**数据库（顶层）设计说明DBDD**

**编写人员：\_\_\_宗亮亮\_\_\_**

**2017 4.8**

目录

[1. 引言 3](#_Toc479608679)

[1.1 标识 3](#_Toc479608680)

[1.2 数据库概述 3](#_Toc479608681)

[1.3 文档概述 3](#_Toc479608682)

[2. 引用文件 3](#_Toc479608683)

[3. 数据库设计决策 4](#_Toc479608684)

[4. 数据库详细设计 4](#_Toc479608685)

[4.1 运行环境 4](#_Toc479608686)

[4.2 基本设计概念和处理流程 5](#_Toc479608687)

[4.2.1 管理员登录功能 5](#_Toc479608688)

[4.2.2 管理员发布活动功能 6](#_Toc479608689)

[4.2.3 管理员更改个人信息功能 7](#_Toc479608690)

[4.2.4 手机端用户查询发布活动功能 7](#_Toc479608691)

[4.2.5 手机端用户报名活动功能 8](#_Toc479608692)

[4.3 概念结构设计 8](#_Toc479608693)

[4.4 逻辑结构设计 10](#_Toc479608694)

[4.5 功能需求与程序模块的关系 14](#_Toc479608695)

[4.6 人工成立过程 14](#_Toc479608696)

[4.7 尚未解决的问题 14](#_Toc479608697)

[5. 用于数据库访问或操纵的软件配置项的详细设计 15](#_Toc479608698)

[6. 需求的可追踪性 15](#_Toc479608699)

[7. 注解 16](#_Toc479608700)

[8. 附录 17](#_Toc479608701)

# 引言

## 标识

|  |  |
| --- | --- |
| **文档标识** | AM-DOC-2017-4-8 |
| **英文标题** | Campus Activity Management System |
| **软件名称** | 校园活动管理系统 |
| **软件标识号** | ActMan |
| **版本号** | 1.0 |
| **状态** | 修改版 |

## 数据库概述

校园活动管理系统的数据库需要录入学校学院信息、社团信息以及管理员信息，并且维护管理员发布的活动信息。本系统由西北工业大学软件与微电子学院软件工程专业的五名学生开发，交由西北工业大学团委、社团团长使用，由开发组负责维护，本系统于2016年3月由大学生创新竞赛资金支持开发，团队成员共五人，均为在校本科生，预计该系统于2017年5月1日开发完成，可交由西北工业大学团委、社团团长使用，系统运行环境：网页端windows xp professional版本以上，安卓端android 4.0以上。详情可参见《校园活动管理系统可行性研究报告》《校园活动管理系统需求分析报告》以及《校园活动管理系统数据库（顶层）设计说明》。

## 文档概述

《数据库设计说明》用于说明数据库设计，方便用户了解系统的数据组织结构，使用仅限用户，谨防外泄。

# 引用文件

第07讲 软件工程文档写作

GB/T 8567 2006计算机软件文档编制规范

# 数据库设计决策

1. 数据库接受活动查询操作，输入数据为此活动信息的主键：活动id，输出信息为活动的名称、发布人、具体内容、发布时间等信息。数据库接受用户查询操作，输入数据为此用户信息的主键：用户id，输出信息为用户的姓名、性别和年龄等信息；
2. 在网页服务器端以及安卓客户端查询数据库的动作都是通过按钮点击，后台将会自动执行数据库的查询操作，响应时间在1s以内。数据库的各个表之间存在多对多关系，此时采用第三张表将两张表连接起来的技术，所以在查询时会经常用到多表查询。对于不允许的操作，在网页端和安卓端都会给出相应的错误信息呈现给用户；
3. 数据库中的数据在网页端通过table表格呈现，在安卓手机端通过自定义的listView呈现给用户；
4. 本系统使用的数据库管理系统为MySql 5.7.17，为了灵活适应需求变化，将数据库中表格尽量细分，使得用户需求变更时需要修改的功能点尽量减少；
5. 本系统的数据库具有充分的可用性，支持对大量数据的增删改查。而且将保密用户信息放在首位，数据库为独立的一个模块，只能通过外部指定的接口访问，防止数据的丢失或泄漏；
6. 数据库的备份策略是每隔一小时检测数据是否变更，如果变更则做一次备份，备份保留时间为一周，一周过后删除一周之前的备份数据。

# 数据库详细设计

## 运行环境

软件环境：Inter core i3 380 1GHz 以上

512MB RAM

Windows 7 home basic 32位

Microsoft SQL Server 2003

磁盘设备空间>10G

Android 4.0以上

硬件环境：PC机一台：物理内存1G以上，物理外存2G以上；

可上网的智能手机一部：物理内存1G以上，物理外存2G以上。

## 基本设计概念和处理流程

## 管理员登录功能



## 管理员发布活动功能



## 管理员更改个人信息功能



## 手机端用户查询发布活动功能



## 手机端用户报名活动功能



## 概念结构设计

根据用户需求梳理数据之间关联如下：

每个管理员和用户属于某一个学院；

每个管理员可以发布多个活动；

每个用户可以收藏多个活动，每个活动可以有多个用户收藏；

每个用户可以报名多个活动，每个活动可以有多个用户报名；

每个活动可以属于多个分类，每个分类下可以有多个活动；

每个活动可以由多个社团共同发布，每个社团可以发布多个活动；

超级管理员具有最高权限，可以对所有活动、用户、管理员操作。

根据以上分析抽象出信息结构，可得该系统的E-R图：



## 逻辑结构设计

根据上述概念结构设计出逻辑结构，将E-R图转换为关系模型：

数据库activity共包含以下11个表：学院表academy、用户表member、管理员表administrator、超级管理员表sAdministrator、社团表organization、活动表activity、类别表category、用户收藏活动表mem\_act\_collect、用户报名活动表mem\_act\_signUp、活动属于分类表act\_cat\_belongTo、活动属于社团表act\_org\_belongTo。

1. 学院表academy

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 字段名称 | 数据结构 | 说明 |
| 1 | id | INT | 学院id，主键，自增长 |
| 2 | name | VARCHAR(50) | 学院名，非空 |

1. 用户表member

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 字段名称 | 数据结构 | 说明 |
| 1 | id | INT(10) | 用户id，主键 |
| 2 | name | VARCHAR(20) | 用户名，非空 |
| 3 | password | VARCHAR(20) | 用户密码，非空 |
| 4 | sex | VARCHAR(2) | 用户性别，非空 |
| 5 | telephone | VARCHAR(11) | 用户手机号，可选填 |
| 6 | qq | VARCHAR(10) | 用户qq号，可选填 |
| 7 | email | VARCHAR(20) | 用户邮箱，可选填 |
| 8 | academyId | INT | 用户所在学院id，可选填 |
| 9 | grade | VARCHAR(10) | 用户所在年级，可选填 |
| 10 | image | VARCHAR(100) | 用户头像，可选填 |

1. 管理员表administrator

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 字段名称 | 数据结构 | 说明 |
| 1 | id | INT(10) | 管理员id，主键 |
| 2 | name | VARCHAR(20) | 管理员名，非空 |
| 3 | password | VARCHAR(20) | 管理员密码，非空 |
| 4 | sex | VARCHAR(2) | 管理员性别，非空 |
| 5 | telephone | VARCHAR(11) | 管理员手机号，可选填 |
| 6 | qq | VARCHAR(10) | 管理员qq号，可选填 |
| 7 | email | VARCHAR(20) | 管理员邮箱，可选填 |
| 8 | academyId | INT | 管理员所在学院id，外键，可选填 |
| 9 | grade | VARCHAR(10) | 管理员所在年级，可选填 |

1. 超级管理员表sAdministrator

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 字段名称 | 数据结构 | 说明 |
| 1 | id | INT(10) | 超级管理员id，主键 |
| 2 | name | VARCHAR(20) | 超级管理员名，非空 |
| 3 | password | VARCHAR(20) | 超级管理员密码，非空 |

1. 社团表organization

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 字段名称 | 数据结构 | 说明 |
| 1 | id | INT | 社团id，主键，自增长 |
| 2 | name | VARCHAR(20) | 社团名，非空 |
| 3 | introduction | TEXT | 社团介绍，非空 |

1. 活动表activity

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 字段名称 | 数据结构 | 说明 |
| 1 | id | INT | 活动id，主键 |
| 2 | name | VARCHAR(100) | 活动名，非空 |
| 3 | location | VARCHAR(50) | 活动地点，非空 |
| 4 | administratorId | INT | 活动发布管理员id，外键，非空 |
| 5 | startTime | DATETIME | 活动开始时间，非空 |
| 6 | endTime | DATETIME | 活动结束时间，非空 |
| 7 | introduction | TEXT | 活动介绍，可选填 |
| 8 | linkmanInfo | TEXT | 活动联系人，可选填 |
| 9 | lastModifyTime | DATETIME | 活动最后修改时间 |
| 10 | createTime | DATETIME | 活动创建时间 |
| 11 | image | VARCHAR(400) | 活动图片，最多10张 |
| 12 | mainOrg | VARCHAR(100) | 活动主办方 |
| 13 | subOrg | VARCHAR(100) | 活动承办方 |
| 14 | minNum | INT | 活动最少组队报名人数，默认0 |

1. 类别表category

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 字段名称 | 数据结构 | 说明 |
| 1 | id | INT | 类别id，主键，自增长 |
| 2 | name | VARCHAR(20) | 类别名称，非空 |

1. 用户收藏活动表mem\_act\_collect

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 字段名称 | 数据结构 | 说明 |
| 1 | memberId | INT | 收藏活动的用户id，外键 |
| 2 | activityId | INT | 收藏活动的活动id，外键 |
| 3 | collectTime | DATETIME | 收藏活动的时间 |

1. 用户报名活动表mem\_act\_signUp

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 字段名称 | 数据结构 | 说明 |
| 1 | memberId | INT | 收藏活动的用户id，外键 |
| 2 | activityId | INT | 收藏活动的活动id，外键 |
| 3 | collectTime | DATETIME | 收藏活动的时间 |
| 4 | othermessage | VARCHAR(1000) | 其他信息（如组队报名的其他组员信息） |

1. 活动属于分类表act\_cat\_belongTo

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 字段名称 | 数据结构 | 说明 |
| 1 | activityId | INT | 活动id，外键 |
| 2 | categoryId | INT | 类别id，外键 |

1. 活动属于社团表act\_org\_belongTo

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 字段名称 | 数据结构 | 说明 |
| 1 | activityId | INT | 活动id，外键 |
| 2 | organizationId | INT | 社团id，外键 |

## 功能需求与程序模块的关系

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 所属模块  功能需求 | 管理员发布活动模块 | 管理员修改个人信息模块 | 用户报名/收藏活动模块 | 用户修改个人信息模块 |
| 管理员发布活动 | **√** |  |  |  |
| 显示活动列表 | **√** |  | **√** |  |
| 管理员更新活动 | **√** |  |  |  |
| 显示活动信息 | **√** |  | **√** |  |
| 显示已有社团名 | **√** | **√** |  |  |
| 显示已有学院名 | **√** | **√** |  |  |
| 显示活动报名信息 | **√** |  |  |  |
| 导出报名信息到xls文件 | **√** |  |  |  |
| 显示管理员个人信息 |  | **√** |  |  |
| 活动开始提醒 |  |  | **√** |  |
| 显示用户个人信息 |  |  |  | **√** |

## 人工成立过程

活动管理员需在活动发布后进行活动的信息维护，活动变更后在客户端可以接收到活动变更提醒；管理员通过查看用户报名信息进行一系列的活动准备。

## 尚未解决的问题

管理员进行活动的变更操作一旦成功不能回滚。

# 用于数据库访问或操纵的软件配置项的详细设计

1. 本系统采用的是MySql数据库，进行数据库连接遵循MFC设计模式，通过指定要连接的数据库名称、密码采用JDBC获取连接，并将获取连接操作封装为一个类。
2. 管理员通过与数据库建立起来的连接进行活动的增删改查：

① 增：先根据管理员传入信息进行活动创建，之后再将创建的活动加入活动-类别、活动-社团的多对多关系；

② 删：先在活动-类别、活动-社团多对多关系中删除对应活动，然后再从活动表中删除活动；

③ 改：对于活动表，直接使用Update更新，而对于活动-类别、活动-社团的多对多关系，先将原数据删除再添加修改后的数据；

④ 查：采用多表查询返回所有最终数据（如社团名等）。

1. 用户通过与数据库建立起来的连接进行活动查询、收藏、报名：

① 查询：同2中管理员④；

② 收藏：在对应用户-收藏表中增加表项；

② 报名：在对应用户-报名表中增加表项。

1. 网页端后台通过Servlet进行数据库操作，采用MyEclipse编程，故MyEclipse作为一个外部软件配置项与本地Tomcat关联。
2. 后台使用Tomcat搭建服务器，Tomcat的相关使用参考Tomcat官网提供的文档。

# 需求的可追踪性

本数据库设计根据用户需求梳理数据之间的逻辑关系，详见4.3节。故客户需求的变更可以向前跟踪到文档形式的需求，如《软件需求规格说明书》以及本文档中涉及到的需求。这样就能区分开开发过程中或开发结束后由于客户需求变更而受到影响的需求，编程人员只需对此部分需求变更进行系统的修改即可。同时，可以从文档中描述的功能回溯到相应的用户需求，完成功能与需求相对应。

# 注解

理解本文档需要掌握的术语和定义：

Attribute（属性）（关系模型）：属性是关系中命名的列。

Binary relationship（二元关系）：一个ER术语，用于描述两个实体间的关系。例如， panch Has Staff。

Candidate key（候选键，ER关系模型）：仅包含唯一标识实体所必须得最小数量的 属性/列的超键。

Column（列）：参加属性（attribute）。

Constraint（约束）：数据库不允许包含错误数据的一致性规则。

CSCI（软件配置项）：软件生存周期各个阶段活动的产物经审批后即可称之为软件配置项。

Data redundancy（数据冗余）：在多个表中存储的重复数据。

Data security（数据安全）：包括对数据库对象（如表和视图）的访问和使用以及用 户可以在这些对象上实施的操作。

Database design（数据库设计）：数据库应用生命周期中的一个阶段，包括创建一个 支持公司的操作和目标的数据库的设计。

Database Management System，DBMS（数据库管理系统）：一个能够让用户定义、 创建和维护数据库并控制对数据库的访问的软件系统。

Foreign key（外健）：一个表中的一个列或者多个列的集合，这些列匹配某些其他（也 可能是同一个）表中的候选键。

Integrity constaints（完整性约束）：防止出现数据库中的数据不一致的约束。

Primary key（主健，关系模型）：在一个表中用来标识记录唯一性的候选键。

Relational model（关系模型）：以表（或关系）的形式表示数据的数据模型。

Server（服务器）：为发出请求的客户提供服务的软件应用程序。参见两层/三层客 户端－服务器体系结构。

Transaction（事务）：由用户和应用程序执行的一个动作或一系列动作，这些动作 访问或修改数据库的内容。

Update anomalies（更新异常）：当用户视图更新一个包含冗余数据的标识可能引 起的不一致。有三种类型的异常：插入、删除和更新。

# 附录

数据库创建脚本：

CREATE TABLE academy (

id INT PRIMARY KEY AUTO\_INCREMENT ,

name VARCHAR(50) NOT NULL

)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

CREATE TABLE member (

id INT(10) PRIMARY KEY ,

name VARCHAR(20) NOT NULL ,

password VARCHAR(20) NOT NULL ,

sex VARCHAR(2) NOT NULL ,

telephone VARCHAR(11) ,

qq VARCHAR(10) ,

email VARCHAR(20) ,

academyId INT ,

grade VARCHAR(10) ,

image VARCHAR(100) ,

CONSTRAINT fk\_acaid1 FOREIGN KEY(academyId) REFERENCES academy(id)

)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

CREATE TABLE administrator (

id INT(10) PRIMARY KEY ,

name VARCHAR(20) NOT NULL ,

password VARCHAR(20) NOT NULL ,

sex VARCHAR(2) NOT NULL ,

telephone VARCHAR(11) ,

qq VARCHAR(10) ,

email VARCHAR(20) ,

academyId INT ,

grade VARCHAR(10) ,

CONSTRAINT fk\_acaid2 FOREIGN KEY(academyId) REFERENCES academy(id)

)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

CREATE TABLE sAdministrator (

id INT(10) PRIMARY KEY ,

name VARCHAR(20) NOT NULL ,

password VARCHAR(20) NOT NULL

)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

CREATE TABLE organization (

id INT PRIMARY KEY AUTO\_INCREMENT ,

name VARCHAR(50) ,

introduction TEXT

)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

CREATE TABLE activity (

id INT PRIMARY KEY AUTO\_INCREMENT ,

name VARCHAR(100) ,

location VARCHAR(50) ,

administratorId INT ,

startTime DATETIME ,

endTime DATETIME ,

introduction TEXT ,

linkmanInfo TEXT ,

lastModifyTime DATETIME ,

lastModifySAdmin INT ,

lastModifyAdmin INT ,

flag INT ,

createTime DATETIME ,

image VARCHAR(400) ,

mainOrg VARCHAR(100) ,

subOrg VARCHAR(100) ,

minNum INT DEFAULT 0,

CONSTRAINT fk\_admid1 FOREIGN KEY(administratorId) REFERENCES administrator(id) ,

CONSTRAINT fk\_admid2 FOREIGN KEY(lastModifyAdmin) REFERENCES administrator(id) ,

CONSTRAINT fk\_sadmid1 FOREIGN KEY(lastModifySAdmin) REFERENCES sAdministrator(id)

)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

CREATE TABLE category (

id INT PRIMARY KEY AUTO\_INCREMENT ,

name VARCHAR(20)

)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

CREATE TABLE mem\_org\_collect (

memberId INT ,

organizationId INT ,

CONSTRAINT fk\_mid2 FOREIGN KEY(memberId) REFERENCES member(id) ,

CONSTRAINT fk\_oid1 FOREIGN KEY(organizationId) REFERENCES organization(id)

)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

CREATE TABLE mem\_act\_collect (

memberId INT ,

activityId INT ,

collectTime DATETIME ,

CONSTRAINT fk\_mid3 FOREIGN KEY(memberId) REFERENCES member(id) ,

CONSTRAINT fk\_actid1 FOREIGN KEY(activityId) REFERENCES activity(id)

)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

CREATE TABLE mem\_act\_signUp (

memberId INT ,

activityId INT ,

signUpTime DATETIME ,

othermessage VARCHAR(1000),

CONSTRAINT fk\_mid4 FOREIGN KEY(memberId) REFERENCES member(id) ,

CONSTRAINT fk\_actid2 FOREIGN KEY(activityId) REFERENCES activity(id)

)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

CREATE TABLE org\_act\_deliver (

organizationId INT ,

activityId INT ,

deliverTime DATETIME ,

CONSTRAINT fk\_oid2 FOREIGN KEY(organizationId) REFERENCES organization(id) ,

CONSTRAINT fk\_actid3 FOREIGN KEY(activityId) REFERENCES activity(id)

)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

CREATE TABLE act\_cat\_belongTo (

activityId INT ,

categoryId INT ,

CONSTRAINT fk\_actid4 FOREIGN KEY(activityId) REFERENCES activity(id) ,

CONSTRAINT fk\_cid1 FOREIGN KEY(categoryId) REFERENCES category(id)

)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

CREATE TABLE act\_aca\_belongTo (

activityId INT ,

academyId INT ,

CONSTRAINT fk\_actid5 FOREIGN KEY(activityId) REFERENCES activity(id) ,

CONSTRAINT fk\_acaid3 FOREIGN KEY(academyId) REFERENCES academy(id)

)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;

CREATE TABLE act\_org\_belongTo (

activityId INT ,

organizationId INT ,

CONSTRAINT fk\_actid6 FOREIGN KEY(activityId) REFERENCES activity(id) ,

CONSTRAINT fk\_orgid1 FOREIGN KEY(organizationId) REFERENCES organization(id)

)ENGINE=InnoDB DEFAULT CHARSET=utf8 ;