基于拆分存储模式的题库系统①

肖 刚

(宁波广播电视大学 信息与教学资源中心, 宁波 315016)

摘 要: 通过比较分析现有题库系统的存储模式, 提出拆分存储模式, 即将试题参数拆分为试题公共基础部分和题型专有部分. 基于该模式采用面向对象的方法, 设计实现了题库系统的试题管理和组卷算法, 小规模的实验结果表明组卷算法是正确和有效的.

关键词: 拆分存储; 题库; 组卷算法

Item Bank System Based on the Split Storage Mode

XIAO Gang

(Information and Instructional Resources Center, Ningbo Radio and Television University, Ningbo 315016, China)

Abstract: Through comparing and analyzing the existing test bank system storage mode, this article puts forwards the split storage mode, viz., the test parameters is splited into public infrastructure part and the kinds of questions proprietary parts. Based on the model by using the oriented object method, test management and Algorithmic of test bank system is designed and implemented, small scale experiments show that the algorithm is corrective and effective.

Key words: split storage mode; item bank; combining algorithmic

题库(ITEM BANK)是“按照一定的教育测量理论, 在计算机系统中实现的某个学科题目的集合”, 计算机题库是计算机考试系统的基础[1]. 完整意义上的题库, 具有试题管理功能、组卷、统计分析功能[2], 试题的存储模式决定了题库各项功能的实现方式, 探索和研究结构合理、维护方便的试题存储模式, 是题库系统需要面对的共同难题. 本文在研究现有试题存储模式的基础上, 提出了拆分存储模式, 将试题参数拆分为公共基础部分和题型专有部分, 并应用面向对象的方法, 给出了基于 Linq 的试题管理和组卷的设计与实现.

1 试题存储模式

现代远程教育资源建设规范(试行, 2000.5 版)对题库的试题参数有详细的说明, 试题参数共有 21 项, 但由于没有考虑题型的差别, 该规范的试题参数说明不能直接应用于试题存储设计, 主要表现为: 首先, “试题正文”参数为 Memo 类型, 但不同题型的题干是不同的, 如阅读理解一般有 5 个题干, 简答题仅 1 个题干, 简单的 Memo 类型很难灵活显示试题; 其次, “参考答案”参数也为 Memo 类型, 对于作文题该类型是合适的, 但对于答案仅一个字符单选题, 字符类型更合适; 最后, “建议考试得分”参数直接与试题相关, 没有考虑试卷总分和试题总数的情况. 因此, 该规范只能作为试题存储设计的指导, 在实际设计试题存储时, 要对规范试题参数进行修改, 主要的修改是对“试题正文”参数, 进行细分, 细分方案根据题型变化, 例如, 单选题的试题正文会被分解成试题主干以及 4 个备选答案.

1.1 现有试题存储模式

“试题正文”被细分后, 各题型的试题参数结构有较大的差异, 按照试题存储模式分, 现有的试题存储模式大致可以分为两类. 一类是集中存储模式, 所有题型的试题数据集中保存在单个数据表中, 数据表的字段集是所有题型试题参数的并集. 该方法的优点是试题数据集中存储, 组卷和统计分析只需访问单个数据表, 缺点有: (1)字段冗余, 特定题型的某些试题参数不适用其它题型, 例如, 单选题型的 4 个备

① 收稿时间:2012-10-11;收到修改稿时间:2012-11-12

选答案字段, 对填空、判断等题型是没有意义的; (2)字段存储空间设置冗余, 对于多个题型共有的字段, 字段存储空间的设置要满足所有题型中最大的存储要求; (3)降低数据的自解释性, 相同试题参数对不同题型有不同的语义和判分标准, 例如, 正确答案同为“T”试题记录, 对于判断题表示真假, 对于单选题, 就没有任何意义.

另一类是独立存储模式, 不同题型的试题分别存储在不同的数据表中, 数据表的字段集是对应题型的试题参数集, 数据表的数量等于题型数量. 该模式解决了集中存储模式的问题, 但在组卷和统计分析时, 要访问所有的试卷数据表, 增加了代码开发和维护的难度, 其次, 尽管各题型试题参数结构存在差异, 但有部分试题参数是相同的, 对相同部分的试题参数进行操作的代码也是相同的, 独立存储模式在减少数据存储冗余的同时, 导致代码的冗余.

1.2 拆分存储模式

在综合比较现有试题存储模式的基础上, 本文提出拆分模式, 将试题参数拆分为试题公共基础部分和题型专有部分, 其中公共基础部分是所有题型的共有参数集, 包括试题编号、课程编码、题型、知识点、难度、试题正文、出题人、抽取次数等参数, 其中题型参数区分各试题题型, 题型专有部分包括 id, 备选答案、参考答案和试题编号, 其中试题编号是公共基础与题型专有部分的关联字段, 将两部分内容组合成一个完整的试题, 备选答案的个数和类型、参考答案的类型与具体题型相关. 根据此拆分模式设计的数据库实体关系如图 1 所示, 数据库包括一个试题公共基础数据表和多个题型专有数据表.

2 试题管理

试题管理是题库系统的基本功能, 包括试题的增加、删除、修改、查找[2], 本系统的试题管理采用传统的 3 层结构, 应用面向对象的设计方法, 数据访问应用 Linq 技术.

2.1 Linq 简介

Linq 是 Language Integrated Query 的缩写, 翻译成中文就是语言集成查询. LINQ 是一系列编程接口, 借助 Linq 技术, 可以以一种统一的方式查询不同类型的数据. Linq 在对象领域和数据领域之间架起了一座桥梁[3], 为面向对象技术开发提供了便利.

2.2 基于 Linq 的分层设计

基于 Linq 的试题管理分为三层: DataLinq 层、试题操作层、试题显示层, 软件层次结构如图 2 所示.

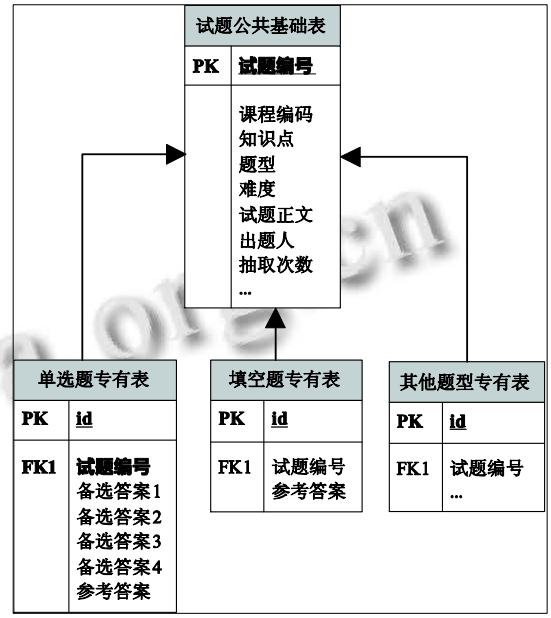

图 1 题库试题关系图

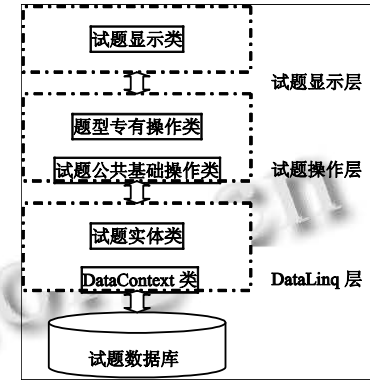

图 2 基于 Linq 的试题管理的类层次结构

2.2.1 DataLinq 层

DataLinq 层实现关系数据库数据与对象之间的映射, 包括试题公共基础实体类、各题型专有实体类和 DataContext 类, 代码由系统自动生成. 每个试题数据表映射为一个实体类, 每个实体类属性的类型、属性与数据表字段相同, 每个实体类对象与一条数据库记录对应. DataContext 是.NET Framework 3.5 中新增的一个类, 是实体类与数据库之间的桥梁, 用于从数据库中返回简单数据或实体类集、增加、修改或删除的实体写入数据库. 在 Data Linq 层, 都需要从 Data Context 继承一个新类, 与系统所用的数据库对应. 本

系统数据存储使用关系数据库 SQL Server, DataLinq 层的数据访问应用框架是 Linq to Sql.

2.2.2 试题操作层

试题操作层的主要功能是对试题的增加、删除、修改、查找等基本操作, 操作层包括试题公共基础操作类和各题型专有操作类, 题型专有操作类继承试题公共基础类, 用统一建模语言 UML(unified modeling language)设计的试题操作类如图 3 所示.

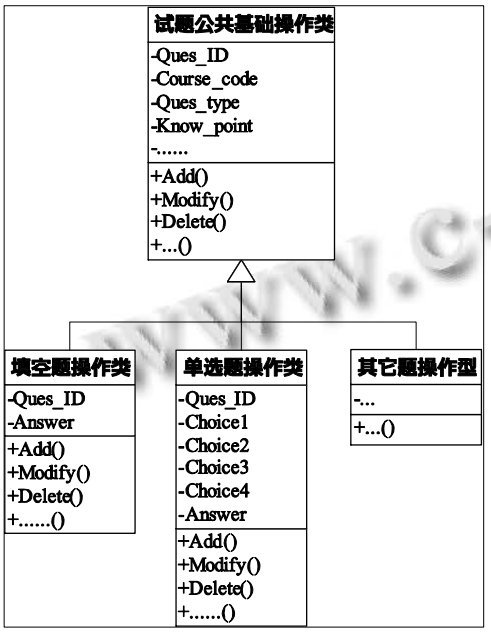

图 3 试题操作类图

试题公共基础类的属性包括试题公共基础部分的参数, 类方法 Add()、Modify()、delete()实现对试题公共基础部分的基本操作, 这些类方法的类型都是公开的虚函数; 题型专有类的属性包括各题型的专有部分参数, 类方法 Add()、Modify()、delete()对基类中的虚函数重新定义, 完成对试题题型专有部分的操作, 并调用基类中的函数, 完成对试题公共部分参数的操作, 从而实现对完整试题的操作.

在确定题型和题型所属试题参数的前提下, 以填空题为例, 增加一道试题的过程为:

(1) 调用工厂类 ClassFactory 对象的构造方法 create(), 该方法根据输入的题型参数创建填空题专有操作类 QB_FillSpaceBll 的对象;

(2) 调用新建的 QB_FillSpaceBll 类对象的 Add()方法, 该方法首先在填空题专有表中增加一条记录, 然后调用基类 QuestionBaseBll 的 Add()方法, 在试题公共基础表中增加一条记录, 从而完成一道填空题所有试题参数数据的保存.

图 4 为增加一道填空题的顺序图.

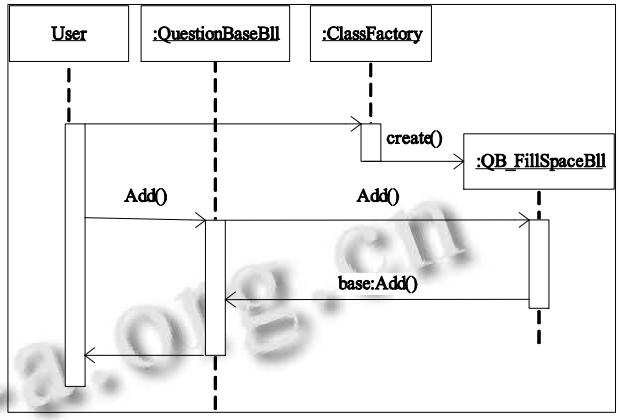

图 4 增加填空题的顺序图

3 组卷

组卷是指按照一定的组卷条件, 组成符合学生和教师使用的试卷. 组卷条件就是一些试题查询参数, 系统将根据这些参数抽出最适合参数要求的试题, 组成能够实际使用的试卷, 定义这种查询参数以及对这些参数进行变换算法, 称之为组卷策略. 完整的组卷策略应该由三部份组成: 试题属性项定义、组卷参数的定义、变换算法的说明. 快速组卷需要设置试卷的一些整体属性参数和题型结构参数[2].

3.1 组卷算法

本系统采取快速组卷方式, 题型结构参数包括试卷各题型的试题数、考察知识点、单题分值, 组卷策略由课程责任教师管理并存储在单独的数据表中.

题库中各知识点对应的试题数量不同, 简单随机抽取试题会导致试卷中各知识点的考题分布不均, 解决的方法是保证各知识点的试题在试卷中的比例与在题库中的比例一致. 本系统的组卷采取分步抽取的方法, 按照课程、题型、知识点的次序, 逐步加强筛选条件, 缩小待抽取试题集, 以循环的方式建立试卷试题集. 在拆分存储模式中, 组卷所需的主要题型结构参数都保存在试题公共基础数据表中, 在筛选试题的过程中, 只需要访问该数据表. 组卷算法(Gener_paper)的伪代码如下, 其中, 输入参数 course 是组卷的课程代码, 输出是满足组卷策略的试题集 R:

```
Function Gener_paper(course)
{
```

R＝｛｝;//R 是初始试卷试题集合

TC=｛tc1,tc2,…,tck｝;//TC 是课程 course 组卷策略中的题型集

ForEach 题型 type in TC

｛

令 KC＝｛k|k∈组卷策略集 && k. course=course && k.type=type｝; //KC 是课程 course、题型 type 的考察知识点集

令 Q＝｛q|q∈基本题库表 && q.course＝course && q.type=type && q. knowledgeNode in KC｝;

令 S＝｛｝;

ForEach 知识点 knowledgeNode in KC

｛

令 T ＝ ｛ q|q ∈ Q&& q. knowledgeNode ＝ knowledgeNode｝; // T 是 Q 中覆盖特定知识点的集合

kt＝|T| / |Q| 取整; //kt 是试卷中该知识点按比例的试题数量;

For (i=0;i<kt;i++)

从集合 T 中简单无放回随机抽取试题 q, 加入集合 S 中: S←S∨q

｝

While (|S|<题型 type 要求的试题总数)

从 Q 中随机抽取未被选中的试题 q, 加入集合 S 中: S←S∨q;

将集合 S 中的记录顺序打乱顺序, 加入到集合 R 中;

｝

返回集合 R;

｝

3.2 组卷的实验数据及分析

为了验证组卷算法抽题的正确性和有效性, 本文模拟小规模题库进行组卷实验, 测试各知识点抽题的比例和分布情况.

实验数据: 题库有 18 道试题, 包括 3 个知识点, 要求每次生成 5 份试卷, 每份试卷包括 8 道题, 组卷策略覆盖 3 个知识点, 题库中试题分布如表 1 所示.

表 1 题库中试题分布图

	知识点 1	知识点 2	知识点 3
试题总数	11	5	2
试题比例	0.6 1	0.28	0.11

实验共进行了 7 轮, 各知识点抽题数及比例的统计数据如表 2 所示, 7 轮后按被抽取次数对试题进行统计, 数据如表 3 所示. 实验数据表明, 各知识点试题在试卷中的比例与在题库中的比例一致, 各知识点抽取的试题的分布均匀, 没有抽不到的题, 并且随着实验次数的增加, 试题被抽取次数的差距比例会进一步缩小. 因此, 此算法是可行的、有效的.

表 2 各轮知识点抽题数及比例

轮次	知识点 1		知识点 2		知识点 3		试题总数
第 1 轮	23	0.57	12	0.3	5	0.12	40
第 2 轮	48	0.6	24	0.3	8	0.1	80
第 3 轮	71	0.59	39	0.32	10	0.08	120
第 4 轮	95	0.59	52	0.32	13	0.08	160
第 5 轮	115	0.67	68	0.34	17	0.09	200
第 6 轮	141	0.59	79	0.33	20	0.08	240
第 7 轮	164	0.58	92	0.33	24	0.09	280

表 3 试题抽取分布数据

抽取次数	9	13	14	15	16	17	18	20	21
试题数	3	1	1	1	4	2	4	1	1

4 结语

拆分存储模式的题库系统有下列优点: (1)在数据存储方面减少了存储冗余, 包括考试元素冗余和数据存储类型冗余; (2)在代码维护方面, 试题公共基础操作类负责试题公共元素的管理维护, 提高了代码的可读性和可维护性, 增加新的题型只需解决题型的专有部分, 较少代码开发量; (3)在组卷和统计分析方面, 试题公共基础数据表存储了试题的课程代码、知识点、题型、抽取次数等关键字段, 在组卷和数据分析的过程中, 只需访问该数据表, 不需要访问其它的数据表, 并且增加新的题型后, 不需要修改组卷和统计分析的源代码, 保证了系统的稳定性.

基于拆分存储模式的题库系统为考试系统的开发打下了良好的基础, 但在本文介绍的组卷算法中, 只考虑了知识点分布, 对试题难度和区分度没有考虑, 在进一步的开发中要考虑它们.

参考文献

1 张厚粲,刘昕.考试改革与标准参照测验.沈阳:辽宁教育出版社,1992.

2 余胜泉,何克抗.网络题库系统的设计与实现.中国远程教育, 2000(9):53–57.

3 解本巨,李宗颜,宫文生.LINQ 从实践到项目实战.北京:化学工业出版社,2010:2–3.