存放于数据库（OnlineJudge）中，包括代码表（codes），用户参加比赛权限表（contestpermission），比赛表（contests），讨论表（discusses），错误信息表（errors），文件表（files），比赛文件表（filestocontests），题目文件表（filestoproblems），输入输出文件关联表（intooutfiles），邮件表（mails），新闻表（news），题目表（problems），比赛题目列表（problemtocontests），实时状态表（statues）和用户表（users）共 15 张表。数据库的 ER 图如图 3-1：

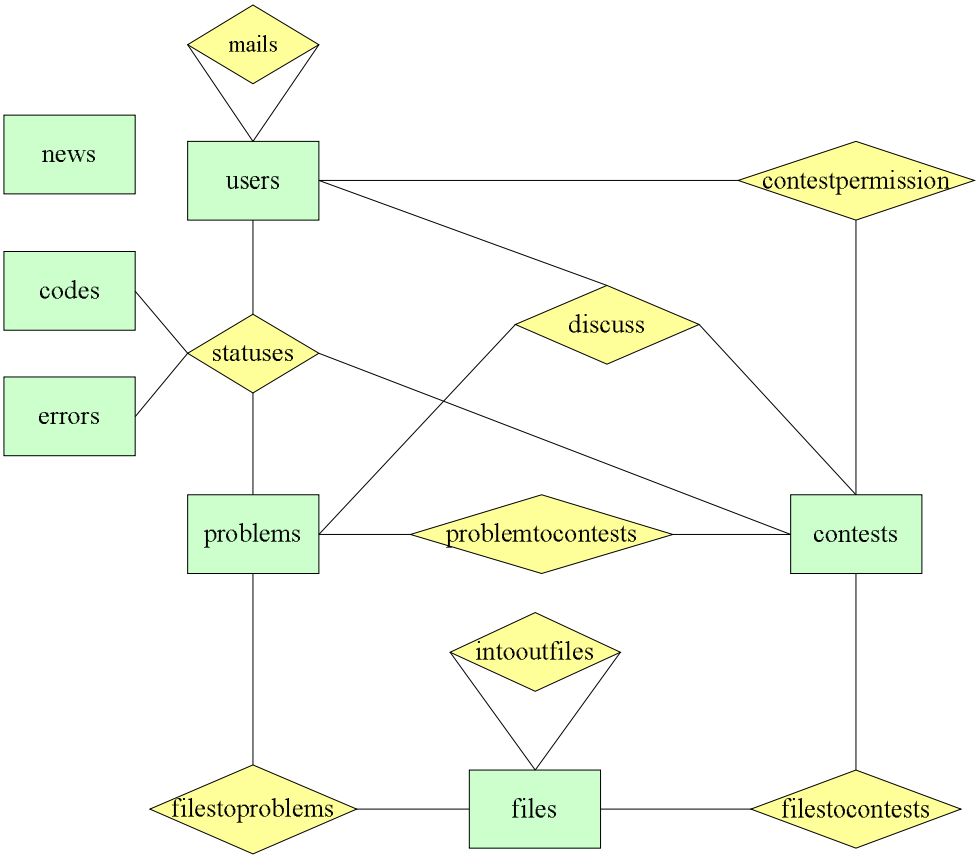


图 3-1 数据库 ER 图

数据库的具体描述如下所示：

1．**代码表（codes）**用于存放用户每次提交的代码，具体代码表信息如下表：

表 3-1 代码表

| 字段名 | 数据类型 | 是否可空 | 默认值 | 字段说明 |
|---|---|---|---|---|
| code_id | int(11) | no | | 代码 ID，主码，自动生成 |
| share | char(1) | yes | Y | 代码是否共享 |
| code_content | text | yes | | 代码内容 |

2．**用户参加比赛权限表**（contestpermission）保存能够参加比赛的用户 ID，具体设计如下表：

表 3-2 用户参加比赛权限表

| 字段名 | 数据类型 | 是否可空 | 默认值 | 字段说明 |
|---|---|---|---|---|
| user_id | varchar(20) | no | | 用户的 ID，主码 |
| contest_id | int(11) | no | | 比赛的 ID，主码 |

3．**比赛表（contests）**保存比赛的基本信息，具体设计如下表：

表 3-3 比赛表

| 字段名 | 数据类型 | 是否可空 | 默认值 | 字段说明 |
|---|---|---|---|---|
| contest_id | int(11) | no | | 比赛 ID，主码 |
| public_id | int(11) | no | 0 | 公开显示的比赛号 |
| title | varchar(200) | no | | 比赛的标题 |
| start_time | datetime | no | | 开始时间 |
| end_time | datetime | no | | 结束时间 |
| contest_type | char(1) | no | N | 比赛类型 |
| description | text | yes | | 比赛的描述 |
| version | int(11) | no | 1 | 比赛的版本号 |
| available | char(1) | no | Y | 比赛是否有效 |

public_id 用于对外显示的非虚拟比赛的比赛号，虚拟比赛显示的比赛号直接使用 contest_id，虚拟比赛的 public_id 为 0。contest_type 可以为 N，P，A 和 V 共 4 种，分别代表公开比赛，私有比赛，有奖比赛和虚拟比赛。

4．**讨论表（discusses）**保存用户的讨论的信息，具体设计如下表：

表 3-4 讨论表

| 字段名 | 数据类型 | 是否可空 | 默认值 | 字段说明 |
|---|---|---|---|---|
| message_id | int(11) | no | | 讨论 ID，主码，自动生成 |
| reply_id | int(11) | no | | 回复的讨论的 ID |
| topic_id | int(11) | no | | 讨论主题 ID |
| user_id | varchar(20) | no | | 用户的 ID |
| problem_id | int(11) | no | 0 | 相关题目的 ID |
| contest_id | int(11) | no | 0 | 相关比赛的 ID |
| title | varchar(200) | yes | | 讨论的条目的标题 |
| content | text | yes | | 讨论的内容 |
| time | datetime | no | | 讨论提交的时间 |
| available | char(1) | no | Y | 讨论是否有效 |

每一个讨论主题都对应一个 topic_id，新的 topic_id 产生于每一个非回复的讨论条目的发表。

5．**错误信息表（errors）**保存提交的代码编译或运行中产生的错误信息，具体设计如下表：

表 3-5 错误信息表

| 字段名 | 数据类型 | 是否可空 | 默认值 | 字段说明 |
|---|---|---|---|---|
| error_id | int(11) | no | | 错误信息 ID，主码，自动生成 |
| content | text | no | | 错误信息的内容 |

6．**文件表（files）**保存系统用到的文件的基本信息，具体设计如下表：

表 3-6 文件表

| 字段名 | 数据类型 | 是否可空 | 默认值 | 字段说明 |
|---|---|---|---|---|
| file_id | int(11) | no | | 文件 ID，主码，自动生成 |
| path | text | no | | 文件的绝对路径 |
| style | tinyint(4) | no | 0 | 文件类型 |
| disable | char(1) | no | N | 文件是否无效 |

style 可以分为 5 种，从 1-5。1 表示题目的输入文件，2 表示题目的输出文件，3 表示图片文件，4 表示题目的 Special Judge 文件，5 表示题目的参考程序文件。

7．**比赛文件表（filestocontests）**保存每个版本的比赛的文件的关联，具体设计如下表：

表 3-7 比赛文件表

| 字段名 | 数据类型 | 是否可空 | 默认值 | 字段说明 |
|---|---|---|---|---|
| file_id | int(11) | no | | 文件的 ID，主码 |
| contest_id | int(11) | no | | 隶属比赛的 ID，主码 |
| version | int(11) | no | 1 | 该文件对应比赛的版本号 |

8．**题目文件表（filestoproblems）**保存每个版本的题目的文件的关联，具体设计如下表：

表 3-8 题目文件表

| 字段名 | 数据类型 | 是否可空 | 默认值 | 字段说明 |
|---|---|---|---|---|
| file_id | int(11) | no | | 文件的 ID，主码 |
| problem_id | int(11) | no | | 隶属题目的 ID，主码 |
| version | int(11) | no | 1 | 该文件对应题目的版本号 |

9．**输入输出文件关联表（intooutfiles）**保存每对输入输出文件之间的关联，具体设计如下表：

表 3-9 输入文件输出文件关联表

| 字段名 | 数据类型 | 是否可空 | 默认值 | 字段说明 |
|---|---|---|---|---|
| in_id | int(11) | no | | 输入文件的 ID，主码 |
| out_id | int(11) | no | | 输出文件的 ID，主码 |

10．**邮件表（mails）**用于保存用户发送的邮件的基本信息，具体设计如下表：

表 3-10 邮件表

| 字段名 | 数据类型 | 是否可空 | 默认值 | 字段说明 |
|---|---|---|---|---|
| mail_id | int(11) | no | | 邮件 ID，主码 |
| topic_id | int(11) | no | | 邮件主题 ID |
| title | varchar(200) | no | | 邮件标题 |
| content | text | yes | | 邮件内容 |
| unread | char(1) | no | Y | 邮件是否已读 |
| time | datetime | no | | 邮件发送时间 |
| to_user | varchar(20) | no | | 发送邮件的用户的 ID |
| from_user | varchar(20) | no | | 收取邮件的用户的 ID |
| reader_del | char(1) | no | N | 邮件是否被收件人删除 |
| writer_del | char(1) | no | N | 邮件是否被发件人删除 |

当邮件为回复一封邮件的邮件时，topic_id 为它回复的邮件的 topic_id，否则为其 mail_id。

11．**新闻表（news）**保存发布的新闻的信息，具体设计如下表：

表 3-11 新闻表

| 字段名 | 数据类型 | 是否可空 | 默认值 | 字段说明 |
|---|---|---|---|---|
| news_id | int(11) | no | | 新闻 ID，主码，自动生成 |
| publishtime | datetime | no | | 新闻发布的时间 |
| title | varchar(200) | no | | 新闻的标题 |
| content | text | yes | | 新闻的内容 |

12．**题目表（problems）**保存题目的信息，具体设计如下表：

表 3-12 题目表

| 字段名 | 数据类型 | 是否可空 | 默认值 | 字段说明 |
|---|---|---|---|---|
| problem_id | int(11) | no | | 题目的 ID，主码，自动生成 |
| title | varchar(200) | no | | 题目的标题 |
| description | text | no | | 题目的描述 |
| input | text | no | | 输入描述 |
| out_put | text | no | | 输出描述 |
| sample_input | text | no | | 输入样例 |
| sample_output | text | no | | 输出样例 |
| hint | text | yes | | 题目提示 |
| source | varchar(100) | yes | | 题目来源 |
| addin_time | int(11) | no | | 加题时间 |
| time_limit | int(11) | no | | 题目总时间限制 |
| case_time_limit | int(11) | no | | 单组测试用例时间限制 |
| memory_limit | int(11) | no | | 题目内存限制 |
| available | char(1) | no | Y | 题目是否有效 |
| accepted | int(11) | no | 0 | 题目被通过次数 |
| submit | int(11) | no | 0 | 题目被提交次数 |
| solved_users | int(11) | no | 0 | 通过该题的用户数目 |
| submit_users | int(11) | no | 0 | 提交该题的用户数目 |
| standard_time_limit | int(11) | yes | | 参考程序的运行所用时间 |
| standard_momory_limit | int(11) | yes | | 参考程序的运行所需内存 |
| version | int(11) | no | 0 | 题目的版本号 |
| spj | char(1) | no | N | 题目是否是 Special Judge |

13．**比赛题目列表（problemtocontests）**保存比赛的题目列表，具体设计如下表：

表 3-13 比赛题目列表

| 字段名 | 数据类型 | 是否可空 | 默认值 | 字段说明 |
|---|---|---|---|---|
| problem_id | int(11) | no | | 题目 ID，主码 |
| contest_id | int(11) | no | | 比赛 ID，主码 |
| in_contest_id | int(11) | no | | 比赛中题目的编号 |

14．**实时状态表（statuses）**保存每次提交代码产生的实时状态的信息，具体设计如下表：

表 3-14 实时状态表

| 字段名 | 数据类型 | 是否可空 | 默认值 | 字段说明 |
|---|---|---|---|---|
| status_id | int(11) | no | | 实时状态 ID，主码 |

| | | | | |
|---|---|---|---|---|
| user_id | varchar(20) | no | | 用户的 ID |
| problem_id | int(11) | no | | 题目的 ID |
| contest_id | int(11) | no | 0 | 比赛的 ID |
| time | int(11) | no | 0 | 运行所用时间 |
| memory | int(11) | no | 0 | 运行所用内存 |
| submit_time | datetime | no | | 提交的时间 |
| result | tinyint(4) | no | 15 | 实时状态的结果 |
| language | tinyint(4) | no | | 提价的代码的编程语言 |
| code_id | int(11) | no | | 提交代码的 ID |
| code_length | int(11) | no | | 提交代码的长度 |
| submit_ip | varchar(20) | no | | 提交的 IP |
| error_id | int(11) | no | 0 | 错误信息的 ID |
| type | char(1) | no | N | 实时状态的类型 |

type 可以分为 N，R，S，D 共 4 种，其中 N 为普通提交，R 为管理员提交，S 为测试参考程序提交，D 表示条目被删除。

15．**用户表（users）**保存用户的基本信息以及相关的用户设置等，具体设计如下表：

表 3-15 用户表

| 字段名 | 数据类型 | 是否可空 | 默认值 | 字段说明 |
|---|---|---|---|---|
| user_id | varchar(20) | no | | 用户名（ID），主码 |
| email | varchar(100) | yes | | 用户的邮箱 |
| show_email | char(1) | no | Y | 是否显示用户的邮箱 |
| submits | int(11) | no | 0 | 用户的提交数 |
| sovleds | int(11) | no | 0 | 用户通过的题目数 |
| available | char(1) | no | N | 用户是否有效 |
| last_login_ip | varchar(20) | yes | | 用户上次登录的 ip |
| last_login_time | datetime | yes | | 用户上传登录的时间 |
| volume | int(11) | yes | | 用户上次的题目卷号 |
| language | tinyint(4) | yes | | 用户默认提交语言 |
| password | text | no | | 用户的密码 |
| reg_time | datetime | no | | 注册的时间 |
| nickname | varchar(100) | yes | | 用户的昵称 |
| school | varchar(100) | yes | | 用户的学校 |
| permission | char(6) | no | ------ | 权限字段 |
| share_code | char(1) | no | N | 默认是否共享代码 |

权限字段若有一个字符为’A’，则表示该用户有管理员（Admin）权限，如果有一个字符为’V’，则表示该用户有查看代码（ViewSource）权限。

## 3.3 总体框架设计

可分布式部署的在线评测系统是一个复杂的，综合分布式系统，按照其部件功能的划分可以分为三层结构：网络交互层，数据控制层和内核层。数据控制层主要为中心服务模块，数据接口模块作为中心服务模块的一个子模块，控制着中心服务模块的所有数据的存储与查询。因此有必要对系统的功能划分以及中心服务模块的功能划分做出说明，来支持解释数据

接口模块的设计与实现。

### 3.3.1 系统框架以及可分布式部署解决方案

整个可分布式部署的在线评测系统包括五个大的模块，即 Web 服务模块、中心服务模块、评测模块、数据库服务模块、用户终端模块。每个模块可以分别在多台不同的服务器上，其中 Web 服务模块，用户终端模块以及评测模块都可以部署多个，它们都通过网络 socket 连接唯一的中心服务模块，并由中心服务模块统一控制数据库中的数据以保证数据的唯一性，以此来实现系统的可分布式部署。

整个系统的软件层次图如图 3-2：

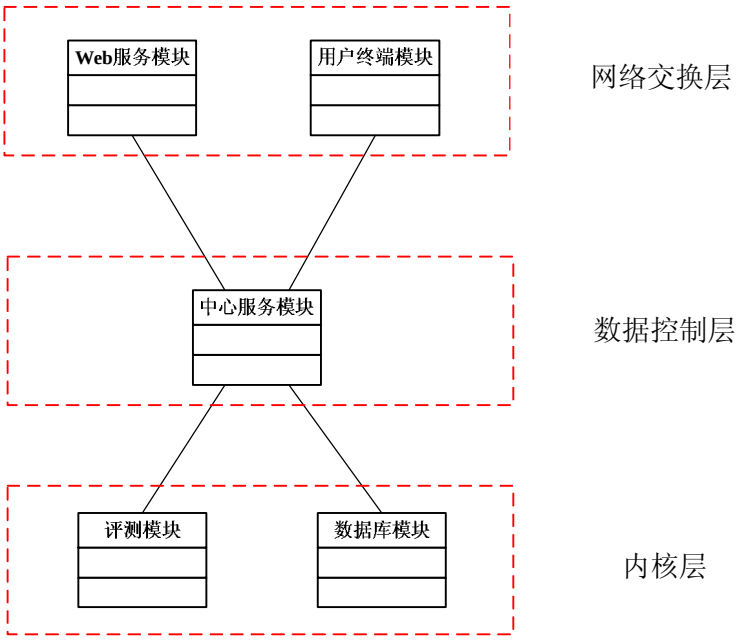


图 3-2 系统框架图

Web 服务模块主要完成与用户的交互。用户通过 Web 服务模块可以查看题库中的题目，且登录后的用户可以通过它请求查看某个题目是否通过，但查看完题目相处解决方案后，用户可以通过它向系统提交该题目的解答程序的源代码并请求系统评判这次提交的解答程序是否符合题目的要求。Web 服务模块还向用户提供查看解答程序的评定结果的接口。系统不定期的举办各种类型的比赛，所有比赛的信息，包括比赛的统计信息，都通过 Web 服务模块直观的反馈给用户。Web 服务模块还向用户提供邮件功能的接口，包括查询邮件，发送邮件等。Web 服务模块对用户提供讨论区接口，使用户能够通过 Web 服务模块与其他用户讨论相关问题。Web 服务模块可以有多个，并可以部署到多个服务器上，并通过网络 socket 套接字与中心服务模块相连。当一个 Web 服务模块的一些网页数据被改变时，通知中心服务模块网页数据被改变，此后其它 Web 服务模块使用这些网页数据向中心服务模块查询是否有数据改变时，发现了这些网页数据的改变后，向中心服务模块获取新的网页数据。通过这些以保证 Web 服务模块部署在多台服务器时仍然能够保证数据的一致性，从而实现 Web 服务模块的可分布式部署。

中心服务模块负责相应从 Web 服务模块和用户终端模块的用户请求。根据由 Web 服务模块和用户终端模块传来的请求，中心服务模块识别这些请求，根据这些请求向数据库模块或者在系统所属文件中查询和更新用户请求的数据，并将执行的结果反馈给 Web 服务模块和用户终端模块，从而将请求执行结果返回给用户。如果中心服务模块识别出该请求是提交

题目的解答程序的源代码或者重新评判提交的解答程序的源代码，中心服务模块将评判解答程序所需的所有信息全部传送给评测模块，并根据评判模块返回的结果执行相应的更新数据库的操作。在整个系统中，中心服务模块是唯一的，统一控制系统数据的处理，以保证系统的数据唯一性。

评测模块负责编译执行用户提交的解答程序的代码，并将执行的结果反馈给中心服务模块。具体功能概括为在练习和比赛时候，编译用户提交上来的解答程序的代码，然后运行程序，把所有测试数据运行完毕后，根据解答程序的输出结果与标准答案做比较，并由此得出用户解答程序的运行结果，同时将结果通过 socket 传给中心服务模块，若这个过程中出现了错误，如编译错误、运行错误等，则终止这一过程，并将错误信息传给中心服务模块。在整个系统中，评测模块也可以是多个的，每一个都可以分别部署在单独的服务器上，通过 socket 套接字与中心服务模块相连，实现评测模块的可分布式部署。

数据库模块用于存储可分布式部署的在线评测系统的大部分数据，包括用户信息，题目信息，比赛信息，新闻，提交的代码，提交的代码的评测结果，题目或者比赛附属的文件的绝对路径，用户的邮件，讨论区的讨论条目等数据。

用户终端模块设计为当 Web 服务过于拥挤时，临时给用户提交代码和查询自己提交的解答程序的源代码的结果。用户登录用户终端时，向中心服务模块发送登录请求，中心服务模块接受了服务请求后将题目与比赛的列表发送给用户终端，之后用户可以通过用户终端向系统提交解答程序源代码，查询结果等操作请求。

### 3.3.2中心服务模块框架

中心服务模块正如 3.3.1 中所述，接受并识别来至 Web 服务模块和用户终端模块的用户交互请求。

中心服务模块分为 3 个子模块，网络接口模块，评测控制模块和数据接口模块。中心服务模块的框架结构图如图 3-3：

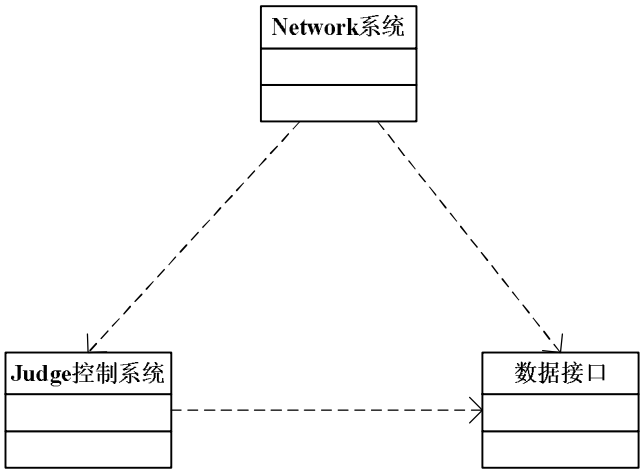


图 3-3 中心服务模块框架图

网络接口模块负责接受和识别来至 Web 服务模块和用户终端模块的用户交互请求，如果用户的请求中需要用到评测模块，则通过评测控制模块让评测模块进行代码评测处理。否则根据用户请求类型通过数据接口获取请求的信息或者用请求中的信息更新数据库以及文件系统中的数据文件。

评测控制模块负责控制评测模块进行评测任务并接受评测的结果。评测控制模块用 socket 与评测模块相连接，当网络接口接受了一个评测请求时，评测控制模块控制评测模块

开始评测任务，并将评测任务进行中所需的所有数据按序传送给评测模块，并等待来至评测模块的评测结果，最后通过数据接口将结果插入到数据库中。

数据接口模块向网络接口模块以及评测控制模块提供查询，更新以及插入数据的接口。当网络接口模块或者评测控制模块通过数据接口查询数据时，根据其所要查询的数据的类型与内容，将数据封装成好，以便网络接口模块或者网络接口模块直接获取封装完成的数据。

### 3.3.3数据接口模块框架

数据接口模块作为查询，更新和插入数据的接口，需要控制整个系统的所有运行中出现的数据，包括数据库中的数据以及文件系统中相关的文件。因此，数据接口必须能够保证系统数据的完整性与正确性，并在适当的情况下提供数据加速的服务。根据这些功能的描述，数据接口模块设计分为数据库接口，文件接口，以及 Cache（缓存加速）管理 3 大部分。数据接口模块框架设计图如图 3-4：

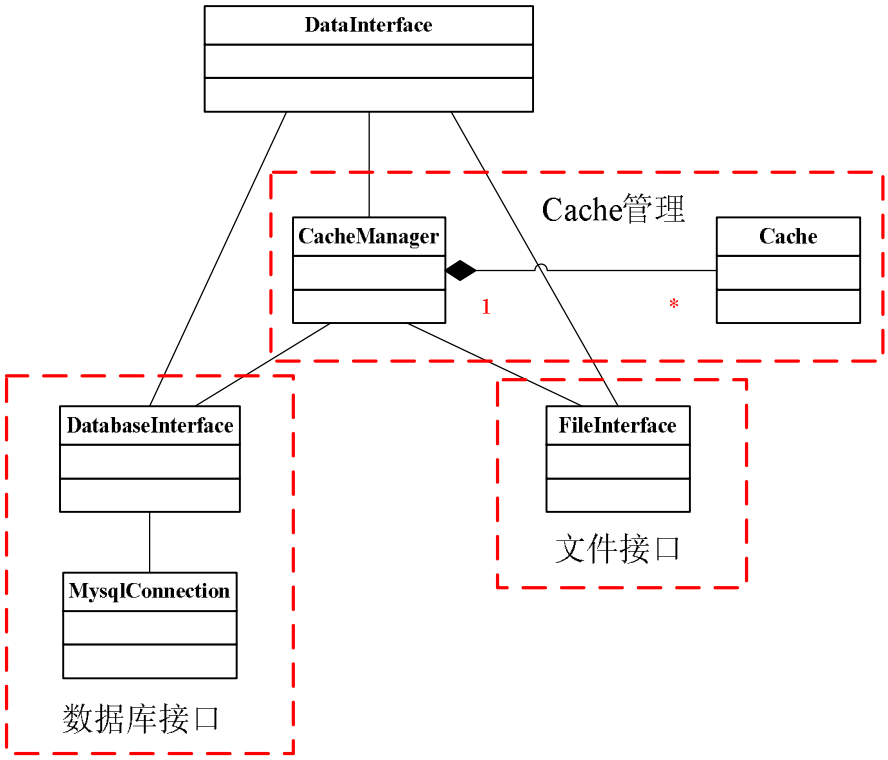


图 3-4 数据接口模块结构图

数据库接口负责与 MySQL 数据库通信，从 MySQL 数据库中查询数据，更新 MySQL 数据库中的数据和向 MySQL 数据库中插入数据。数据库接口提供操作数据库中所有数据的接口，包括添加用户，修改用户，添加题目，修改题目，查询题目，添加比赛，修改比赛，查询比赛，添加代码，查看代码等接口。

文件接口负责管理文件系统中的相关文件，添加或者获取与可分布式部署的在线评测系统有关的文件的内容，包括通知信息，图片，输入输出文件，Special Judge 文件，参考解答程序源文件等。

Cache 管理负责对可分布式部署的在线评测系统某些数据进行数据加速，使之不需要从数据库或者文件系统中读出，而直接在内存中获取。Cache 管理提供获取实时状态列表，比赛统计信息，比赛排名信息，文件内容等的查询接口，其中对于实时状态还提供更新接口。

DataInterface 是整个数据接口模块的入口，数据接口的外观对象，所有数据接口的调用

都是通过 DataInterface 调用数据接口模块的数据库接口，文件接口和 Cache 管理的接口。

## 3.4 数据接口模块组成

数据接口模块提供操作 MySQL 数据库（数据库模块）与文件系统中的与可分布式部署的在线评测系统的文件中的数据的接口，并将操作的数据封装成系统可用的数据类型，供系统使用。数据接口模块如 3.3.3 中所述大体上分为 3 个部分，将这些部分全部综合在一起，数据接口模块主要的类大概有数据库接口类（DatabaseInterface），文件接口类（FileInterface），缓存管理器（CacheManager），缓存（Cache）类。

### 3.4.1 数据库接口（DatabaseInterface）

数据库接口提供对 MySQL 数据库（数据库模块）操作的接口，如果操作有返回数据时，将数据封装成可用的数据后返回。

数据库接口提供的主要接口有：

1．添加比赛（addContest），将一次比赛的基本信息，包括标题，内容，起始时间等插入数据库，并返回添加的比赛的比赛号。

2．提交比赛的题目列表（addProblemListtoContest），向数据库中存在的一次比赛添加比赛的题目列表。

3．添加可以参加比赛的用户列表到比赛（addUserListtoContest），向数据库中存在的一次比赛添加能够参加比赛的用户列表。

4．添加新闻（addNews），

5．添加题目（addProblem），向数据库中添加一个题目的基本信息，包括标题，描述，输入，输出等信息，并返回添加的题目的题目号。

6．添加实时状态（addStatus），向数据库中添加一个实时状态的基本信息，包括题目号，运行时间，代码编号等信息，并返回添加的实时状态的编号。

7．添加邮件（addMail），向数据库中添加一封邮件的基本信息，包括主题，内容等信息。

8．添加代码（addCode），向数据库中添加一个代码的基本信息，包括内容和是否共享等信息，并返回刚添加的代码的编号。

10．添加错误信息（addError），向数据库中添加一条错误信息，并返回刚添加的错误信息的编号。

11．添加题目有关的文件路径（addFilePathtoProblem），将一个文件关联到它相关的题目号上。

12．添加比赛有关的文件路径，（addFilePathtoContest），将一个文件关联到它相关的比赛号上。

13．关联输入与输入文件（addInputtoOutput），将某个输入文件的文件与某个输出文件相关联。

14．获取与题目有关的文件的路径列表（getProblemFile），从数据库中得到与某个题目相关的文件的路径的列表。

15．获取与比赛有关的文件的列表（getContestFile），从数据库中得到与某个比赛相关的文件的路径的列表。

16．获取代码（getCode），从数据库中得到某个提交的解答程序的源代码的内容。

17．获取比赛信息（getContest），从数据库中得到某个比赛的基本信息，包括标题，起

始时间等。

18．获取比赛列表（getContestList），从数据库中得到比赛的列表。

19．获取比赛排名（getContestRankList），从数据库中获取与比赛有关的实时状态，并根据这些实时状态计算出比赛中用户的排名列表。

20．获取比赛统计信息（getContestStatistics），从数据库中获取与比赛有关的实时状态，并根据这些实时状态计算出比赛的统计信息。

21．获取错误信息（getError），从数据库中获取某一错误信息。

22．获取邮件信息（getMail），从数据库中获取某一邮件的内容。

23．获取题目（getProblem），从数据库中获取题目的基本信息，包括标题，描述，输入输出描述，输入输出样例等信息。

24．获取题目列表（getProblemList），从数据库中得到题目的列表

25．获取排名列表（getRankList），从数据库中获取用户的排名列表。

26．获取用户信息（getUserInfo），从数据库中获取用户的基本信息，包括用户名，密码，昵称，邮箱等信息。

27．获取用户列表（getUserList），从数据库中获取用户列表。

28．更新题目（updateProblem），更新数据库中的题目的信息，包括题目的标题，描述，输入输出说明等。

29．更新题目的参考程序执行信息（updateProblemStandardLimit），更新数据库中题目的参考程序的运行信息，包括 standard_time_limit，standard_memory_limit

30．更新比赛（updateContest），更新数据库中的比赛的信息，包括比赛的标题，起始时间等等。

31．更新比赛的题目列表（updateProblemListtoContest），更新数据库中的某场比赛的题目列表。

32．更新可以参加某一比赛的用户列表（updateUserListtoContest），更新数据库中允许参加某场比赛的用户的列表。

33．更新用户信息（updateUser），更改数据库中的某一用户的基本信息，包括昵称，邮箱等信息。

34．更新用户的权限设置（updateUserPermission），更改数据库中某一用户的权限设置信息。

35．更新用户的密码（updateUserPassword），更改数据库中的某一用户的密码。

36．获取题目的统计信息（getProblemStatistics），获取数据库中所有与某个题目有关的实时状态，根据这些实时状态计算题目的统计信息并返回。

37．添加用户（addUser），向数据库中添加用户的基本信息，包括用户名，密码，昵称，邮箱，学校等信息。

38．屏蔽或者取消屏蔽题目（disableProblem），设置题目的屏蔽位（Available）为屏蔽或者非屏蔽状态。

39．屏蔽或者取消屏蔽用户（disableUser），设置用户的屏蔽位（Available）为屏蔽或者非屏蔽状态。

40．屏蔽或者取消屏蔽比赛（disableContest），设置比赛的屏蔽位（Available）为屏蔽或者非屏蔽状态。

41．检查比赛中用户是否通过某个题目（checkContestAcBefore），

42．更新用户通过的题目数（updateUserSolved），增加或减少用户通过的题目数。

43．更新用户提交的次数（updateUserSubmit），增加或减少用户提交的次数。

44．测试是否有权限（checkPermission），测试用户是否有权限参加某个比赛。

45．获取比赛的题目列表（getContestProblemList），从数据库中获取比赛的题目的列表。

46．获取比赛的题目个数（getContestProblemNum），从数据库中获取比赛的题目的数目。

47．获取用户通过的题目号列表（getUserACProblem），

48．获取题目状态列表（getProblemStatus），从数据库中查询与题目有关的实时状态列表。

49．获取用户的排名（getUserRank），从数据库中查询某一用户的排名。

50．获取题目的输入输出文件路径列表（getProblemInAndOutFile），从数据库中获取某一题目的输入文件和输出文件的路径表。

51．获取题目的 Special Judge 文件路径（getProblemSpjFile），从数据库中获取某一题目的 Special Judge 文件的路径表。

52．获取题目的参考程序的路径地址（getProblemStandardSource），从数据库中查询某个题目的参考程序存放的路径的地址。

53．更新题目提交的次数（addProblemSubmit），增加或减少该题的提交次数。

54．更新题目通过的次数（addProblemSolved），增加或减少通过该题的提交次数。

55．更新题目通过的用户数目（addProblemUserSolved），增加或减少通过该题目的用户数。

56．更新题目的提交的用户数目（addProblemUserSubmit），增加或减少提交了该题题目的用户数。

### 3.4.2 文件接口（FileInterface）

文件接口提供对存储在文件系统中的文件数据操作的接口，如果操作有返回数据时，将数据封装成可用的数据后返回。

文件接口提供数接口有：

1．更新链接列表（addLink），更新首页上友情链接的列表。

2．添加文件（addFile），创建指定文件名的文件，并将文件的数据写入文件。

3．获取文件（getFile），根据文件名获取文件数据内容。

4．更新文件（updateFile），更新文件的数据内容。

5．获取链接列表（getLink)，获取友情链接的列表。

6．更新通知信息（updateNotice），更新通知信息文件的内容。

7．获取通知信息（getNotice)，获取通知信息。

8．获取文件大小（getFileSize），查看某一文件的文件大小。

### 3.4.3 缓存管理器（CacheManager）

缓存管理器顾名思义负责对加速缓存的管理，控制缓存（Cache）的整个生命线上的操作。经过分析可分布式部署的在线评测系统中需要进行缓存加速的信息包括比赛的统计信息，比赛的用户排名列表，实时状态列表等，因此缓存管理器内部管理着比赛统计信息缓存（Contest Statistics Cache），比赛用户排名列表缓存（Contest Ranklist Cache），实时状态信息缓存(Status Cache)。当从获取某个 Cache 内容时，缓存管理器首先判断他需求的缓存(Cache)内容是否已缓存，若缓存则直接将缓存内容返回，否则通过数据库接口或者文件接口查询并缓存该数据内容然后才将数据内容返回。

### 3.4.4 缓存（Cache）

缓存作为加速信息读取的核心，使用一个 map 作为数据存储和与关键字关联的容器，维护一个访问存储的数据内容的队列。缓存不仅保存数据内容与与该数据内容关联关键字，还保存该数据内容的访问时间，失效时间等信息。当缓存空间不足时，缓存将上次访问时间距离现在最长的数据内容移出缓存。为了防止不用的数据内容长时间占用缓存空间，每个缓存都有一个缓存清理器，清理器每隔一段时间将时间超过失效时间的缓存数据内容移出。

# 第4章　数据安全与防范措施

## 4.1 SQL 注入攻击与防范

SQL 注入通过把 SQL 命令插入到 Web 表单或者输入域名或者页面请求的查询字符串，最终达到欺骗服务器执行恶意的 SQL 命令。由于可分布式部署的在线评测系统已经将数据库模块与 Web 服务模块物理隔开，所有数据库操作都是通过中心服务模块的数据接口来实现的，数据库的错误信息并不会显示到页面上，所以并不需要对系统的入侵。然而 Web 服务模块并没有将 Web 表单或者输入域名或者页面请求的查询字符串进行封装，因此中心服务模块（主要是数据接口模块）必须对防范 SQL 的注入攻击对系统的破坏。根据 SQL 注入的原理，可以得出 SQL 注入对系统的破坏主要有两种方式，在查询字符串中加入导致数据库查询出错的字符（如‘’）和在查询字符串中加入 SQL 命令。

对于第一种情况来说，实践证明这种攻击如果不被防范时，直接导致的是系统的异常终止。为了防范这种原因导致系统的异常终止，数据接口采取了转义这些字符使包含这些字符的 SQL 语句能够正常工作的。

对于第二种情况来说，数据接口采用在所有 SQL 语句中的用单引号将有数据接口提供的数据内容引起来的做法，使得非法添加的 SQL 命令并不会被执行。

## 4.2 数据同步安全

为了并行的响应网络交互层的请求，中心服务模块采取的是多线程 Socket 服务器模式，使得系统能够同时并行提供服务，同时也带来了多线程的特有问题——数据的同步。如果在中心服务模块的网络接口模块保证数据的同步，将会导致系统大量的不必要的开销（因为存在数据同步问题的地方主要是加速缓存上）。由此可见合理的做法是在数据接口中解决此问题。为了防止线程的并发访问缓存导致缓存的破坏，在所有缓存操作中加入互斥锁使之只能够在访问缓存时是串行操作的。

此外，系统运行中的数据并不是一成不变的，缓存的内容必须要与数据库和文件中的数据一致，否则缓存中的数据就会失效。为了保证缓存中内容的与数据库和文件中的数据同步，数据接口在修改可能会导致缓存失效的数据的同时修改 Cache 中的内容。

## 4.3 数据库安全性约束

本系统存储在数据库中的数据有一定的存储规则，例如不能存在 statuses 表中有两条或者两条以上的条目的 code_id 相同。为了防止出现这种情况而破坏数据库的正确性，则必须对数据库的数据进行约束。本系统的数据库中采用的办法是利用触发器。以上面所述的情况为例：

```
create trigger double_code_check before insert on statuses
   for each row begin
      set @ct = (select count(*) from statuses where code_id = NEW.code_id);
```

```
    if @ct > 0 then    //  如果已存在有条目的 code_id 和新插入的条目的 code_id  相同
      set NEW.type = 'D';      //则设置为删除态
    end if;
  end
```

在插入或者更新 statuses 表中的数据时，判断是否合法，如果不合法则把刚才操作的条目设置为删除态。

# 第5章　　结果分析

## 5.1 相关技术的应用

### 5.1.1 利用缓存加速的过程以及 LRU 技术的应用

用户每次查询利用缓存加速的数据时（后面用实例查询比赛统计信息来说明），数据接口模块先使用缓存管理器查询需要查询的比赛统计信息是否已被缓存管理器缓存，如果已被缓存则直接返回该数据，否则通过数据库接口或者文件接口将数据读入缓存再返回该数据。当缓存容量已满时，缓存根据一定的规则从缓存移除一些数据内容，以装载新的数据内容。其整个过程如下图 5-1：

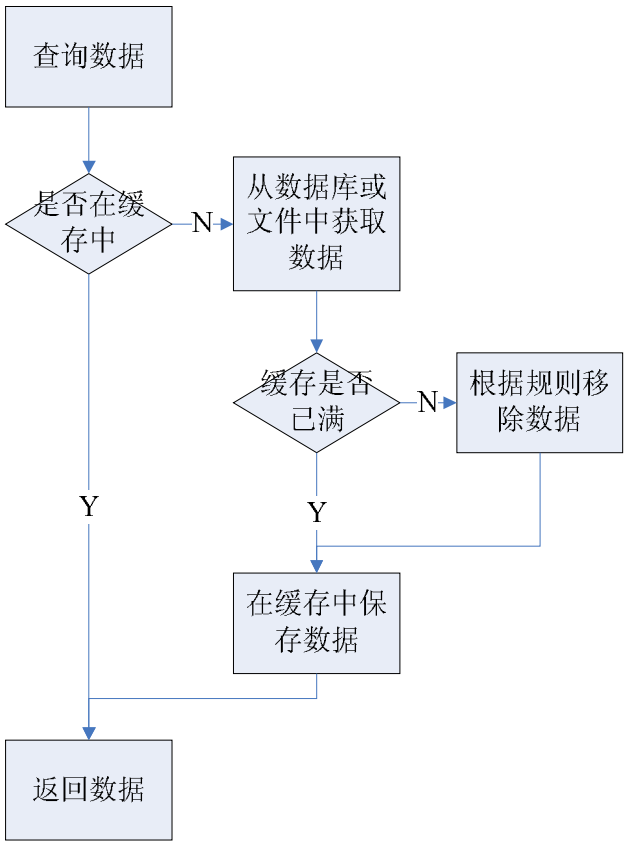


图 5-1 缓存查询流程图

在移除数据时，数据的置换算法对于用于加速的缓存来说显得尤为重要。常用的置换算法有 FIFO 和 LRU 算法。FIFO 算法简单但命中率并不高，LRU 算法稍微复杂一些但命中率高。为了获得更佳的性能借用操作系统中高速缓存的替换方案，这里的缓存使用最近最久没被访问的数据移出缓存的模式，即 LRU 算法。缓存利用其数据内容的访问队列来实现 LRU 置换算法的应用，每次访问操作都会从访问队列中取出访问数据的访问单元，并重新设置到队列尾部来保证队列首位始终是最近最久没被访问的数据的访问单元，当移除缓存中的数据时，只需移除队首的访问单元所对应的数据内容以实现删除最近最久没被访问的数据内容。

### 5.1.2 设计模式的应用

数据接口模块现阶段的内部现行结构设计并不意味这是最好的结构设计。或许在未来的某个时候对数据接口模块进行改进，从而修改数据接口模块的内部结构。为了应付这种未来可见的修改，就必须在设计数据接口模块时要更加倾向于模块化，用以减少数据接口模块的变化对中心服务模块的其他两个模块的影响，使得中心服务模块的其他两个模块关心的重点放置在接口上而不关心数据接口模块的内部实现。外观（Facade）模式为数据接口模块提供了一个单一而简单的界面对象 DataInterface。DataInterface 对中心服务模块的其他两个模块屏蔽了数据接口内部的各个类，提供了一个数据接口模块的统一接口。这样不仅使得调用数据接口模块中的接口更加方便和容易，而且将数据接口模块对外的通信和相互依赖关系降到到最小，实现了数据接口模块与中心服务程序的其他两个模块的松耦合关系。

缓存管理器对于整个数据接口模块，乃至整个中心服务程序来说，都不会产生多个个实例，为了保证缓存管理器只有一个实例并提供一个众所周知的访问点，最佳的选择是采用单件（Singleton）模式。使用单件模式不仅对缓存管理器的唯一实例提供了的受控访问，而且操作起来比类操作更加方便。

# 第6章　结论与展望

本课题涉及的在线评测系统目前已运行在武汉大学 ACM/ICPC 集训队内网服务器上。至论文完稿之日，本评测系统在需求提出之时的绝大部分功能都已经完成且稳定运行在实际环境中，未完成的功能均为不影响正常使用的小功能改进和用户体验改善。

时至今日数据接口模块与数据库已经基本达到了预期的设计要求，缓存加速功能能够正常工作，所有提供的基本接口都能正常操作数据库和文件。在系统的实际运行中与测试修改过程中，数据接口模块的这种设计使得它能够很好的进行扩展和修改而不影响到其他的模块，数据接口模块还能够很好的与其他模块的协调运行。

数据接口模块的设计与实践综合运用了面向对象设计的设计模式，编程语言，算法数据结构，数据库系统，Linux 编程，操作系统等知识。在数据接口设计与实现的过程中，查看了许多书籍与文档，与整个系统的其他开发人员进行了很好的协作。

当然数据接口模块并不是完美的设计，有些地方仍然需要改进。像数据接口模块的数据库接口类（DatabaseInterface）太过庞大，不利于该类的修改编译等。又如数据接口模块提供的接口的操作粒度的调制并没到最佳化。粒度过大会导致并行操作时数据接口模块使某些数据的修改失败，粒度过小则会导致数据接口的实现复杂化。

总之，数据接口模块在现阶段仍然是成功的设计，数据接口模块能够稳定的工作，中心服务程序也能稳定的运行。然而随着时间的发展，必然有许多新的应用需要系统的支持、新的数据需要数据接口模块的支持。因此开发的系统中必定存在不少的缺点和不足，其中也必然包括数据接口模块，希望未来系统能够在功能扩展与功能改进过程中，功能能够更加齐全，性能能够更加稳定。

# 参考文献


[1] （美）W.Richard Stevens，Stephen A.Rago 著，尤晋元、张亚英、戚正伟译，UNIX 环境高级编程（第二版）[M]， 机械工业出版社， 2006

[2] （美）Erich Gamma，Richard Helm，Ralph Johnson，John Vlissides 著，李英军、马晓星、蔡敏、刘建中等译，设计模式：可复用面向对象软件的基础[M]，机械工业出版社，2000

[3] （美）W.Richard Stevens，Bill Fenner，Andrew M.Rudoff 著，杨继张译，UNIX 网络编程（第一卷：套接口 API）（第三版）[M]，清华大学出版社，2005

[4] （美）Stanley B. Lippman，Josee Lajoie，Barbara E. Moo 著，李师贤、蒋爱军、梅晓勇、林瑛等译，C++ Primer 中文版（第四版）[M]，人民邮电出版社，2006

[5] （美）Samuel P.Harison III, Guy L. Steele Jr. 著，邱仲潘等译，C 语言参考手册[M]，机械工业出版社，2003

[6] 赵丰、赵端正，基于 B/S、C/S 集成模式应用软件的开发研究[J]，中国科技信息，2006(18)

[7] 李文新、郭炜，北京大学程序在线评测系统及其应用[J]，吉林大学学报(信息科学版)，2005(S2)

[8] 苑文会，基于黑箱测试的源代码在线评测[D]，北京化工大学，2005

[9] 曹玉峰，国家信息学奥林匹克竞赛在线评测系统[D]，吉林大学，2006

[10] 周高嵌，基于白箱测试的源代码在线评测系统[D]，北京化工大学，2005

# 致　　谢

在我们完成毕业设计的过程中，有非常多给予我们帮助的师长和同学。

感谢我们毕业设计的指导老师董文永老师，在完成毕业设计的过程中给予了我们大量专业知识和求学处事的指导。董文永老师身为武汉大学 ACM/ICPC 集训队的总教练，在我们课题小组人员参加 ACM/ICPC 竞赛的过程中给予了大量的帮助和支持，让我们在完成毕业设计的过程中发扬武汉大学 ACM/ICPC 集训队快速高效能吃苦的精神，使得我们在短期内非常好的完成了这一复杂的毕业设计的设计实现和测试。

感谢武汉大学在线评测系统（noah 及 oak）的开发人员：杨传辉、吴永坚、刘高杰、杨宝奎、董超、高双星，有他们的工作才奠定了本课题最原始的理论和实践基础，他们也本毕业设计实现的过程中给出了大量目的明确、效果明显的建议和帮助。

感谢武汉大学 ACM/ICPC 集训队，给我们提供了开发的环境和平台，让我们能完成毕业设计并能将其部署以提供服务，感谢集训队的同学们给我们大量有重要价值的用户反馈和改进建议，让我们将毕业设计做的更好。

感谢在我大学四年中遇到的各位老师的辛勤的培养和教诲，才能有我们今天的知识储备和专业修养，能在面对未来的学习和工作时能有的放矢、得心应手。

感谢我的爸爸和妈妈，他们在我求学的道路上给予了无私的关心，支持和鼓励。在这里，我深深的感谢他们，祝他们身体健康，万事如意。

刘潜

二〇〇八年五月

于武汉大学 ACM/ICPC 集训队训练基地