数据库课程设计

————某学校的题库管理系统设计说明书

机械*1712 U201710958* 黎洲波

2018 年 6 月 28 日

目录

1 问题的描述

随着当今中小学校各科的题量增多，教师们时刻面对着浩繁的卷帙和茫茫的题海，许多时候都会出现同样的题反复考察，学生不耐其烦，教师的出题意图也得不到较好的体现。此时如果能对每一科的各类题型的信息进行规范管理，以及对题库中题的添加、删除、更新、查询操作，便能极大减轻管理者的工作负担。当老师组卷时也希望根据题目所属章节和所要考察的知识范围对题目进行选择，同时希望能根据题型的难度系数筛选出需要考察的难易程度。

2 需求分析

2.1 需求分析

目前市面上的各类题书，有模拟题卷、中高考真题卷等多种类型，但是许多教师面临的问题是如何更加高效地选择某个科目各个章节的习题，或者是单独某一种题型进行针对性突破练习，此时如果利用数据库的特点，将大量的题目进行分类管理，在需要时对某类题进行选择，根据特别的需要（题组的考察范围、难度系数等），选出满足的题组，既能节省时间，还对具体的班级和学生有专一性。此外，教育者也能通过此系统来查看某章节某类题型的统计数据，能够据此完善出题的涵盖面，并更新题目、题型的信息，使题库成为更加完善的规范化管理系统。

2.1.1 需求信息

●实现课程、题型等基本信息的管理。

●能管理每一门课程的题型，每门课程的章节。

●每个题能通过题号（由系统自动按顺序生成,要求从1开始编号）迅速查找并且能够依据科目、章节、题型和难度系数进行分类。

●固定的题型有固定的分值，教师组成试卷时可根据总分挑选一定数目的题。

●题的创建时间或最后更改时间为默认的系统时间。

●定义存储过程实现查询各门课程、各种题型的习题数量以及查询指定课程各种题型和各章节的习题数量。

●可以自动抽题组成套题，习题每抽取一次，要使习题的抽取次数加1。

2.1.2 数据项和数据结构

经上述系统功能分析和需求总结，考虑到将来功能的扩展，设计如下的数据项和数据结构：

表 1: 数据项和数据结构表

信息	数据项	数据结构	备注
权限	权限人姓名	字符串型CHAR()	主键KEY
	授权时间	时间型DATETIME	默认为触发当前系统时间
学科	学科编号	整型INT	主键KEY
	学科名称	字符串型CHAR()	学科名称不能重复UNIPUE
章节	章节序号	字符串型CHAR()	序号不能重复UNIPUE
	所属学科	整型INT	从学科编号处索引
题型	编号	整型INT	主键KEY
	学科	整型INT	从学科编号处索引
	题型	字符串CHAR()	约束限制在('选择题','填空题','解答题'等)
	分值	整型INT	约束限制在(0,100)
题目	题目编号	整型INT	主键KEY，由系统自动生成
	题目内容	字符串CHAR()	可后续拓展
	所属章节	字符串CHAR()	从章节编号处索引
	题目类型	INT	从题型编号处索引
	难度系数	整型INT	约束限制在0–10
	操作人	字符串型CHAR()	从权限人处索引
	创建或更新时间	时间型DATETIME	默认为触发当前系统时间
	抽取次数	整型INT	为组卷抽到题的累积次数,默认为0
组卷	组卷编号	整型INT	主键KEY，由系统自动生成
	组卷科目	整型INT	主键KEY，从学科编号处索引
	组卷难度	整型INT	约束限制在0–10，由选题难度在分值上的加权平均聚合
	组卷总分	整型INT	在此为简洁，默认每类题型的个数，可后续扩展
	组卷时间	时间型DATETIME	默认为系统时间

2.2 系统功能结构

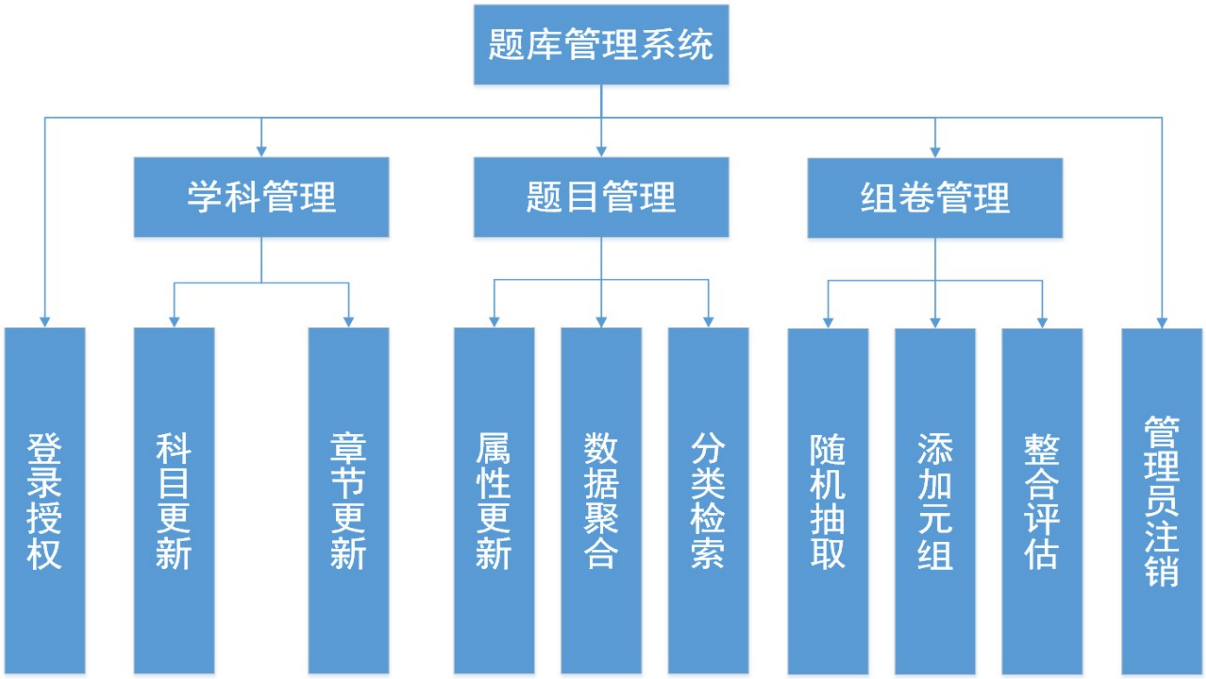

图 1: 系统功能结构图

2.3 数据流图

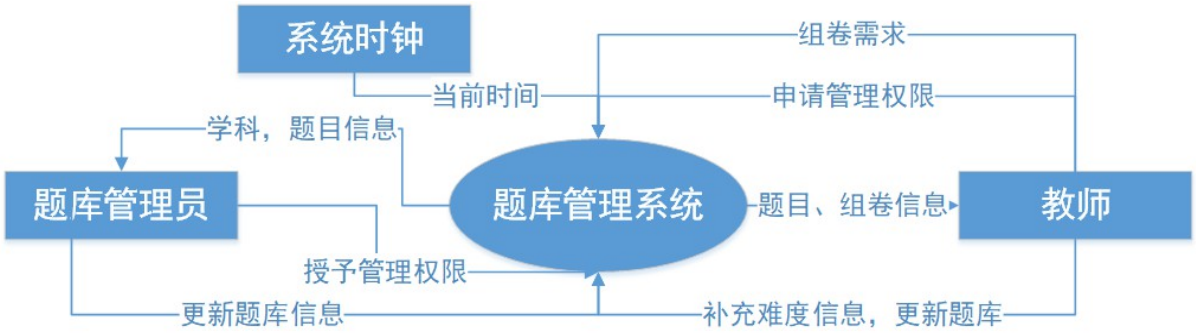

图 2: 题库管理系统顶层图

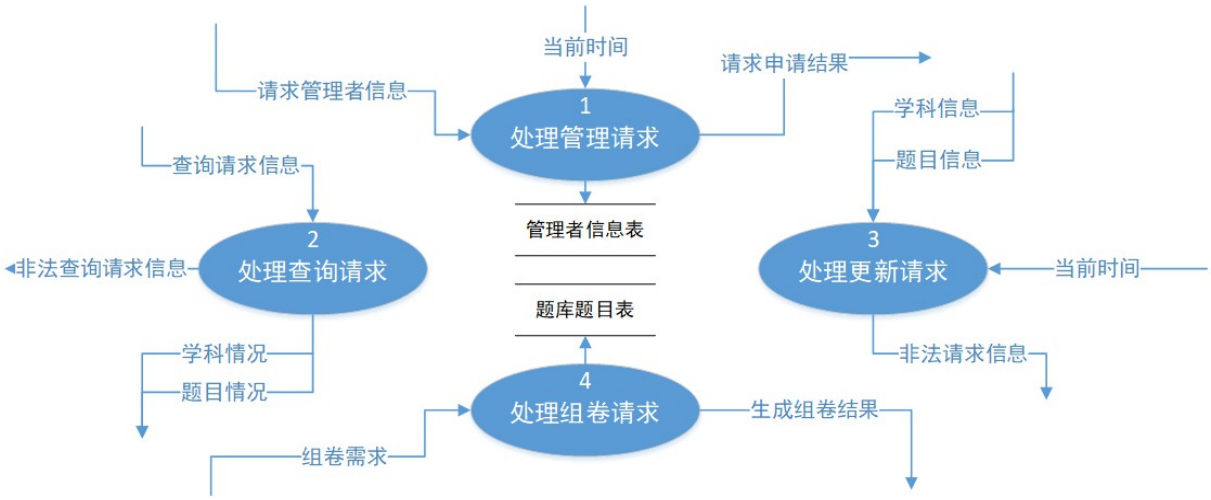

图 3: 题库管理系统0层图

3 逻辑结构设计

3.1 局部E-R图

图 4: 局部E-R图(1)

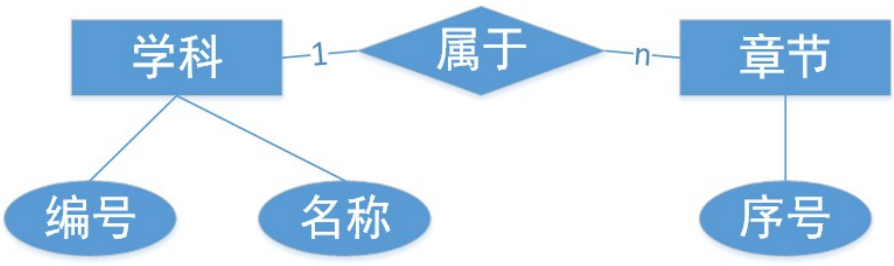

图 5: 局部E-R图(2)

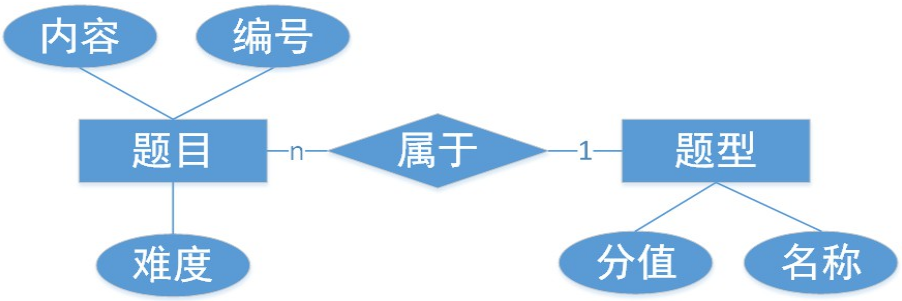

图 6: 局部E-R图(3)

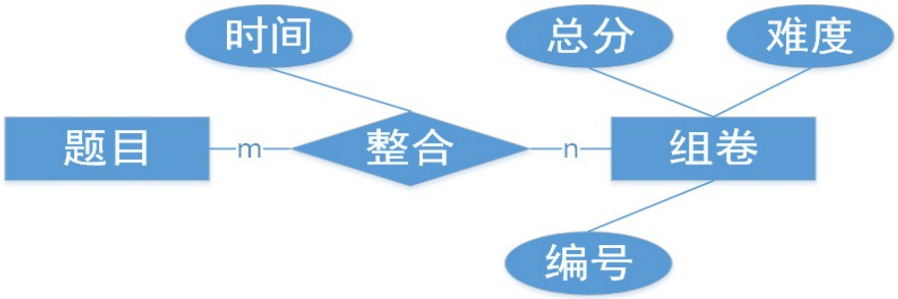

图 7: 局部E-R图(4)

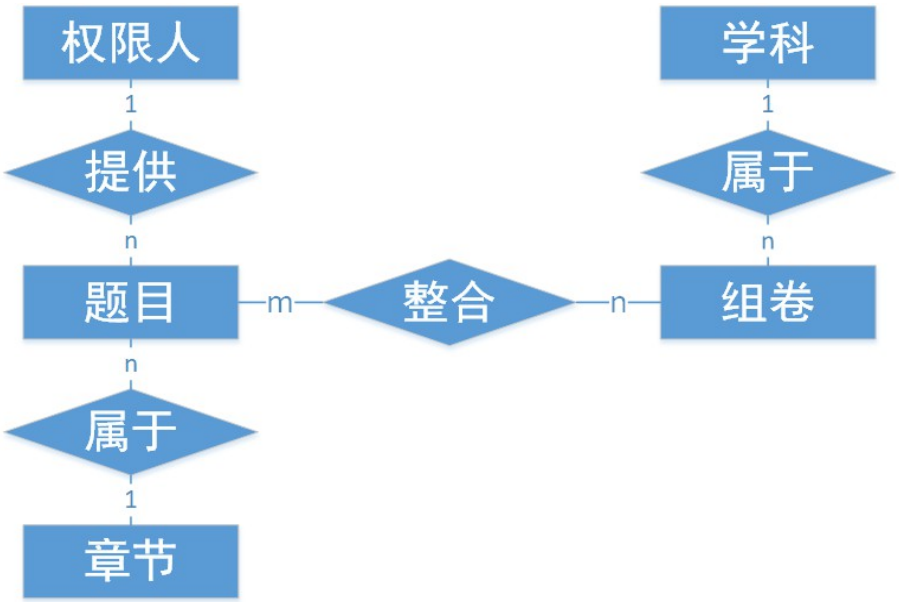

图 8: 局部E-R图(5)

3.2 全局E-R图

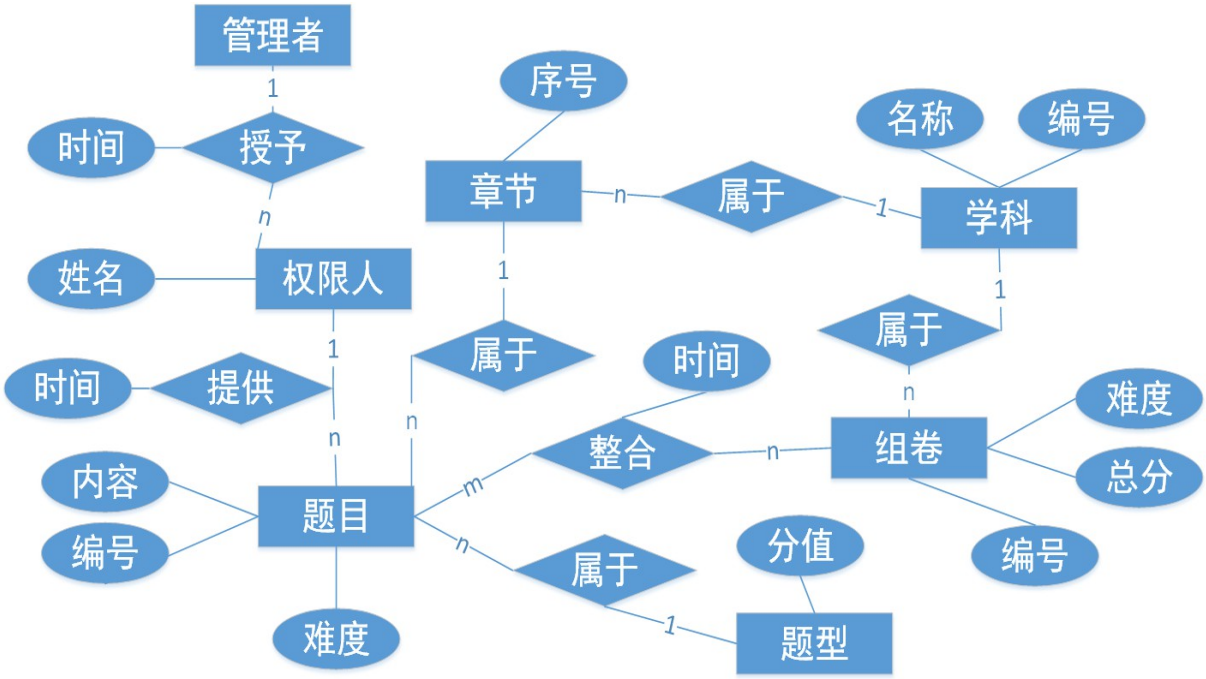

图 9: 全局E-R图

3.3 数据字典

表 2: 数据库表与关系模式

表名	意义	关系模式
Authority	权限	（姓名,授权时间）
Subject	学科	（编号,名称）
Chapter	章节	（序号,所属学科）
Type	题型	（编号，所属学科,名称,分值）
Question	题目	（编号,类型，所属章节,难度系数,操作人,时间,抽取次数）
Paper	组卷	（编号,学科,难度,总分,时间）

表 3: Authority表

列名	数据项	数据类型	允许*Null*值
Aname	姓名	CHAR(10)	否
Atime	授权时间	DATETIME	否

表 4: Subject表

列名	数据项	数据类型	允许*Null*值
Sno	编号	INT	否
Sname	名称	CHAR(15)	否

表 5: Chapter表

列名	数据项	数据类型	允许*Null*值
Cno	序号	CHAR(5)	否
Sno	所属学科	INT	否

表 6: Type表

列名	数据项	数据类型	允许*Null*值
Tno	编号	INT	否
Sno	学科	INT	否
Tname	题型	CHAR(10)	否
Tmark	分值	INT	否

表 7: Question表

列名	数据项	数据类型	允许*Null*值
Qno	编号	INT	否
Tno	题型	INT	否
Cno	章节	CHAR(5)	否
Qdiff	难度	INT	否
Aname	姓名	CHAR(10)	否
Qtime	更新时间	DATETIME	否
Qcount	抽取次数	INT	否
Qcon	内容	NVARCHAR(MAX)	否

表 8: Paper表

列名	数据项	数据类型	允许*Null*值
Pno	编号	INT	否
Sno	学科	INT	否
Pdiff	难度	INT	否
Pscore	总分	INT	是
Ptime	时间	DATETIME	否

4 数据库实现

4.1 创建数据库和数据表

4.1.1 创建和删除数据库

创建数据库：

```
CREATE DATABASE 题库管理系统;
```

如果需要可以删除数据库，前提是当前数据库没有被使用：

```
DROP DATABASE 题库管理系统;
```

4.1.2 创建和修改数据表

创建Authority表：

```
CREATE TABLE Authority
(Aname CHAR(10) PRIMARY KEY,
Atime DATETIME NOT NULL DEFAULT(GETDATE())
);
```

创建Subject表：

```
CREATE TABLE Subject
(Sno INT IDENTITY(1,1) PRIMARY KEY,
Sname CHAR(15) UNIQUE NOT NULL
);
```

创建Chapter表：

```
CREATE TABLE Chapter
(Cno CHAR(5) PRIMARY KEY,
Sno INT NOT NULL,
FOREIGN KEY (Sno) REFERENCES Subject(Sno)
);
```

创建Type表：

```
CREATE TABLE Type
(Tno INT IDENTITY(1,1) PRIMARY KEY,
Sno INT,
Tname CHAR(10) CHECK (Tname IN ('选择题','填空题','解答题','判断题','论述题','写作题','阅读题','听力题')),
Tmark INT NOT NULL CHECK (Tmark BETWEEN 0 AND 100),
FOREIGN KEY (Sno) REFERENCES Subject(Sno)
);
```

创建Question表：

```
CREATE TABLE Question
(Qno INT IDENTITY(1,1) PRIMARY KEY,
Tno INT NOT NULL,
Cno CHAR(5),
Qdiff INT NOT NULL CHECK (Qdiff BETWEEN 0 AND 10),
Aname CHAR(10) NOT NULL,
Qtime DATETIME NOT NULL DEFAULT(GETDATE()),
Qcount INT NOT NULL DEFAULT(0),
Qcon NVARCHAR(MAX) NOT NULL,
FOREIGN KEY (Cno) REFERENCES Chapter(Cno),
FOREIGN KEY (Tno) REFERENCES Type(Tno),
FOREIGN KEY (Aname) REFERENCES Authority(Aname)
);
```

创建Paper表：

```
CREATE TABLE Paper
(Pno INT IDENTITY(1,1),
Sno INT,
Pdiff INT NOT NULL CHECK (Pdiff BETWEEN 0 AND 10),
Pscore INT DEFAULT(100),
Ptime DATETIME NOT NULL DEFAULT(GETDATE()),
PRIMARY KEY (Pno,Sno),
FOREIGN KEY (Sno) REFERENCES Subject(Sno)
);
```

修改表操作：

```
ALTER TABLE 表名[] ALTER COLUMN 列名[] 新约束[];
```

删除表操作：

```
DROP TABLE 表名[];
```

4.2 数据操作

4.2.1 数据更新

对各表分别插入数据：

```
INSERT
INTO Subject(Sname)
VALUES('数学'),('语文'),('英语'),('物理'),('化学'),('生物'),('政治'),('历
    史'),('地理'),('军理');
```

```
INSERT
INTO Chapter(Cno,Sno)
VALUES('10001',1),('10002',1),('10003',1),('10004',1),('10005',1),('10006',1),('10007',1),('10008',1),('10009',1),('10010',1),
('20001',2),('20002',2),('20003',2),('20004',2),('20005',2),('20006',2),('20007',2),('20008',2),('20009',2),('20010',2),
('30001',3),('30002',3),('30003',3),('30004',3),('30005',3),('30006',3),('30007',3),('30008',3),('30009',3),('30010',3),
('40001',4),('40002',4),('40003',4),('40004',4),('40005',4),('40006',4),('40007',4),('40008',4),('40009',4),('40010',4),
('50001',5),('50002',5),('50003',5),('50004',5),('50005',5),('50006',5),('50007',5),('50008',5),('50009',5),('50010',5),
('60001',6),('60002',6),('60003',6),('60004',6),('60005',6),('60006',6),('60007',6),('60008',6),('60009',6),('60010',6),
('70001',7),('70002',7),('70003',7),('70004',7),('70005',7),('70006',7),('70007',7),('70008',7),('70009',7),('70010',7),
('80001',8),('80002',8),('80003',8),('80004',8),('80005',8),('80006',8),('80007',8),('80008',8),('80009',8),('80010',8),
('90001',9),('90002',9),('90003',9),('90004',9),('90005',9),('90006',9),('90007',9),('90008',9),('90009',9),('90010',9),
('11001',10),('11002',10),('11003',10),('11004',10),('11005',10),('11006',10),('11007',10),('11008',10),('11009',10),('11010',10);
```

```
INSERT
INTO Type(Sno,Tname,Tmark)
VALUES(1,'选择题',10),(2,'选择题',5),(3,'选择题',16),(4,'选择题',18),(5,'选择题',6),(6,'选择题',8),(7,'选择题',7),
(8,'填空题',9),(9,'填空题',8),(10,'填空题',1),(1,'填空题',13),(2,'填空题',11),(3,'填空题',8),(4,'填空题',16),(5,'填空题',4),
(1,'解答题',9),(3,'解答题',8),(5,'解答题',1),(7,'解答题',13),(9,'解答题',11),(2,'解答题',8),(8,'解答题',16),(6,'解答题',4);
INSERT
INTO Authority(Aname)
VALUES('黎洲波'),('吕梦轩'),('施妙辉'),('孔繁梓'),('红太阳'),('灰太狼');
```

```
INSERT
INTO Question(Tno,Cno,Qdiff,Aname,Qcon)
VALUES(1,'10001',3,'黎洲波','1+1='),
(12,'20001',3,'黎洲波','1+2='),
(3,'30005',3,'吕梦轩','1+3='),
(3,'10004',3,'吕梦轩','1+4='),
(3,'20005',3,'孔繁梓','1+5='),
```

```
(5,'70006',3,'孔繁梓','1+6=');
(8,'90008',3,'孔繁梓','1+7=');
(9,'50006',3,'黎洲波','1+8=');
(15,'40008',3,'施妙辉','1+9=');
(12,'30002',3,'施妙辉','1+10=');
(11,'20005',3,'施妙辉','2+1=');
(4,'10006',3,'施妙辉','3+1=');
(8,'11003',3,'施妙辉','4+1=');
(10,'11004',3,'黎洲波','5+1=');
(15,'50003',3,'黎洲波','6+1=');
(2,'40004',3,'施妙辉','7+1=');
(3,'60005',3,'施妙辉','8+1=');
(8,'70004',3,'黎洲波','9+1=');
(6,'90005',3,'黎洲波','1+1=');
```

更新数据：

```
UPDATE Question SET Qcon='1+2=' WHERE Qno=1;
```

4.2.2 建立查询

1.查询各种题型和章节的题目数量：

```
SELECT Tno,COUNT(*)
FROM Question
GROUP BY Tno;
```

查询结果如下：

	Tno	(无列名)
1	1	1
2	2	1
3	3	4
4	4	1
5	5	1
6	6	1
7	8	3
8	9	1
9	10	1
10	11	1
11	12	2
12	15	2

图 10: 结果

```
SELECT Cno,COUNT(*)
```

```
FROM Question
GROUP BY Cno;
```

查询结果如下：

	Cno	(无列名)
1	10001	1
2	10004	1
3	10006	1
4	11003	1
5	11004	1
6	20001	1
7	20005	2
8	30002	1
9	30005	1
10	40004	1
11	40008	1
12	50003	1
13	50006	1
14	60005	1
15	70004	1
16	70006	1

图 11: 结果

2.查询各个学科的题目数量，在此查询数学学科的题目数量：

```
SELECT COUNT(*)
FROM Question
WHERE Tno IN
(SELECT Tno
FROM Type
WHERE Sno=1
);
```

	(无列名)
1	2

图 12: 结果

4.2.3 创建视图

定义视图查询各门课程使用的题型：

```
CREATE VIEW TS_1
AS
SELECT Tname
FROM Type
WHERE Sno=1;
```

```
6 SELECT * FROM TS_1
```

创建视图结果如下：

	Tname
1	选择题
2	填空题
3	解答题

图 13: 结果

4.2.4 功能实现

1.运用触发器，在更改Question表时把时间更新为当前时间(此处为防止嵌套UPDATE数据，运用了IF判断)：

```
1 CREATE TRIGGER Update_Q ON Question
2 FOR UPDATE
3 AS
4 IF UPDATE(Qdiff) OR UPDATE(Tno) OR UPDATE(Aname) OR UPDATE(Qcount) OR
      UPDATE(Qcon)
5 BEGIN
6         UPDATE Question SET Qtime=GETDATE() WHERE Question.Qno=(SELECT Qno
              FROM inserted);
7 END;
```

```
1 CREATE TRIGGER Update_P ON Paper
2 FOR UPDATE
3 AS
4 IF UPDATE(Pdiff) OR UPDATE(Sno) OR UPDATE(Pscore)
5 BEGIN
6         UPDATE Paper SET Ptime=GETDATE() WHERE Paper.Pno=(SELECT Pno FROM
              inserted);
7 END;
```

2.自动抽取题组成套题，习题每抽取一次要使习题抽取次数加1,此处建立临时表PQ用于储存题目信息（其中A_1为选择题题号，其中B_1为填空题题号）：

```
1 IF (SELECT COUNT(*) FROM Paper)=0
2         CREATE TABLE PQ
3         (NO INT IDENTITY(1,1) PRIMARY KEY,
4         A_1 INT,
5         B_1 INT);
```

a.在抽取题库中题目时，对Question表中的Qcount属性进行加1处理，利用触发器实现：

```
CREATE TRIGGER Insert_PQ1 ON PQ
AFTER UPDATE
AS
IF (SELECT B_1 FROM inserted) IS NULL
BEGIN
UPDATE Question SET Qcount=Qcount+1 WHERE Question.Qno=(SELECT A_1 FROM
    inserted);
END;
CREATE TRIGGER Insert_PQ2 ON PQ
AFTER UPDATE
AS
BEGIN
UPDATE Question SET Qcount=Qcount+1 WHERE Question.Qno=(SELECT B_1 FROM
    inserted);
END;
```

b.在此以数学学科抽取题组为例，进行抽取题目处理，计算试卷总分传递给Paper表，并将题组的题目序号信息传递给临时表PQ:

```
CREATE TRIGGER Insert_P1 ON Paper
FOR INSERT
AS
IF (SELECT Sno FROM inserted)=1
BEGIN
INSERT INTO PQ(A_1) VALUES(Null)
UPDATE PQ SET A_1=
          (SELECT TOP 1 Qno
          FROM Question
          WHERE Tno IN(SELECT Tno FROM Type WHERE Tname='选择题' AND Sno=1)
          ORDER BY NEWID())
          WHERE NO=(SELECT Pno FROM inserted)
UPDATE PQ SET B_1=
          (SELECT TOP 1 Qno
          FROM Question
          WHERE Tno IN(SELECT Tno FROM Type WHERE Tname='填空题' AND Sno=1)
          ORDER BY NEWID())
          WHERE NO=(SELECT Pno FROM inserted)
UPDATE Paper SET Pscore=
          (SELECT Tmark
          FROM Type WHERE Tname='选择题' AND Sno=1)+
          (SELECT Tmark
          FROM Type WHERE Tname='填空题' AND Sno=1)
```

```
24          WHERE Paper.Pno=(SELECT Pno FROM inserted)
25  END;
```

在Paper表中插入数据：

```
1  INSERT
2  INTO Paper(Sno,Pdiff)
3  VALUES(1,5);
```

操作结果如下：

	Qno	Tno	Cno	Qdiff	Aname	Qtime	Qcount	Qcon
1	1	1	10001	3	黎洲波	2018-06-28 17:23:58.587	1	1+1=
2	2	12	20001	3	黎洲波	2018-06-28 17:21:32.130	0	1+2=
3	3	3	30005	3	吕梦轩	2018-06-28 17:21:32.130	0	1+3=
4	4	3	10004	3	吕梦轩	2018-06-28 17:21:32.130	0	1+4=
5	5	3	20005	3	孔繁梓	2018-06-28 17:21:32.130	0	1+5=
6	6	5	70006	3	孔繁梓	2018-06-28 17:21:32.130	0	1+6=
7	7	8	90008	3	孔繁梓	2018-06-28 17:21:32.130	0	1+7=
8	8	9	50006	3	黎洲波	2018-06-28 17:21:32.130	0	1+8=
9	9	15	40008	3	施妙辉	2018-06-28 17:21:32.130	0	1+9=
10	10	12	30002	3	施妙辉	2018-06-28 17:21:32.130	0	1...
11	11	11	20005	3	施妙辉	2018-06-28 17:23:58.587	1	2+1=
12	12	4	10006	3	施妙辉	2018-06-28 17:21:32.130	0	3+1=
13	13	8	11003	3	施妙辉	2018-06-28 17:21:32.130	0	4+1=
14	14	10	11004	3	黎洲波	2018-06-28 17:21:32.130	0	5+1=

图 14: 结果1

	Pno	Sno	Pdiff	Pscore	Ptime
1	1	1	4	23	2018-06-28 17:23:58.590

图 15: 结果2

	NO	A_1	B_1
1	1	1	11

图 16: 结果3

5 结束语

5.1 总结与反思

题库的管理系统的设计和维护有赖于MSSQL的各种约束条件，限制使用谓词，限制操作人等等，而所有的限制都是靠定义来实现的，在处理题库需要的运行功能时，总会以一般编程语言的思维习惯，比

如后续扩展项：计算组卷的难度加权，若是定义了全局变量，代码量会少很多，在上网查询的过程中发现SQL确有像这样的操作，只是说我们的教材未做过多的要求。而同时当定义触发器时新增加（或修改），删除的行信息保存在临时表Inserted和Deleted中，此时要在原表中找到改动行的信息，必须进行表的等值连接，此处想到Pyshon里面的Pandas库里面改动单行操作或只是单纯停留在改动层面上去处理某些数据，而数据库在改动某数据时通过事件触发，去影响其他表中的数据或者约束自身改动的信息，更加方便于整个库的管理。

当建表时，如果没有考虑到要求实现的各种约束，比如章节和题目的关系，题型和科目的关系，其中建立的表级完整性约束有些复杂，当在建立E-R图和转换为关系模式的时候就必须要列清楚这些项，否则便会像此次设计数据库建表时增添很多不必要的麻烦。建立视图好比其他语言里对表格进行切片操作，但本身没有行列索引项导致视图的建立仍然依赖着SELECT谓词和WHERE子句，所以在建立时也要服从标准语法限制。

5.2 后续的问题

按此触发器编写规则，当进行组卷随机抽取题库中指定要求的题时都对应了不同的触发器类型，若能通过函数，对触发器里的设置参数进行修改，那么便可以解决这一问题。同时，组卷触发器存在一定的问题，即使用NEWID()制造的ORDER并没有随着触发次数的改变而改变，该处还有待探索，但是仅对于一次抽取操作，该触发器表现正常，即满足了题目要求。

数据库系统在数据的管理上相对于传统的数据处理语言会有一定的优势，但其他语言所表现出的数据处理和运算能力在各种谓词的使用限制里变得十分繁琐，即使具备了做到各种聚合和统计的能力，SQL的语言框架决定了在使用过程中的各种不方便的问题，例如该题库管理系统抽取题组的后续问题，即计算试卷难度加权，以及对于PQ表中题组的处理。此外，在建立视图解决查询每门课程使用题型操作时，由于GROUP BY的聚合函数使用限制，不能依据Sno对Type表中的Tname进行分类，查阅资料发现处理该类问题是这么进行的:

用户表(姓名,编号,爱好)user(name,id,hobby),里面的数据有：

name	*id*	*hobby*
张三	001	篮球
张三	001	电影
李四	002	足球

表 9: 数据

现在要求写一SQL语句，使查出结果为：

name	*id*	*hobby*
张三	001	篮球，电影
李四	002	足球

表 10: 结果

方案一

```
1 IF OBJECT_ID('USER') IS NOT NULL
```

```
DROP TABLE [USER]
GO
CREATE TABLE [USER]( NAME VARCHAR(10) ,ID VARCHAR(5) ,HOBBY VARCHAR(10) )
INSERT INTO [USER]
SELECT '张三','001','篮球' UNION ALL
SELECT '张三','001','电影' UNION ALL
SELECT '李四','002','足球'
SELECT NAME,ID ,STUFF((SELECT ','+HOBBY FROM [USER] T2
WHERE T2.NAME=T1.NAME FOR XML PATH('')),1,1,'')
FROM [USER] T1
GROUP BY NAME,ID
```

方案二

```
CREATE TABLE tb( NAME VARCHAR(10) ,ID VARCHAR(5) ,HOBBY VARCHAR(10) )
INSERT INTO tb
SELECT '张三','001','篮球' UNION ALL
SELECT '张三','001','电影' UNION ALL
SELECT '李四','002','足球'
GO
CREATE FUNCTION dbo.f_str(@id int)
RETURNS VARCHAR(8000)
AS
BEGIN
DECLARE @r VARCHAR(8000)
SET @r = '' SELECT @r = @r + ',' + hobby
FROM tb
WHERE id=@id
RETURN STUFF(@r, 1, 1, '')
END
GO
SELECT name, hobby = dbo.f_str(id) FROM tb GROUP BY name,id
DROP TABLE tb
DROP FUNCTION dbo.f_str
```

上述方案涉及到为学过的语句以及函数的定义和调用（好像跟一般语言的函数定义差不多，只希望不要限制太多），但是解答的复杂嵌套着实令人望而生畏，如果能将间接性和约束的严密性结合在一起，数据库的功能可能会更多一些，而且受众会更广泛，甚至可大面积使用数据库来结合前端的查询操作，再结合统计数据分析，如此便可以发挥数据库体系更大的效用。