《数据库系统原理》课程设计

系统设计报告

题目名称：GRE(美国研究生入学考试)学习系统

学号及姓名：15231031 张楠

15231036 杨旺旺

2017年 11月 16 日

组内同学承担任务说明

|  |  |  |
| --- | --- | --- |
|  | 张楠 | 杨旺旺 |
| 系统设计阶段 | 1. E-R图的建立 2. E-R图到关系模型的转换 3. E-R图的分析 4. 关系模式范式等级的判定与规范化 | 1. 需求分析 2. 数据流图的建立 3. 数据元素表的建立 4. E-R图到关系模型的转换 |
| 系统实现阶段 | 1. 商讨并在数据库中建立基本表并维护数据库 2. 实现后端程序中的数据库交互操作，编写大部分请求处理操作 3. 定义并实现了触发器 4. 设计了前端用户界面的样式，为前端界面添加功能组件 5. 依照各个功能进行程序的测试 | 1. 商讨并在数据库中建立基本表并维护数据库 2. 配置并建立后端程序的主模块部分，负责处理前端请求分类；并编写一部分请求处理模块 3. 实现了前端程序的逻辑控制，在处理了后端回复的基础上完成了用户界面的交互、显示等功能 4. 依照各个功能进行程序的测试 |
| 系统报告撰写 | 1. 完成设计报告与实现报告的需求分析部分 2. 完成设计报告中的E-R图设计部分、范式判定与规范化部分和优化部分 3. 完成实现报告中的后端程序结构设计部分、触发器实现与定义部分、存储过程与定义部分和数据库基本表部分 | 1. 完成设计报告与实现报告的需求分析部分 2. 完成设计报告中的数据流图部分、数据元素表部分和数据关系模式部分 3. 完成实现报告中的前端程序结构设计部分、系统实现结果和总结部分 |

## 一．需求分析

**1．需求描述**

经过分析与设计，我们的GRE学习系统的功能主要包括用户登陆与注册、用户做历次考试的试卷、自动评判与打分、试题评论浏览与添加、用户错题库管理、用户生词库管理等五个主要部分。详细需求如下：

①用户登陆与注册：

管理用户注册账户的操作，并验证用户登录的信息。

②用户完成历次GRE考试的试卷：

用户可以从系统提供的历次GRE考试的试卷中下载所需试卷，并完成试卷中的试题。

③自动评判与打分机制：

当用户完成试卷提交后，可以自动判断用户提交答案的正误，并给出用户完成试卷的总分数。

④试题评论浏览与添加管理：

在用户完成试卷后，可以浏览试卷中每道题目下的评论区中其他用户对该题的评论，自己也可以添加或回复评论。

⑤用户错题库管理：

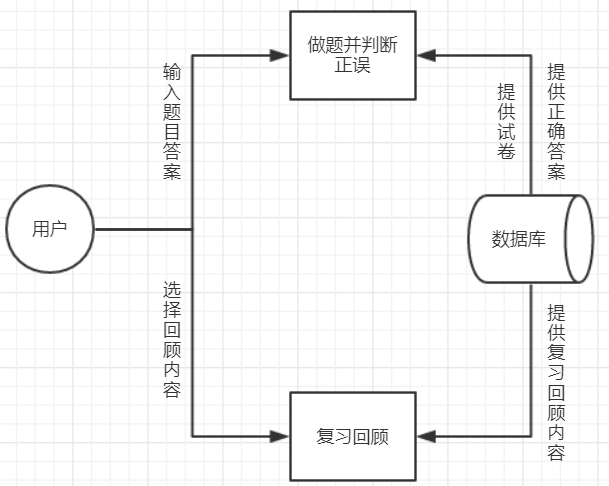
生成用户专属的错题库，管理用户做错的题目，也支持用户将需要记录的题目添加到错题库中，方便用户日后复习回顾。

⑥用户生词库管理：

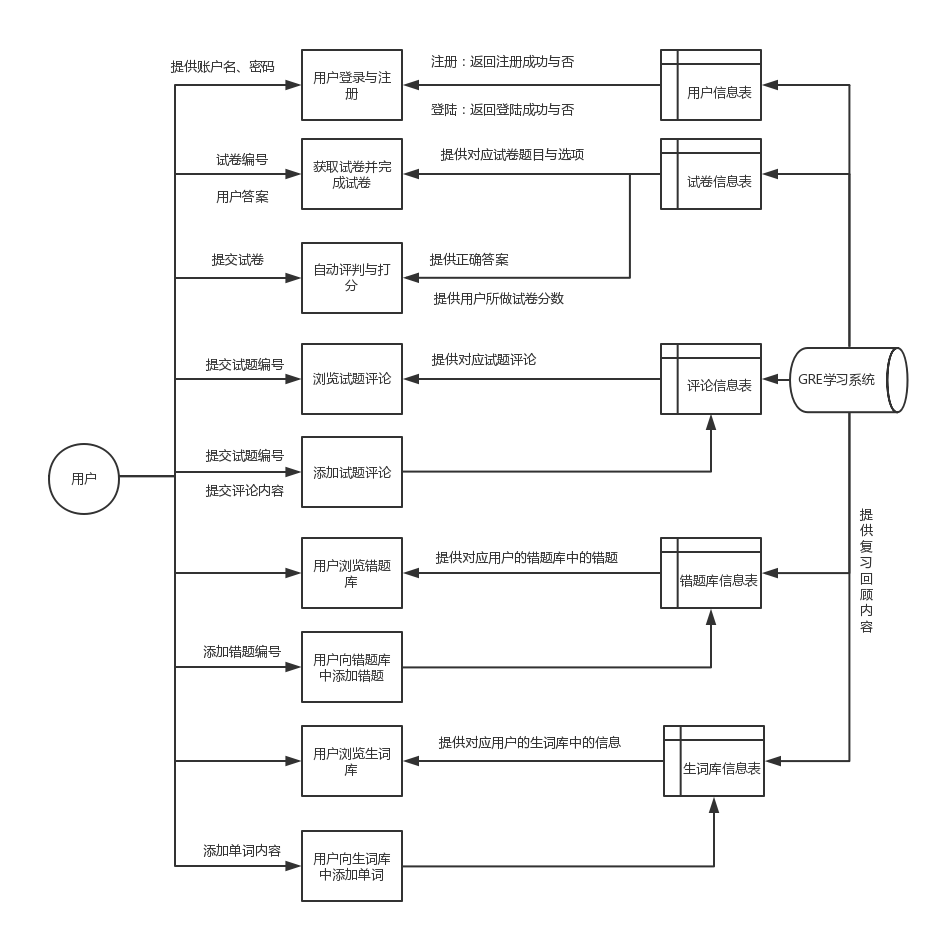
管理用户需要记忆的生词，提供生词的拼写以及中文解释等信息，方便用户复习回顾。

1. **数据流图**

从学习系统的使用用户的角度描述了该系统的顶层视图如下：



对系统进行细化，描述该系统的具体功能的数据流图如下：



1. **数据元素表**

**用户信息**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **数据名** | **数据类型** | **长度** | **能否为NUL** | **是否为KEY** | **备注** |
| USID | int |  | 否 | 是 | 用户ID |
| USACCOUNT | char | 50 | 否 | 否 | 用户账户名 |
| USPASSWORD | char | 20 | 否 | 否 | 用户登陆密码 |

**试卷信息**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **数据名** | **数据类型** | **长度** | **能否为NULL** | **是否为KEY** | **备注** |
| PAID | int |  | 否 | 是 | 试卷ID |
| PATIME | char | 11 | 否 | 否 | 试卷时间 |
| PAQUES1 | int |  | 否 | 否 | 题目1的ID |
| PAQUES2 | int |  | 否 | 否 | 题目2的ID |
| PAQUES3 | int |  | 否 | 否 | 题目3的ID |
| PAQUES4 | int |  | 否 | 否 | 题目4的ID |
| PAQUES5 | int |  | 否 | 否 | 题目5的ID |
| PAQUES6 | int |  | 否 | 否 | 题目6的ID |
| PAQUES7 | int |  | 否 | 否 | 题目7的ID |
| PAQUES8 | int |  | 否 | 否 | 题目8的ID |
| PAQUES9 | int |  | 否 | 否 | 题目9的ID |
| PAQUES10 | int |  | 否 | 否 | 题目10的ID |

**题目类型信息**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **数据名** | **数据类型** | **长度** | **能否为NULL** | **是否为KEY** | **备注** |
| QUESID | int |  | 否 | 是 | 题目ID |
| TTYPE | char | 3 | 否 | 否 | 题目类型 |

**单词信息**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **数据名** | **数据类型** | **长度** | **能否为NULL** | **是否为KEY** | **备注** |
| WOID | int |  | 否 | 是 | 单词ID |
| WOENG | char | 20 | 否 | 是 | 单词英文 |
| WOCHI | char | 20 | 否 | 否 | 单词中文 |

**用户错题库信息**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **数据名** | **数据类型** | **长度** | **能否为NULL** | **是否为KEY** | **备注** |
| USID | int |  | 否 | 是 | 用户ID |
| QUESID | int |  | 否 | 是 | 错题ID |

**评论信息**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **数据名** | **数据类型** | **长度** | **能否为NULL** | **是否为KEY** | **备注** |
| COID | int |  | 否 | 是 | 评论ID |
| COTIME | varchar | 30 | 否 | 否 | 评论时间 |
| COCON | varchar | 300 | 否 | 否 | 评论内容 |
| CONICE | int |  | 否 | 否 | 点赞数量 |
| COQUESID | int |  | 否 | 否 | 题目编号 |
| COFROM | int |  | 否 | 否 | 发送用户ID |

**用户生词库信息**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **数据名** | **数据类型** | **长度** | **能否为NULL** | **是否为KEY** | **备注** |
| USERID | char | 9 | 否 | 是 | 用户ID |
| UWENG | char | 20 | 否 | 是 | 单词英文 |
| UWCHI | char | 20 | 否 | 否 | 单词中文 |
| UWTIME | char | 11 | 否 | 否 | 记录时间 |

**点赞记录信息**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **数据名** | **数据类型** | **长度** | **能否为NULL** | **是否为KEY** | **备注** |
| NIUSERID | int |  | 否 | 是 | 用户ID |
| NICOMID | int |  | 否 | 是 | 评论ID |
| NITIME | varchar | 50 | 否 | 否 | 点赞时间（精确到分钟） |

**做题痕迹信息**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **数据名** | **数据类型** | **长度** | **能否为NULL** | **是否为KEY** | **备注** |
| USID | int |  | 否 | 是 | 用户ID |
| PAID | int |  | 否 | 是 | 试卷ID |
| ANS1 | char | 4 | 能 | 否 | 用户试题1的答案 |
| ANS2 | char | 4 | 能 | 否 | 用户试题2的答案 |
| ANS3 | char | 4 | 能 | 否 | 用户试题3的答案 |
| ANS4 | char | 4 | 能 | 否 | 用户试题4的答案 |
| ANS5 | char | 4 | 能 | 否 | 用户试题5的答案 |
| ANS6 | char | 4 | 能 | 否 | 用户试题6的答案 |
| ANS7 | char | 4 | 能 | 否 | 用户试题7的答案 |
| ANS8 | char | 4 | 能 | 否 | 用户试题8的答案 |
| ANS9 | char | 4 | 能 | 否 | 用户试题9的答案 |
| ANS10 | char | 4 | 能 | 否 | 用户试题10的答案 |

## 数据库概念模式设计

## 下面给出本数据库系统设计的完整E-R图：**E-R**

以下给出一些比较重要的E-R图分析，虽然本部分并不在文档要求内，但出于明确设计细节与易于交互理解的目的，我们还是决定给出由E-R图到关系模式中的一些关键拆分与合并步骤。

首先，我们需要解释对题目实体的设计。可以看到，我们将题目拆分为了三种题目，而这看似与数据库设计的原则相违背，因为不管有多少种题目，他们终究是题目，可以用属性将他们区分。但是事实上，这三种题目的题干长度有所不同，选项数量有所不同，答案数量有所不同，因此在设计实体属性时，每种题目的属性数量有所不同，而不仅仅是相同属性的值不同。对于这种情况，一种解决方式是强行同化三中题的属性数量，比如只设置一个选项属性，而将所有的选项都存入这一个属性中。但是这样的做法很明显加大了非数据库操作的工作量，因为程序在数据库查询之后还需要自行解析数据。并且，对于“查询第5题C选项”这样的操作，这样的存储方式并不能达到目的。因此，我们选择的解决方式为将其分为三个实体分别存储，当需要对题目进行操作时，先查询题目的类型，再通过索引到对应的题目表中进行操作，这样不仅完全消除了冗余，而且减少了表的规模，减少了查询时间。因此我们认为这样的题目实体设计是合理的。

其次，我们需要减少表的数量。对于最简单的E-R图转化步骤，只需将实体属性和关系转化为表即可，但鉴于我们的E-R图比较复杂，如果不加以合并则会产生数量非常多的表。为了减少表的数量，我们在以下关系的转化中采取了合并表的操作。第一，一个试卷包含多道题目，此关系应使用一张表记录，但鉴于GRE考试中一张卷子的题目数量是一定的（为10道），因此只需向试卷实体表中加入10个属性即可，分别索引题号，则该关系完全可以被正确表达。第二，从E-R图可以看出，评论实体与用户实体有两个关系，与赞扬实体有一个关系，与三种题目各有一个关系，如果简单地进行关系与表的转化，则会出现多张表。对此，我们发现，可以将这所有的表存成一张。具体的操作方法是在评论中添加多条属性，比如表表评论的用户ID，被评论用户的ID，点赞数量，评论针对的题号等，这样一来，一张评论表就完全能够记录以上所有的关系了，同时，在下面的分析中我们会指出，这样的设计是满足3NF的。

以上是我们我们在设计中两个比较特殊的地方，其它的表都可以对应到E-R图上的关系或是实体。

**三、数据库逻辑模式设计**

**1．数据库关系模式**

**US**(USID,USACCOUNT,USPASSWORD)

**PAPER**(PAID,PATIME,PAQUES1,PAQUES2,PAQUES3,PAQUES4,PAQUES5,PAQUES6,PAQUES7,PAQUES8,PAQUES9,PAQUES10)

**WORD**(WOID,WOENG,WOCHI)

**USER\_WORD**(USERID,UWENG,UWCHI,UWTIME)

**USER\_FAULT**(USID,QUESID,QUESANS)

**USANS**(USID,PAID,ANS1,ANS2,ANS3,ANS4,ANS5,ANS6,ANS7,ANS8,ANS9,ANS10)

**QUESTYPE**(QUESID,TTYPE)

**PAPER\_WORD**(PWPAPERID,PWWORDID)

**PAPER**(PAID,PATIME,PAQUES1,PAQUES2,PAQUES3,PAQUES4,PAQUES5,PAQUES6,PAQUES7,PAQUES8,PAQUES9,PAQUES10)

**NICE**(NIUSERID,NICOMID,NITIME)

**COMENT**(COID,COTIME,COCON,CONICE,COQUESID,COFROM)

**QUES51**(QUID,QUCON,QUOPTA,QUOPTB,QUOPTC,QUOPTD,QUOPTE,QUANS1)

**QUES62**(QUID,QUCON,QUOPTA,QUOPTB,QUOPTC,QUOPTD,QUOPTE,QUOPTF,QUANS1,Q UANS2)

**QUES93**(QUID,QUCON,QUOPTA,QUOPTB,QUOPTC,QUOPTD,QUOPTE,QUOPTF,QUOPTG,Q UOPTH,QUOPTI,QUANS1,QUANS2,QUANS3)

数据库关系模式详细完整性约束

#=============================

#用户表

#属性 用户ID：int USID

# 账号：char[50] USACCOUNT

# 密码：char[20] USPASSWORD

create table US

(

USID int not null unique auto\_increment,

USACCOUNT char(50) not null unique,

USPASSWORD char(20) not null,

primary key(USID)

);

#=============================

#考卷表

#属性 试卷ID：int PAID

# 时间：char[11] PATIME

# 题目编号：int\*10 PAQUES(n)

# 题目类型：char[3]\*10 PATYPE(n)

create table PAPER

(

PAID int not null unique,

PATIME char(11) not null,

PAQUES1 int not null,

PAQUES2 int not null,

PAQUES3 int not null,

PAQUES4 int not null,

PAQUES5 int not null,

PAQUES6 int not null,

PAQUES7 int not null,

PAQUES8 int not null,

PAQUES9 int not null,

PAQUES10 int not null,

primary key(PAID)

);

#================================

#题目类型表

#属性 题目ID：int

create table QUESTYPE

(

QUESID int not null unique,

ttype char(3) not null,

primary key(QUESID)

);

insert into QUESTYPE (QUESID, ttype) VALUES(1, '51');

insert into QUESTYPE (QUESID, ttype) VALUES(2, '51');

insert into QUESTYPE (QUESID, ttype) VALUES(3, '62');

insert into QUESTYPE (QUESID, ttype) VALUES(4, '62');

insert into QUESTYPE (QUESID, ttype) VALUES(5, '62');

insert into QUESTYPE (QUESID, ttype) VALUES(6, '93');

insert into QUESTYPE (QUESID, ttype) VALUES(7, '62');

insert into QUESTYPE (QUESID, ttype) VALUES(8, '62');

insert into QUESTYPE (QUESID, ttype) VALUES(9, '62');

insert into QUESTYPE (QUESID, ttype) VALUES(10, '62');

#==============================

#单词表

#属性 单词ID：int WOID

# 英文：char[20] WOENG

# 中文：char[20] WOCHI

create table WORD

(

WOID int not null unique auto\_increment,

WOENG char(20) not null unique,

WOCHI char(20) not null,

primary key(WOID, WOENG)

);

#==============================

#用户错题表

#属性 用户ID：int USID

# 题目ID：int QUESID

# 题目答案 : char(4) QUESANS

create table USER\_FAULT

(

USID int not null,

QUESID int not null,

QUESANS char(4) not null,

primary key(USID, QUESID)

);

#==============================

#五选一题目表

#属性 题目编号：int QUID

# 题目内容：varchar QUCON

# 五个选项：char[50] QUOPT\*5

# 一个答案: char[2] QUANS

create table QUES51

(

QUID int not null unique,

QUCON varchar(500) not null unique,

QUOPTA char(50) not null,

QUOPTB char(50) not null,

QUOPTC char(50) not null,

QUOPTD char(50) not null,

QUOPTE char(50) not null,

QUANS1 char(2) not null,

primary key(QUID)

);

#==============================

#六选二题目表

#属性 题目编号：int QUID

# 题目内容：char[500] QUCON

# 六个选项：char[50] QUOPT\*6

# 两个答案: char[2] QUANS\*2

create table QUES62

(

QUID int not null unique,

QUCON varchar(500) not null unique,

QUOPTA char(50) not null,

QUOPTB char(50) not null,

QUOPTC char(50) not null,

QUOPTD char(50) not null,

QUOPTE char(50) not null,

QUOPTF char(50) not null,

QUANS1 char(2) not null,

QUANS2 char(2) not null,

primary key(QUID)

);

#==============================

#九选三题目表

#属性 题目编号：int QUID

# 题目内容：char[500] QUCON

# 九个选项：char[50] QUOPT\*6

# 三个答案: char[2] QUANS\*2

create table QUES93

(

QUID int not null unique,

QUCON varchar(500) not null unique,

QUOPTA char(50) not null,

QUOPTB char(50) not null,

QUOPTC char(50) not null,

QUOPTD char(50) not null,

QUOPTE char(50) not null,

QUOPTF char(50) not null,

QUOPTG char(50) not null,

QUOPTH char(50) not null,

QUOPTI char(50) not null,

QUANS1 char(2) not null,

QUANS2 char(2) not null,

QUANS3 char(2) not null,

primary key(QUID)

);

#================================

#评论表

#属性 评论编号 int COID

# 评论时间 char[11] COTIME

# 评论内容 char[50] COCON

# 点赞数量 int CONICE

# 题目编号 int COQUESID

# 发送用户编号 int COFROM

create table COMENT

(

COID int not null unique auto\_increment,

COTIME varchar(30) not null,

COCON varchar(300) not null,

CONICE int not null,

COQUESID int not null,

COFROM int not null,

primary key(COID)

);

#===============================

#考卷-生词表

#属性 考卷编号：int PWPAPERID

# 单词编号：int PWWORDID

create table PAPER\_WORD

(

PWPAPERID int not null,

PWWORDID int not null,

primary key(PWPAPERID, PWWORDID)

);

#===============================

#用户生词表

#属性 用户ID：int USERID

# 单词英文：char[20] UWENG

# 单词中文：char[20] UWCHI

# 记录时间：char[11] UWTIME

create table USER\_WORD

(

USERID int not null,

UWENG varchar(30) not null,

UWCHI varchar(100) not null,

UWTIME varchar(50) not null,

primary key(USERID, UWENG)

);

#================================

#点赞记录表

#属性 用户ID：int NIUSERID

# 评论ID：int NICOMID

# 点赞时间(精确到分钟)：char[17] NITIME

create table NICE

(

NIUSERID int not null,

NICOMID int not null,

NITIME varchar(50) not null,

primary key(NIUSERID, NICOMID)

);

#==================================

#做题痕迹表

#属性 用户ID：int

# 试卷ID：int

# 答案：char[4]

create table USANS

(

USID int not null,

PAID int not null,

ANS1 char(4),

ANS2 char(4),

ANS3 char(4),

ANS4 char(4),

ANS5 char(4),

ANS6 char(4),

ANS7 char(4),

ANS8 char(4),

ANS9 char(4),

ANS10 char(4),

primary key(USID, PAID)

);

**2．关系模式范式等级的判定与规范化**

下面对我们的设计进行范式分析。由于系统中包含多张表并且每个表中均包含多个属性，因此这里不会一一地对所有表的范式分析进行说明。我们在这里的展示方式为，找到一些复杂的表，对其进行3NF范式分析，从而展示我们的证明方法。对于其他比较容易分析的表，我们同样使用这种分析方法进行分析。在这里，我们选择的表为“评论表（COMENT）”和“试卷表（PAPER）”，因为根据上面的分析，它们包含了最多的关系（或不等价的属性），分析起来最为困难。下面我们给出分析步骤。

COMENT表中只含有一个码，为COID，即评论的编号。值得一提的是有一组属性为（COTIME，COQUESID，COFROM），依次为评论时间，评论针对的题号，评论发出者ID，这一属性组并不能确定一条评论因为时间属性可能会重复（如多个用户在同一时间对同一题目进行评论），而我们无法排除两个用户在一分钟内互动多次的可能。因此，除了COID，其他属性均为非主属性。因为只有一组码且码中只有一条属性，因此非主属性均是完全函数依赖于码的。接下来，再分析所有的传递函数关系。可以看到，表中的非主属性虽然很多，但没有一个非主属性或一组非主属性可以决定另一个的。综上COMENT关系模式符合3NF的要求。

PAPER表中含有两组码，分别为卷子标号PAID和当次考试时间PATIME。其它的十个属性均为非主属性，而且性质是等价的，表示的是十道题的编号。首先，两个码中都只含有一个属性，因此非主属性均是完全依赖于码的。其次，十个非主属性互相独立，彼此没有函数关系，因此分析传递性关系时不需考虑它们之间的关系。而又由于它们的等价性，我们取其中一个进行分析（如第一题编号PAQUES1）。可以看到，PAID->PAQUES1，PATIME->PAQUES1，而PAID与PATIME均为码，互为依赖关系，即PAID->PATIME且PATIME->PAID，因此不存在传递函数关系。综上所述，PAPER表这一关系模式同样满足3NF的要求。

依照上述分析方法，我们确定，所有的关系模式均满足3NF定义。

1. **数据库设计优化**

对于数据库优化的方面，我们从两个方面入手：使用范式规范化关系模式、增加触发器以维护数据的关联性。

对于范式规范化来说，上面的部分已经给出了我们规范化后的最终结果——即所有的关系模式均满足3NF。从宏观来看，整个数据库中已经没有冗余信息了。以试卷、单词、试题这三个实体为例，一种最原始的想法是建立一张表格记录一张卷子上的所有信息，包括其预置单词，题目（包括题目内容、选项、答案等）。然而，这样的处理方式显然会造成数据冗余，例如一个单词（或一道试题）被多张试卷引用。因此，我们将这样的关系模式改写，分别记录创建记录单词、试题与卷子的表，并使用编号对它们进行索引，如此一来，同样的试题只需存储一次（单词同理），只是他们的编号会被存储多次。而当我们要删除试卷时，单词与试题其实也并不需要改变，因为整个题库在这种状态下并不需要记录非常多的数据，单词与试题也均可以继续服务于其他试卷。使用这种优化方法，我们将不同的实体从整体中拆分出来，并使用额外的表去记录它们之间的关系，使得整个数据库在我们的需求范围内不会出现插入、删除、修改等异常。

在以上过程中，我们将关系模式进行了拆分，防止了冗余。但这也就要求系统在操作时需要兼顾各个数据间的关联关系。为了方便地维护这些关系，我们向数据库中加入了触发器，以此来完成一系列具有原子性的操作。以下内容在系统实现报告中也有体现，这里作为一种设计方法进行介绍。

经过分析我们发现，本系统中主要需要使用触发器维护“评论”与“点赞”之间的关系，其表现为：删除一条评论的同时需要删除关于这条评论的全部点赞信息，给一个评论点赞后需要将这个评论的点赞数自增。这样的操作在后端程序中可以实现，但使用触发器就显得格外简单，其定义与实现如下：

#================================

# 当评论被删除时，同时删除一切与之相关的点赞信息

create trigger nice\_delete

after delete on COMENT

for each row

delete from NICE where NICOMID=old.COID;

#================================

# 用户点赞时更新评论的点赞数

create trigger nice\_add1

after insert on NICE

for each row

update coment set CONICE = CONICE + 1 where COID=new.NICOMID;

此外，我们还使用了触发器维护卷子、预置单词与用户做题痕迹之间的关系：

#================================

# 删除试卷时同时删除试卷单词关系与所有用户在该卷子上的做题记录

create trigger paper\_word

after delete on paper

for each row begin

delete from paper\_word where paper\_word.PWPAPERID = old.PAID;

delete from USANS where USANS.PAID = old.PAID;

end

通过这样的优化手段，数据库操作的完整性得到了进一步加强。同时，添加这些触发器也很大地帮助了后端程序——触发器的操作可以简化后端程序的代码编写，同时使代码更具有调理。

1. **最终版修改说明**

相较于最初的设计，我们在最终版中做出了以下修改：

1. 我们将相关表中的用户ID均修改为了int型（此前为char型，其设计目的是便于插入字母且用户数量不会受到限制），因为int型可以支持自增操作并且更方便于后台的管理与操作
2. 删除了COMENT表中的“被评论用户ID”域，因为前端程序中并没有使用这一信息，产生了冗余,删除了COMENT表中的”评论的用户ID“COTO域，因为评论都是针对题目的，只需要记录评论者不需要记录被评论者，产生了冗余，所以删除了COTO。
3. 将与试题内容、评论内容有关的所有信息均修改为了varchar型变量(此前为char型变量)，因为char型变量不足以存储超长的文字内容
4. 将用户ID、评论ID等设置为了auto\_increment型，便于后端将该项属性自增
5. 删除了NOTE表，我们取消了记录笔记的功能，改成了收藏题目的功能，所以不需要NOTE表了。
6. 向用户的错题库记录表中增加了“答案”域，用于记录用户自己的答案，这样的设计并不与“用户做题痕迹表”相冲突，反而是必要的，因为“做题痕迹表”会随着用户的重复做题而改变，但是用户在记录错题时可能更希望记录自己第一次做错时的错误答案。
7. 关于数据可的设计优化，主要是增加了相应的触发器，保证了相关数据项的完整性，也减少了后端程序的代码量，其主要说明请见“系统实现报告”