诚 信 声 明

我声明，所呈交的毕业论文是本人在老师指导下进行的研究工作及取得的研究成果。据我查证，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得其他教育机构的学位或证书而使用过的材料。我承诺，论文中的所有内容均真实、可信。

毕业论文作者签名：　　签名日期：　年　月　日

基于知识的专业选取决策工具的实现

[摘 要]

智能导学系统可以给学习者提供即时的及个性化的指导[1]。通常情况下，这种指导可以在网络上完成，不必面对面进行。此前，[3]提出了一个一阶逻辑推理形式系统模型 KMCD，以不确定推理为基础，针对在校生或远程学习者的需要，根据其学习能力辅助其进行专业选取。以[3]为基础，本文使用逻辑程序设计语言 Prolog 实现了一个专家系统，以四个学年的本科毕业生数据为分析对象，对 KMCD 模型进行验证。根据这些数据，本文分析了具体参数对运行结果有效性的影响，并针对 Prolog 语言的特性进行了优化，提高了程序执行效率，使得系统具备了实用性和可扩展性。此外，基于 ThinkPHP 网络架构，本文设计了在线系统供远程用户访问。

[关键词] 专业选取；决策支持；基于知识的系统；一阶逻辑；Prolog

A Knowledge-Based Practical Tool For Major-Choosing Decision Making

Abstract：

Intelligent tutoring system (ITS) aims to provide the learners immediate and customized instructions for their studies[1], usually without intervention from a human teacher. Based on the first-order logic model proposed in [3], an expert system is built in this paper to assist college students or remote learners in choosing their majors according to their learning abilities. The system is written in the logic programming language Prolog. It verifies the correctness of the form model [3], with the graduates from 4 academic years as its input data. With respect to the data, the system is optimized and the arguments are adjusted. In addition, an online system is constructed based on the ThinkPHP network architecture, which enables the remote learners to choose their suitable majors at home.

Keywords： Major choosing; decision making; knowledge-based system; first-order logic; Prolog

目　　录

1 绪论

1.1 选题背景及意义

1.1.1 背景

现代的智能化远程教育系统中，对学生学习的支持是一个很重要的研究领域。帮助学生做课程规划是智能导学系统的核心技术之一，它的目的是根据学生的个人情况为其规划学习内容，寻找达到学习目标的“最优”路径[2]。但是我们知道，学生确定其学习内容的前提是明确自己所学的是什么专业，而选择专业恰好是缺乏专业人士指导的网络学生最感到困惑的问题之一。目前，一些中国院校已推行大类招新政策，即入校后第一年不分专业，一年后根据志愿和成绩原则和个性化人才培养模式选择专业。这一政策在国外院校中十分常见。部分学生在修读完大一的课程后，也会有进修双学位（辅修）或转专业的意向。另外，一些远程的学习者，如参加成人教育、大规模开放网络课程（MOOC）等课程的网络学生，也希望在缺乏专业老师面对面沟通指导的机会下能通过远程导学系统选择合适自己的专业和学习方向。这就为专业规划系统提供了需求。

关于专业规划系统的设计，此前已有论文提出相关系统模型[3]。该模型利用学院数据库中过往毕业生的知识，通过不确定推理为在读学生推荐适合他们的专业。然而，[3]仅仅提出了理论框架，未涉及这种系统模型的实现，也缺乏实验数据支持。不过，只要借助[3]的推理方法，结合实际使用情况对模型的推理流程做出一些改动，便可开发出具有实用性的专业决策选取工具。

1.1.2 选题意义

本文基于[3],利用 Prolog 语言实现了一个具有实用性的自动决策系统 KMCD，并将其进一步开发为可访问的线上系统，且验证了该模型的可行性。此系统可投入实际使用，能帮助学院在读学生选择合适自己的专业，也可为各类远程学习者提供学习方向指导，并为日后类似的推荐系统提供设计与开发的参考。

1.2 项目概要

本文首先简要介绍了一个基于一阶逻辑的规则及推理的模型。该模型来源于指导老师发表的一篇论文[3]。在此模型的基础上，本文使用 Prolog 语言开发了一个专家系统，该系统通过 ODBC 建立了到 DBMS 的连接。随后，本文采用 ThinkPHP 框架构建网络部分，完成了 KMCD 网站的开发。在实现过程中，本文结合 Prolog 语言的特性，并考虑了实际使用时的情况，对系统作出算法及编程上的优化，并以用户友好的原则对网站功能作出了调整。

本文还涉及对 KMCD 系统的验证及数据分析。通过向系统导入四个年级毕业生的数据，本文对网站得出的推荐结果进行分析说明，并确定了使系统性能达到最优的系统参数，从而验证了系统的正确性与实用性。

1.3 系统架构与环境

1.3.1 推理系统架构

KMCD 系统是一个专家系统。专家系统可以对人类的思维方式进行模拟，一般分为两个部分：用于对规则和与规则有关的事实进行存储的记忆机构

——“知识库”，和根据输入数据对规则进行选择并执行的控制机构——“规则解释程序”。在专家系统的运行过程中，规则解释程序不断与用户交换信息，选择满足条件的规则执行，直到得到对问题的明确解释为止。

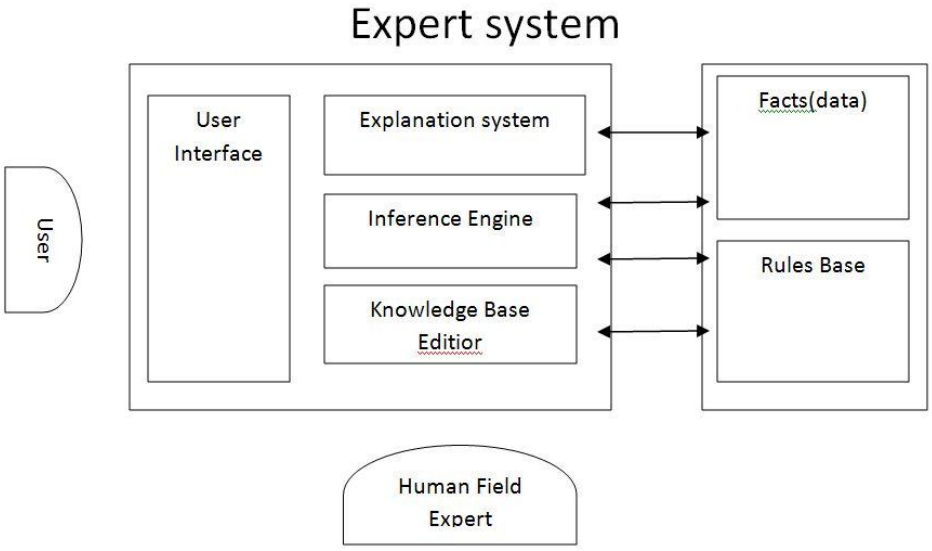

图 1.3-1 简单的专家系统模型

1.3.2 线上系统架构

KMCD 线上系统的开发基于开源框架 ThinkPHP。ThinkPHP 的设计模式为 MVC，是一种将应用程序的逻辑层和表现层进行分离的方法。MVC 实现了 Web 系统的职能分工，其各部分功能为：

模型（M）：实现程序应有的功能，管理数据和设计数据库。由 Model 类定义。

控制器（C）：转发、处理请求。由应用控制器和 Action 控制器负责。

视图（V）：实现图形界面。由 View 类和模板文件组成。

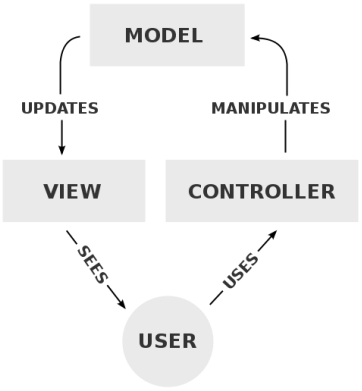

图 1.3-2 MVC 组件之间的典型合作

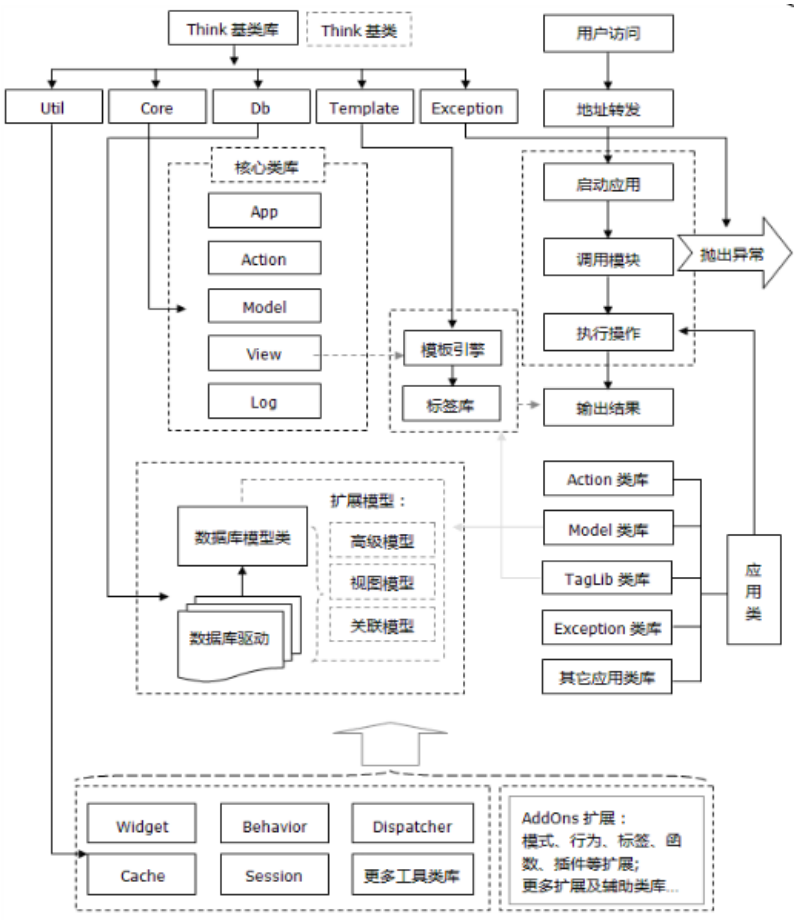

图 1.3-3 ThinkPHP 框架执行流程图

使用 ThinkPHP，可以方便、快捷地开发网站。

1.3.3 开发工具与运行环境

表格 1.3-1 开发工具与运行环境

开发环境		运行环境		
名称	描述	名称	描述	
开发工具	SWI-Prolog	服务器端环境	服务器端运行环境	OS Windows Service2003
	ThinkPHP框架		应用服务器	Apache httpd 2.2.29
	Sublime Text 2		数据库	MySql 5.6.21
	Navicat Premium		PHP	PHP 5.6.3
开发语言	Prolog、PHP、html、SQL等	客户端环境	客户端软件环境	IE6以上、Chrome、FireFox、Opera等主流浏览器

Prolog 为 KMCD 实现功能使用的主要语言。Prolog 具有的以下特征，使得它成为开发专家系统的较为理想的语言[4]：

1. 专家系统所需要的知识——事实与规则和 Prolog 中的事实与规则直接对应。专家系统中规则的条件可以表示成 Prolog 规则的身部的子系统，而它的结论，则可以表示成 Prolog 的规则头部。
2. 专家系统规则解释程序对规则选择与执行对应于 Prolog 中对规则的匹配深度与深度优先的问题求解。即 Prolog 自身的问题求解机制可以完成规则解释程序的功能。

由此可见，Prolog 不仅解决了知识——规则与事实的表示与存储问题，还解决了利用知识进行推理的问题。对专家系统的开发主要工作成为向 Prolog 输入规则与事实以及编写用户接口的工作。从这个意义上说 Prolog 提供了独立于具体应用领域的知识表示方式与推理的工具。

Sublime Text 2 为开发主要使用文本编辑器。它通过包（Package）扩充本身的功能，支持多种语言语法高亮、纠错，并能根据用户习惯进行多种个性化设定。

MySQL 是一款性能高、成本低、可靠性好关系数据库管理系统。Navicat

Premium 是一款可视化的数据库管理工具。由于 MySQL 本身不提供可视化管理功能，配合 Navicat 的使用能够方便直观地进行数据库设计及开发。

XAMPP 是一个把 Apache 网页服务器与 PHP、Perl 及 MySQL 集合在一起的安装包。本次开发使用此安装包提供的软件建立网页服务器。

2 KMCD 模型介绍

本章介绍了构建 KMCD 系统的模型。此模型来源于论文[3]。

2.1 KMCD 模型语言定义

KMCD 模型所使用的语言是一阶逻辑语言。它包含 4 种常量：学生学号 stu；课程编号 c；专业编号 m；学生取得的课程成绩 y。

4 种谓词：三元谓词 $get\_mark(stu, c, y)$，表示当前学生在课程中取得成绩；二元谓词 $choose\_major(stu, m)$，表示当前学生选择了专业；二元谓词 $take\_course(stu, course)$，表示当前学生选修了课程；二元谓词 $pass\_course(stu, course)$，表示当前学生在课程取得了及格。

语言还包括两种函数符号+和-，表示课程分数的加与减。

2.2 KMCD 推理系统

定义 KMCD 的公理集和证据集为 K 和 E。

KMCD 有以下推理法则：

(IR) 由 α 推断出 $\beta \to \beta$ 且 α，其中 α 和 β 是任意公式。

系统的第一步是找出与 $cstu$ 学习能力相近的毕业生 $fstu$。实现这一步的所需公理如下：

1) $get\_mark(cstu, c, y) \to get\_mark(fstu, c, y')(y' \in [y-\delta, y+\delta])$

   表示相似毕业生为在当前学生已修课程中取得落在 $[y-\delta, y+\delta]$ 区间的成绩的毕业生。

2) $take\_course(cstu, fc) \to take\_course(fstu, fc)$

   表示选修了当前学生的可选课程的相似毕业生。

3) $pass(cstu, fc) \rightarrow pass(x, fc)$

表示选修了当前学生的可选课程并取得合格的相似毕业生。

各高校通常要求毕业生必须完成必修课程，并从每个课程知识群中选修满足该知识群最低学分要求的选修课程，才可顺利毕业。针对这一情况，必修课程以第一组公理表示：

4) $choose\_major(cstu, m) \rightarrow take\_course(cstu, fc_{r_i})$：

表示专业 m 所有必修课程。其中 $i = 1,..., k_m$， k_m 为专业 m 的必修课程数目。 fc_{r_1}, fc_{r_2},..., $fc_{r_{k_m}}$ 为专业 m 的必修课程。

选修课程以第二组公理表示：

5) $choose\_major(cstu, c) \rightarrow take\_course(cstu, fc_{e_{i_l}}) \vee ... \vee take\_course(cstu, fc_{e_{j_l}})$：

表示如果当前学生 $cstu$ 选择专业 m，他必须从专业 m 中每一个知识群选择一门课程完成。其中 $l = 1,..., n_m$ 且 $(fc_{e_{i_1}},..., fc_{e_{j_1}}), (fc_{e_{i_2}},..., fc_{e_{j_2}}),..., (fc_{e_{i_{n_m}}},..., fc_{e_{j_{n_m}}})$ 是专业 m 的 n_m 个知识群。

当前学生 $cstu$ 学习能力和相似毕业生课程完成状态的信息通常会保存在学校的数据库中。系统将这些信息以以下形式作为证据：

$(E1) get\_mark(cstu, c, y)$：表示当前学生 $cstu$ 在课程 c 中取得了成绩 y。

$(E2) get\_mark(fstu, c, y) \rightarrow take\_course(fstu, fc)$：表示毕业生 $fstu$ 在课程 c 中取得了成绩 y，并选修了当前学生 $cstu$ 将来可能选修的可选课程。

$(E3) take\_course(fstu, fc) \rightarrow pass(fstu, fc)$：表示毕业生 $fstu$ 选了课程 fc，且取得了合格。构建证据集的步骤详见[3]。

2.3 课程评估

公理 1)、2)、3)选出了当前学生 $cstu$ 的相似毕业生 $fstu$。可使用 $fstu$ 选修的课程及其成绩数据预测当前学生 $cstu$ 对该课程合格率，即求出该课程的课程支持度。

定义 1 $B_f = \{g : K \cup \{f\} \mapsto g \ \& \ K \cup E \mapsto g\}$

定义 1 中， f 是假设，可替换为本章任何一个公式。

使用推理法则 (IR)[3]，将证据 $E1$、$E2$、$E3$ 分别作用于公理 1)可推理得：

$$K \cup E \begin{cases} (i) get\_mark(fstu, c, y) \\ (ii) take\_course(fstu, fc) \\ (iii) pass(fstu, fc) \end{cases}$$

接下来讨论定义 1 的假设部分。$KMCD$ 作出了两个假设。假设 $h1_{fc}$ - $take\_course(cstu, fc)$ 表示当前学生 $cstu$ 选修可选课程 fc；假设 $h2_{fc}$ - $pass(cstu, fc)$ 表示当前学生 $cstu$ 选修可选课程 fc 并取得了合格。

应用公理 2) $take\_course(cstu, fc) \to take\_course(fstu, fc)$ 及 $h1_{fc}$，可得到 $take\_course(fstu, fc)$。即：

$$K \cup \{h1_{fc}\} \mapsto take\_course(fstu, fc)$$

因此，$K \cup E$ 和 $K \cup \{h1_{fc}\}$ 可推导出 $take\_course(fstu, fc)$，得到的公式组成集合 $B_{h1_{fc}}$。 $fstu$ 为当前学生 $cstu$ 的相似毕业生。

在相似毕业生 $fstu$ 中，只有部分学生在可选课程中取得合格。为分析当前学生 $cstu$ 的合格率，系统使用假设 $h2_{fc}$，得到以下推断：

$$K \cup \{h2_{fc}\} \mapsto pass(fstu, fc)$$

其中， $fstu$ 表示全部毕业生。

$K \cup E$ 和 $K \cup \{h2_{fc}\}$ 可推导出 $pass(fstu, fc)$，得到的公式组成集合 $B_{h2_{fc}}$。 *fstu* 为选修了可选课程 *fc* 并取得合格，且在已选课程 *c* 中取得与当前学生 *cstu* 相近的成绩的毕业生。

因此，可以为 $B_{h2_{fc}}$ 与 $B_{h1_{fc}}$ 各自公式的数目的比率下定义：

定义 2 可选课程 *fc* 的支持度为：$T_{fc} = \dfrac{|B_{h2_{fc}}|}{|B_{h1_{fc}}|}$

T_{fc} 表示当前学生当前学生 *cstu* 对于可选课程 *fc* 的通过率。

若没有任何相似毕业生选修某门可选课程 *fc*，系统默认 $T_{fc} = 0$。

2.4 选择选修课

2.4.1 当前学生 *cstu* 的选课时要考虑必修课与选修课。系统将必修课设定为一种特殊情况下的选修课，该课程所属的知识群中只有一门选修课，因此该选修课等于必修。这样方便统一选修课和必修课的计算方法。

假设每个专业 *m* 指定了 n_m 组选修课程知识群：

$$(fc_{e_{i_1}},..., fc_{e_{j_1}}),(fc_{e_{i_2}},..., fc_{e_{j_2}}),...,(fc_{e_{i_{n_m}}},..., fc_{e_{j_{n_m}}})$$

学生必须从每一个知识群中选修 1 门选修课程，即一共需要选修 n_m 门选修课。系统需要从每一个知识群选出支持度最高的课程。因此，有定义：

定义 3 对于选修课程知识群 $G_l = (fc_{e_{il}},..., fc_{e_{ij}})(1 \le l \le n_m)$，推荐课程 $fc \in G_l$ 且

$$T_{fc} = \text{maximum}\,(T_{fc_{eil}},...,T_{fc_{ejl}})$$

2.5 根据课程评估结果选择专业

KMCD 的最后一步为帮助当前学生 *cstu* 选择专业。

公理 4)与公理 5)分别演算为：

$$choose\_major(cstu,m) \rightarrow take\_course(cstu, fc_i)，$$

$$choose\_major(cstu,m) \rightarrow take\_course(cstu, fc_j)，$$

其中 $i = 1,...,k$ ， $j = k+1,...,n$ ， $fc_1,..., fc_k$ 为专业 m 的必修课程， $fc_{k+1},..., fc_n$ 为KMCD 为当前学生 $cstu$ 从每个知识群中选择出来的推荐课程。

假设条件 h_m 为 $choose\_major(cstu,m)$ ，有如下定义：

定义 4

$$B1_{h_m} = B_{h1_{fc_1}} \cup ... \cup B_{h1_{fc_k}} \cup B_{h1_{fc_{k+1}}} \cup ... \cup B_{h1_{fc_n}}$$

$$B2_{h_m} = B_{h2_{fc_1}} \cup ... \cup B_{h2_{fc_k}} \cup B_{h2_{fc_{k+1}}} \cup ... \cup B_{h2_{fc_n}}$$

因此，专业推荐度的计算可定义为：

定义 5

$$T_m = \frac{\left|B2_{h_m}\right|}{\left|B1_{h_m}\right|}$$

T_m 给出了当前学生 $cstu$ 能够顺利从专业 m 毕业的可能性，称之为对当前学生 $cstu$ 选择专业 m 的支持度。

若没有相似毕业生选择专业 m 的任意一门课程，系统默认 $T_m = 0$ ，即不存在支持当前学生 $cstu$ 选择专业 m 的证据。

注意， $T_m = 1$ 并不代表当前学生 $cstu$ 必定能从专业 m 顺利毕业。它仅仅表示数据库中所有当前学生 $cstu$ 的相似毕业生 $fstu$ 的信息都支持当前学生 $cstu$ 选择专业 m 。这些信息表示的是毕业生们的历史状态而非当前学生的未来状态。

3 KMCD 功能实现

3.1 Prolog 语言简介

Prolog 是 Programming in LOGic 的缩写，是一种与人工智能和计算机语言有关的逻辑编程语言[5][6][7]。有别于一般的函数式语言，prolog 基于谓词逻辑理论，是一种描述性语言：它使用事实和规则表达关系，并使用关系解释程序逻辑。问题的求解则是使用这些关系来运行一次询问[8]。Prolog 拥有能够方便解决知识——规则与事实表示与存储问题的特点，因此选用 Prolog 语言作为 KMCD 系统的开发语言十分合适。

逻辑程序设计语言 Prolog 拥有以下特点[9][10]：

1) Prolog 程序没有特定的运行顺序。使用 Prolog 进行问题求解的重点在于逻辑地描述客观世界的对象及它们之间的关系。在进行逻辑描述的时候，Prolog 可通过一系列的规则得到需要解答的问题。问题求解的过程全部留给 Prolog 内部机制完成。；
2) 使用规则求解时，规则中的变元可以作为输入，也可以作为输出。程序员可根据需求自由选择自变量是常量还是变量；
3) 内部的回溯能力及不确定性使得 Prolog 对同一问题可以给出多个解；
4) Prolog 使用匹配与搜索进行问题求解，并具有强大的递归功能；
5) Prolog 程序和数据高度统一。Prolog 中所有数据和程序都拥有相同的形式。

由此可见，Prolog 非常适合解决以规则为基础的逻辑问题，如数据库搜索、模式匹配等。KMCD 解决的是不确定问题，且使用了一阶逻辑语言对

问题进行搜索与匹配求解，因此很适合利用 Prolog 进行系统的功能开发。

3.1.1 Prolog 目标求解的搜索

Prolog 目标求解的搜索过程如下[11]：

1. 求解目标时，从自顶至下搜索数据库会发生下列情景之一：
   a) 找到一个与目标相匹配的事实或规则的头部。对这些事实或规则在数据库中的位置做出标记，并记录下匹配产生的自由变量和具体化信息。若匹配是一个规则的头部，则对它第一个子目标进行求解。若求解成功，则满足该子目标右边的下一个子目标。
   b) 在 a)中没有找到匹配的事实或规则头部，即试图满足目标失败，使得系统向左倒退一步，即重新满足这个失败目标的左边的相邻目标。
2. 当重新满足一个目标时，必须先将由于上次目标的成功而具体化了的变量恢复成自由变量，即清除原有的变量置换。对数据库上一次成功所标记之后开始搜索。回溯的目标若成功将进入过程 1 中的 a)，若失败则进入过程 1 中的 b)。对满足询问而调用的第一个目标的回溯结束，也就是问题求解的结束。

Prolog 对目标的搜索是从左到右，对数据库的搜索是从上到下，即“深度优先”搜索。这种搜索方法能够使得 Prolog 搜索到所有满足条件的可能的解。

3.1.2 关于如何进行 Prolog 程序设计的几点讨论

使用 Prolog 语言进行程序设计的主要任务是描述问题所涉及的物体或

对象之间的关系。一个 Prolog 程序由下述三部分组成[12]：

1. 说明关于事物或对象之间的关系；
2. 定义规则；
3. 询问。

根据 Prolog 程序的特点，在设计程序时因遵循“自顶向下设计，自底向上调试”的原则。“自顶向下设计”指的是要从程序要完成的终极任务除法，为该终极任务设计出应有的规则。这些规则的定义中可能含有一些尚未定义的新规则，设计时先假定它们成立。终极任务设计完成后，再对这些尚未定义的规则进行分解，引入下一层新规则。以此类推，直到最终所有规则都有完整的定义为止。

与设计时相反，调试程序时，应从程序最底层的规则开始，确保这些规则运行无误后，再调试调用本规则的上一层规则，直至终极规则能顺利运行完毕为止。

3.1.3 Prolog 解释器及其预定义谓词

目前，存在多种可供选择的 Prolog 解释器或编译器[13]，每一种解释器或编译器都有自己对应的特点，且语法上存在一定的差异。不同的解释器或编译器提供不同的预定义谓词及接口。预定义谓词可方便地实现一些需要使用算术运算及逻辑性较强的规则；接口则实现 Prolog 与其他语言程序之间的调用。编写代码过程中，了解所使用的解释器或编译器提供的预定义谓词及接口能大大提高开发效率。

本次开发使用 SWI-Prolog 解释器，此解释器也提供了一系列预定义谓

词。例如，3.2.1 节中 $count1$ 需要使用预定义谓词 length(*?List, ?Int*)和 sum_list(*+List, -Sum*)，即计算表（list）中元素的个数及表中所有元素之和；maximum/2需要用到预定义谓词 max_member(*-Max, +List)*，即找出表中最大的元素。[15]

其他预定义谓词及接口可查阅相应的 Prolog 解释器的用户手册。[15]

3.2 用 Prolog 实现 KMCD

3.2.1 谓词的设计

按照 3.1.2 节的设计思想，可设计出全部 KMCD 系统所需规则与事实。以下是程序设计思路：

1. 问题要求最佳推荐专业，即求推荐度最大的专业 m。定义规则 $choose\_major(CSTU, M)$：

$$choose\_major(CSTU, M) :-$$
$$\text{maximum}(Tm, tm(CSTU, M, Tm)).$$

$choose\_major(CSTU, M)$ 表示当前学生 $CSTU$ 推荐选修专业 M。规则 maximum/2表示第一项为所有满足第二项的最大值。此规则的实现可使用预定义谓词，详见附录 A。

2. 规则 $\text{maximum}(Tm, tm(CSTU, M, Tm))$ 的定义中存在未定义规则 $tm(CSTU, M, Tm)$，它表示专业 M 的推荐度为 Tm。定义规则 $tm(CSTU, M, Tm)$：

$$Tm(CSTU, M, Tm) :-$$
$$bhm\_take(CSTU, M, Bhmt),$$
$$bhm\_pass(CSTU, M, Bhmp),$$
$$Tm\ is\ Bhmt / Bhmp.$$

is 是 Prolog 中的预定义谓词，可以把一个变量具体化为一个由+（加）、-（减）、*（乘）、/（除）、mod（求余）组成的结构进行算术运算的结果。

3. 规则 $tm(CSTU,M,Tm)$ 的定义中存在两条未定义规则 $bhm\_take(CSTU,M,Bhmt)$，$bhm\_pass(CSTU,M,Bhmp)$，分别表示在专业 M 中所有知识群的最佳推荐课程选修毕业生人数 $Bhmt$ 与合格毕业生人数 $Bhmp$，且这些毕业生为当前学生 $CSTU$ 的相似毕业生。定义如下：

```
bhm_take(CSTU,M,Bhmt):-
        Bhmt is
        count('major_group(M,G),choose_course(CSTU,G,FC),
              similar_fstu(CSTU,FSTU),take_course(FSTU,FC)').

bhm_pass(CSTU,M,Bhmp):-
        Bhmp is
        count('major_group(M,G),choose_course(CSTU,G,FC),
              similar_fstu(CSTU,FSTU),pass_course(FSTU,FC)').
```

其中，规则 $count/1$ 的实现可使用预定义谓词，详见附录 A。该规则的目的是求得满足第一项描述的所有可能规则的数量。上述定义中，括号内包含事实 $major\_group(M,G)$，表示专业 M 中包含知识群 G；规则 $choose\_course(CSTU,G,FC)$，表示当前学生在知识群 G 中的最佳推荐课程是 FC；规则 $similar\_fstu(CSTU,FSTU)$，表示毕业生 $FSTU$ 的学习能力与当前学生 $CSTU$ 相近，即 $FSTU$ 是 $CSTU$ 的相似毕业生；事实 $take\_course(FSTU,FC)$ 和 $pass\_course(FSTU,FC)$，分别表示学生 $FSTU$ 选修了课程 FC 和在课程 FC 中取得合格。

4. 规则 $bhm\_take(CSTU,M,Bhmt)$，$bhm\_pass(CSTU,M,Bhmp)$ 的定义中包含

两条未定义规则 $choose\_course(CSTU,G,FC)$ 与 $similar\_fstu(CSTU,FSTU)$。

首先定义 $choose\_course(CSTU,G,FC)$：

$$choose\_course(CSTU,G,FC):-$$
$$\text{maximum}(Tfc,(group\_course(G,FC),tfc(CSTU,FC,Tfc))).$$

事实 $group\_course(G,FC)$ 表示知识群 G 包含课程 FC；规则 $tfc(CSTU,FC,Tfc)$ 表示表示课程 FC 的推荐度为 Tfc。

5. 求课程推荐度的规则 $tfc(CSTU,FC,Tfc)$ 可定义为：

$$tfc(CSTU,FC,Tfc):-$$
$$Bhfc\_take(CSTU,FC,Bhfct),$$
$$Bhfc\_pass(CSTU,FC,Bhfcp),$$
$$Tfc\ \ is\ \ Bhfct/Bhfcp.$$

6. $tfc(CSTU,FC,Tfc)$ 包含两条未定义规则 $bhfc\_take(CSTU,FC,Bhfct)$ 与 $bhfc\_pass(CSTU,FC,Bhfcp)$，分别表示在课程 FC 中选修与合格毕业生人数为 $Bhfct$ 和 $Bhfcp$，且这些毕业生为当前学生 $CSTU$ 的相似毕业生。这两条规则分别定义为：

$$bhfc\_take(CSTU,FC,Bhfct):-$$
$$Bhfct\ \ is$$
$$count('similar(CSTU,FSTU),take\_course(FSTU,FC)').$$

$$bhfc\_pass(CSTU,FC,Bhfcp):-$$
$$Bhfcp\ \ is$$
$$count('similar(CSTU,FSTU),pass\_course(FSTU,FC)').$$

7. 定义规则 $similar\_fstu(CSTU,FSTU)$：

$$similar\_fstu(CSTU,FSTU):-$$
$$current\_stu(CSTU),$$
$$former\_stu(FSTU),$$
$$get\_score(CSTU,C,Y1),$$
$$get\_score(FSTU,C,Y2),$$
$$(Y2<=Y1+\delta,Y2>=Y1+\delta).$$

规则中包含 3 条事实：$current\_stu(CSTU)$，表示当前学生$CSTU$；$former\_stu(FSTU)$，表示毕业生$FSTU$；$get\_score(STU,C,Y)$，表示学生STU在课程C中取得的成绩为Y。δ是一个整数参数。

综上所述，系统中的已知的事实有：

1) $current\_stu(CSTU)$；
2) $former\_stu(FSTU)$；
3) $get\_score(STU,C,Y)$；
4) $take\_course(FSTU,FC)$；
5) $pass\_course(FSTU,FC)$；
6) $major\_group(M,G)$
7) $group\_course(G,FC)$；

系统中定义的规则有：

1) $similar\_fstu(CSTU,FSTU)$；
2) $bhfc\_take(CSTU,FC,Bhfct)$；
3) $bhfc\_pass(CSTU,FC,Bhfcp)$；
4) $tfc(CSTU,FC,Tfc)$；
5) $choose\_course(CSTU,G,FC)$；
6) $bhm\_take(CSTU,M,Bhmt)$；
7) $bhm\_pass(CSTU,M,Bhmp)$；
8) $tm(CSTU,M,Tm)$；
9) $choose\_major(CSTU,M)$；

当前用户要向系统询问：

?– *choose _ major*(*CSTU*, *M*)

其中，当前用户需要输入自己的学生学号*CSTU*，系统将返回推荐专业的编号或名称。

```
1 ?- choose_major('2011051742',M).
M = 电子信息工程.
```

图 3.2-1 一个向 KMCD 询问推荐专业的运行例子

3.2.2 数据库设计

KMCD 知识库中的已知事实包括：当前学生及毕业生的个人及相关课程学习信息、课程信息、知识群信息、专业信息、知识群与专业之间的关系、专业与知识群之间的关系；上述关系可归纳为 4 个基本表，分别存放学生、课程、专业及知识群基本信息；3 个反映表间关系的表，分别存放专业包含知识群的关系、知识群包含课程的关系、学生的选修课程学习信息。上述信息所得的实体以及其关系见图 3.2-2。

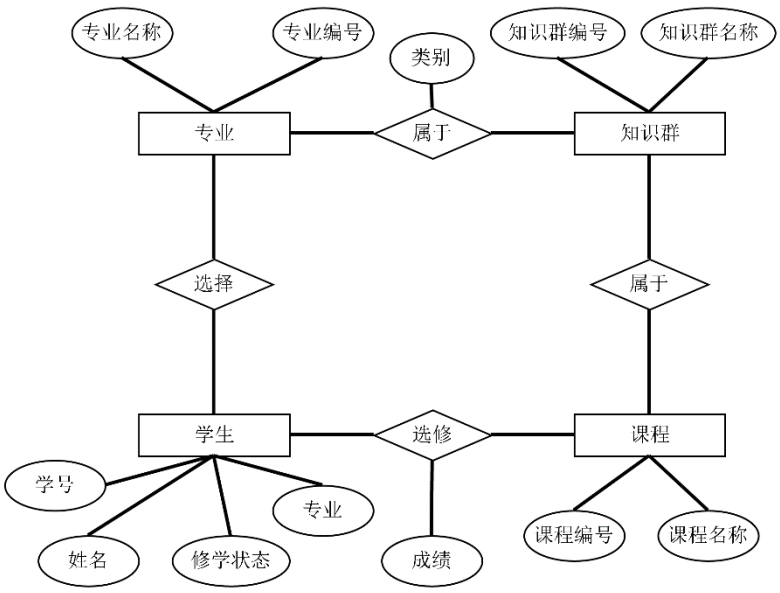

图 3.2-2 KMCD 系统 ER 图

因此，可构造 7 个表：

1) stu 表：存储学生信息。包括在读学生（即使用系统时的当前学生）与毕业生。

表格 3.2-1 stu 表定义

字段名称	字段类型	字段长度	不可为空	是否主键	说明
stu_id	char	10	√	√	学生学号。学生的唯一标识。
stu_name	varchar	100			学生姓名
stu_prospect	varchar	9			具有两个值："unknown"（在读学生）与"graduated"（毕业生）
stu_major	int	11			学生专业。为日后系统扩展功能用。

2) course 表：存储课程信息。

表格 3.2-2 course 表定义

字段名称	字段类型	字段长度	不可为空	是否主键	说明
course_id	char	8	√	√	课程编号。课程的唯一标识。
course_name	varchar	100			课程名称。
course_credit	float	2			学分。
course_thour	int	3			理论学时。
course_phour	int	3			实践学时。
course_semester	int	2			学期。
course_precourse	varchar	255			先修课程。

3) major 表：存储专业信息。

表格 3.2-3 major 表定义

字段名称	字段类型	字段长度	不可为空	是否主键	说明
major_id	int	11	√	√	专业编号。专业的唯一标识。
major_name	varchar	100			专业名称。

4) group 表：存储知识群信息。

表格 3.2-4 group 表定义

字段名称	字段类型	字段长度	不可为空	是否主键	说明
group_id	int	11	√	√	知识群编号。
group_name	varchar	100			知识群名称。

5) major_course 表：存储专业与知识群关系信息。

表格 3.2-5 major_course 表定义

字段名称	字段类型	字段长度	不可为空	是否主键	说明
major_id	int	11	√	√	专业编号。
group_id	int	11	√	√	知识群编号。
r_e	char	1			该知识群在该专业下为必修（取值为"R"）或选修（取值为"E"）
required_credit	float	3			该知识群在该专业内要求选修最少的学分。

6) course_group 表：存储知识群与课程关系信息。

表格 3.2-6 course_group 表定义

字段名称	字段类型	字段长度	不可为空	是否主键	说明
group_id	int	11	√	√	知识群编号。
course_id	char	8	√	√	课程编号。

7) stu_scoreofcourse 表：存储学生的课程成绩信息。

表格 3.2-7 stu_scoreofcourse 表定义

字段名称	字段类型	字段长度	不可为空	是否主键	说明
stu_id	char	10	√	√	学生学号。
course_id	char	8	√	√	课程编号。
course_score	double	0			课程分数。

另外，为了实现 4.2 自主选课功能，需要再添加一个基础表：

8) course_selfselect 表：存储学生自主选课信息（见 4.2 节）。

表格 3.2-8 course_selfselect 表定义

字段名称	字段类型	字段长度	不可为空	是否主键	说明
stu_id	char	10	√	√	学生学号。
group_id	int	11	√	√	该专业下的知识群编号。
course_id	char	8			学生在该专业的该知识群下选择的课程的编号。
major_id	int	11	√	√	专业编号
r_e	char	1			该知识群在该专业下为必修（取值为"R"）或选修（取值为"E"）

3.2.3 Prolog 与数据库的连接

Prolog 在构造知识库中的事实时可通过 ODBC 从 DBMS 中获取数据。SWI-Prolog 提供 ODBC 接口[14]，相关的的预定义谓词有：odbc_connect(+*DSN*, -*Connection*, +*Options*)、odbc_disconnect(+*Connection*)

和 odbc_query(+*Connection*, +*SQL*, -*RowOrAffected*)。

预定义谓词 odbc_connect(+*DSN*, -*Connection*, +*Options*)功能为创建一个新的 ODBC 连接。其中，*Connection* 为指向数据源 *DSN* 的一个句柄，即这个连接的别名。后续操作中可通过这个别名对 *DSN* 指向的数据库进行操作。KMCD 创建 ODBC 连接代码如下：

```
open_odbc:-
    odbc_connect('cpl_kmcd',_,
        [ user('root'),
        password(''),
        alias('kmcd_connection'),
        open('once')]).
:- open_odbc.
```

预定义谓词 odbc_disconnect(+*Connection*)功能为关闭 *Connection* 指向的连接。它将销毁这个连接的别名。KMCD 释放 ODBC 连接代码如下：

```
close_odbc:-
    odbc_disconnect(kmcd_connection).
```

预定义谓词 odbc_query(+*Connection*, +*SQL*, -*RowOrAffected*)的功能为向 *Connection* 指向的数据库发送 *SQL* 语句。若此语句为 SELECT 语句，就将执行结果返回给 *RowOrAffected*。KMCD 通过 ODBC 从数据库中取出数据的一个例子如下：

```
current_stu_get(CSTU_ID,CSTU_NAME):-
    odbc_query(kmcd_connection,'
        select distinct stu_id,stu_name
        from cpl_kmcd_stu stu
        where stu.stu_prospect="unknown"',
        row(CSTU_ID,CSTU_NAME)).
```

3.3 功能优化

3.3.1 推荐度计算方式优化

在知识群内对比课程支持度T_{fc}时，有可能出现一种情况：若干门课程的支持度T_{fc}相等。此时，KMCD 系统认为这些课程的推荐程度是一致的。但是，支持度T_{fc}相等的课程之间有可能存在差别，即选修人数$\left|B_{h1_{fc}}\right|$与及合格人数$\left|B_{h2_{fc}}\right|$的差别。根据定义 2，课程支持度T_{fc}为合格人数$\left|B_{h2_{fc}}\right|$与选修人数$\left|B_{h1_{fc}}\right|$的比。T_{fc}相同只表明合格人数$\left|B_{h2_{fc}}\right|$与选修人数$\left|B_{h1_{fc}}\right|$的比相同，而这些课程的合格人数$\left|B_{h2_{fc}}\right|$及选修人数$\left|B_{h1_{fc}}\right|$，与与其支持度T_{fc}相等的课程的人数可能相差 1 倍以上。此时，仅仅因为课程支持度T_{fc}相等就认为其推荐程度相同显然是不合适的。例如，假设两门课程 A 与 B 的支持度$T_A = T_B = 1$，而$\left|B_{h1_A}\right|$=$\left|B_{h2_A}\right|$=1，$\left|B_{h1_B}\right|$=$\left|B_{h2_B}\right|$=10。显然，B 包含的信息量（选修人数与合格人数）更大。即使$T_B = T_A$， B 仍应该得到更高的推荐度。

因此，可修正定义 2 为定义 6：

定义 6 可选课程 *fc* 修正后的支持度为：$T\_Correction_{fc} = T_{fc} + Correction$

其中，*Correction* 为修正值。该修正值应能反映支持度T_{fc}相等的课程之间的差异，且这个差异与选修人数或合格人数有关。本文选择选修人数作为修正值的判定基础。有如下定义：

定义 7 课程支持度的修正值为：

$$Correction = \frac{\left|B_{h1_{fc}}\right|}{N} \quad (N > 1)$$

其中，*N* 为一个整数。由上文可知，修正值*Correction* 仅在课程支持度T_{fc}相等时发挥作用。当支持度T_{fc}不相等时，修正值*Correction* 不应影响系统对

课程推荐程度的判断。即满足如下条件：

$$if \quad T_{fc} > T_{fc'} \quad then \quad T\_Correction_{fc} > T\_Correction_{fc'}$$

另外，各门课程的选修人数都少于等于相似毕业生人数，各门课程的合格人数少于等于选修人数，即有条件：

$$0 \le \left|B_{h2_{fc}}\right| \le \left|B_{h1_{fc}}\right| \le F$$

其中，F 为相似毕业生总人数。

因此，只要找出满足上述条件的 N，即可得到修正值 $Correction$ 及修正后的支持度 $T\_Correction_{fc}$。

设有课程 fc 和 fc'，选修人数分别为 t_1 和 t_2，合格人数分别为 p_1 和 p_2，则给出以下定理：

定理 1. 存在 N，使得 $\frac{p_1}{t_1}+\frac{t_1}{N}>\frac{p_2}{t_2}+\frac{t_2}{N}$ $(N>1)$

已知 t_1，p_1，t_2，p_2, N, F 皆为整数，N 为决定修正值的参数，F 为相似毕业生总人数。其中 $\frac{p_1}{t_1} > \frac{p_2}{t_2}, 0 \le p_1 \le t_1 \le F, 0 \le p_2 \le t_2 \le F$。

证明：

$\frac{p_1}{t_1} + \frac{t_1}{N} > \frac{p_2}{t_2} + \frac{t_2}{N} \qquad (N > 1)$

变换得 $\quad N > \frac{t_2 - t_1}{\frac{p_1}{t_1} - \frac{p_2}{t_2}}$ $\quad$ 即 $\quad N > \frac{(t_2 - t_1)t_1t_2}{p_1t_2 - p_2t_1}$ $\qquad$ （1）

$\because \frac{p_1}{t_1} > \frac{p_2}{t_2}$ $\quad$ 且p_1，p_2，t_1，t_2是整数，$\quad \therefore p_1t_2 - p_2t_1 \ge 1$ $\qquad$ （2）

综合（1）（2）有 $\frac{(t_2 - t_1)t_1t_2}{p_1t_2 - p_2t_1} \le (t_2 - t_1)t_1t_2$ $\qquad$ （3）

根据（3），如果取$N > (t_2 - t_1)t_1t_2,$，即可满足$N > \frac{(t_2 - t_1)t_1t_2}{p_1t_2 - p_2t_1}$ （4）

观察（4）右部，当t_2取最大值F时，$(t_2-t_1)t_1t_2$取得最大值。

因此将$t_2 = F$代入$(t_2 - t_1)t_1t_2$得

$$g(t_1) = (F - t_1)Ft_1 = F^2t_1 - Ft_1^2 \quad (5)$$

经计算，该式在$t_1 = \frac{F}{2}$时有最大值$(F^2 - \frac{F}{2}) \times F \times \frac{F}{2} = \frac{F^3}{4}$

$\therefore$ 只要$N > \frac{F^3}{4}$, 则$\frac{p_1}{t_1} + \frac{t_1}{N} > \frac{p_2}{t_2} + \frac{t_2}{N}$成立。

证毕。

定理 1 的含义是，根据定义 6 和定义 7，对于课程 *fc* 和 *fc'*，当决定修正值的参数$N > \frac{F^3}{4}$时，若支持度修正前有关系$T_{fc} > T_{fc'}$，则支持度修正后该关系不变，即$T_Correction_{fc} > T_Correction_{fc'}$。

注意到，当$F=1$时，$N<1$，不符合$N>1$的条件。但可通过调整成绩浮动值δ排除$F=1$的情况（见 5.1.1 节）。

$N > \frac{F^3}{4}$为理论上能保证定理 1 成立的值。实际系统中 N 的值可根据实际情况略低于$\frac{F^3}{4}$，只要能保证加上 *Correction* 后系统对所有T_{fc}不等的课程的推荐程度判断不被影响即可。

3.3.2 系统运行效率优化

Prolog 程序的效率主要是指其执行时间与所使用的内存空间的节省程度。Prolog 是一种慢语言，如何提高程序的执行效率显得尤为重要。为了提高 Prolog 程序的效率，需要深入了解 Prolog 程序的过程性语义，并结合解释器的特性对程序进行优化。这在一定意义上违背了逻辑程序设计的最初指导思想，牺牲了 Prolog 的说明性语义。

本系统编程时采用了以下措施提高程序的效率：

1) 将 DBMS 中所需数据存放到解释器内置的数据库机制中

在关系数据库与 Prolog 解释器之间的接口中，效率是一个主要的问题[16]。SWI-Prolog 解释器提供的 ODBC 接口预定义谓词 odbc_query(+*Connection*, +*SQL*, -*RowOrAffected*, +*Options*)允许程序向 DBMS 输出 SQL 语句，从外部数据库中读取需要的数据。然而，使用这种方式，每读取一次数据库数据，解释器都要完成一次“匹配当前已有数据库 *Connection* 句柄 - 向 DBMS 传输 SQL 语句 - 将所得结果依顺序匹配入 *RowOrAffected* 表 - 检查是否存在并执行 *Options* 指定的效果 – 返回最终结果”5 个步骤的操作。这大大降低了程序搜索与匹配效率，在数据量较大时尤为明显。

SWI-Prolog 提供的一个解决办法是，使用预定义谓词 dynamic :*PredicateIndicator*, ...声明所有需要使用 ODBC 接口的谓词，并在程序预分析阶段（即启动程序后、开始询问前）用预定义谓词 assert(+*Term*)将这些要多次读取的数据作为事实或子句存放到解释器内置的数据库中。这样，询问时便不必再通过 ODBC 接口从 DBMS 读取数据，大大提升搜索效率。见附录 A 谓词 initialize/0 的定义。

2) 避免不必要的重复求解

设计 Prolog 程序时，有可能出现反复执行多次完全相同的子目标的情况。这会造成明显的资源浪费。例如，询问规则 $bhfc\_take(CSTU, FC, Bhfct)$ 和 $bhfc\_pass(CSTU, FC, Bhfcp)$ 时，二者都存在子目标 $similar\_fstu(CSTU, FSTU)$，且 $bhfc\_take(CSTU, FC, Bhfct)$ 的子目

标 $take\_course(FSTU,FC)$ 和 $bhfc\_pass(CSTU,FC,Bhfcp)$ 的子目标 $pass\_course(FSTU,FC)$ 实际上都由 stu_scoreofcourse 表的数据组成（高于合格分数线的成绩数据组成 $pass\_course(FSTU,FC)$，全部成绩数据组成 $take\_course(FSTU,FC)$）。若按表面语义定义这两条规则，将会在计算课程支持度时按顺序分别执行 $bhfc\_take(CSTU,FC,Bhfct)$ 与 $bhfc\_pass(CSTU,FC,Bhfcp)$，从而造成对子目标 $s\ i\ \underline{m}f\ i(C\ l\ t\ Sa,uFTr\ S\ l)$ 与数据表 stu_scoreofcourse 的不必要的重复读取。本例的解决方法是，将事实 $take\_course(FSTU,FC)$ 与事实 $pass\_course(FSTU,FC)$ 合并，用一个谓词代表 stu_scoreofcourse 表中所有记录，以项 P_F 表示课程是否合格（1 为合格，0 为不合格）。而规则 $bhfc\_take(CSTU,FC,Bhfct)$ 与规则 $bhfc\_pass(CSTU,FC,Bhfcp)$ 合并，使用一条规则统一表示某门课程选修与合格的人数。代码如下：

```
bhfc_take_pass(CSTU_ID,OFC,BhfcT,BhfcP):-
    current_stu(CSTU_ID,_),
    optional_future_course(CSTU_ID,OFC),
    findall(P_F,similar_fstu_take_course(CSTU_ID,_,OFC,P_F),P_F_L)
    ,length(P_F_L,BhfcT)
    ,sum_list(P_F_L,BhfcP).
```

上述代码的含义为，以所有与课程 OFC 选修记录有关的事实 similar_fstu_take_course/4 的数目表示该课程相似毕业生选修人数，这些记录的 P_F 值之和表示相似毕业生合格人数。这样，只需读取一次 stu_scoreofcourse 表即可得出结果。此举大大提升了程序的执行效率，代价是降低了程序的可读性。

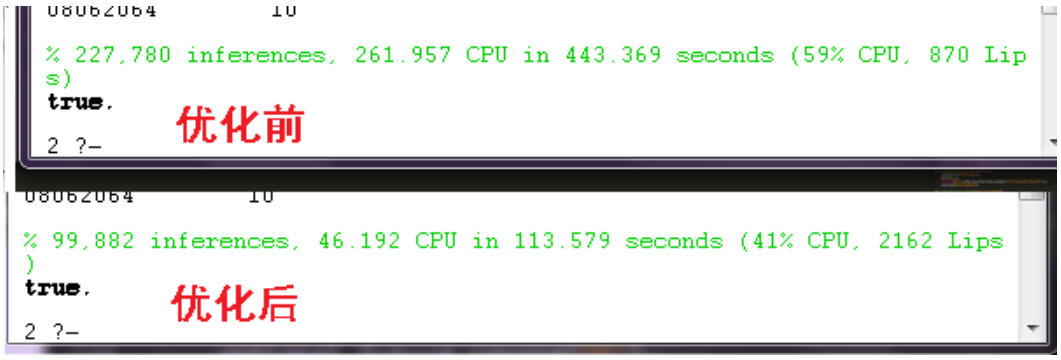

图 3.3-1 代码优化前和优化后计算的执行速度

3) 改变子目标的执行顺序

Prolog 问题的求解是顺序相关的，可以通过调整规则定义中的子目标执行顺序提高程序执行的效率。例如，原有代码：

```
choose_major(CSTU_ID,MAJOR_NAME):-
    current_stu(CSTU_ID,_),
    tm_max(CSTU_ID,TmM),
    tm(CSTU_ID,MAJOR_ID,_,_,Tm_FLOAT),
    Tm_FLOAT is TmM,
    major(MAJOR_ID,MAJOR_NAME).
```

上述代码的含义为，只要所有 tm/5 子目标中项 Tm 等于所有 Tm 中的最大值 TmM，则该规则的专业 M 为推荐专业。按上述的定义，程序会搜索当前学生 CSTU_ID 每一个专业 MAJOR_ID 的 tm/5，得到该专业的支持度 Tm_FLOAT，再判断 Tm_FLOAT 是否等于 TmM。这样，与当前学生 CSTU_ID 有关的每一条规则 tm/5 都会被扫描一次。然而，若把子目标 tm/5 与子目标“Tm_FLOAT is TmM”的执行顺序交换，程序会先设定项 Tm_FLOAT 的值等于 TmM，再在执行 tm/5 时只搜索所有 Tm_FLOAT 值为 TmM 的规则 tm/5。这将大大提升搜索效率。

依据上述的代码优化方法，本文成功将代码量从 483 行优化为 241 行，

询问一次推荐专业的运行时间从半小时以上优化为 1 秒以内，优化效非常显著。

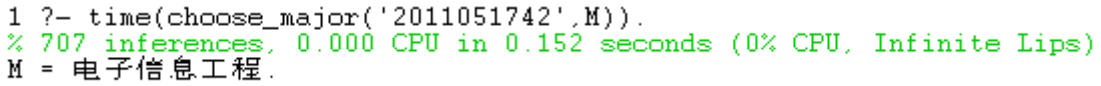

图 3.3-2 优化后询问一次推荐专业的运行效率

4 KMCD 线上系统开发

4.1 开发和运行环境

表格 4.1-1 KMCD 系统开发和运行环境

开发环境		运行环境		
名称	描述	名称	描述	
开发工具	SWI-Prolog	服务器端环境	服务器端运行环境	OS Windows Service2003
	ThinkPHP框架		应用服务器	Apache httpd 2.2.29
	Sublime Text 2		数据库	MySql 5.6.21
	Navicat Premium		PHP	PHP 5.6.3
开发语言	Prolog、PHP、html、SQL等	客户端环境	客户端软件环境	IE6以上、Chrome、FireFox、Opera等主流浏览器

4.2 线上系统功能改动及优化

除第三章提到的基础功能外，在开发线上系统时，为了使系统更加符合现实、具有更好的实用性，本文对系统功能做出如下调整：

1) 选择“知识群最佳课程”时，不仅仅选出推荐度最高的一门，而是根据推荐度排名从高至低选出 N 个课程，直至所选课程的总学分满足该知识群的最低学分学分要求为止；
2) 为系统增添“用户自主选课”功能。用户可根据喜好在各知识群内选择自己希望选修的课程，系统根据这些课程的相似毕业生选修人数与合格人数计算出自主选课的专业推荐度及推荐专业；
3) 考虑到 1)和 2)，将课程/专业推荐的排名规则更改为：
   a) 比较推荐度，推荐度越大推荐排名越高；
   b) （对于课程）若推荐度相同，则比较课程学分，课程学分越大推荐排名越高；
   c) 若推荐度（及课程学分）都相同，比较选修人数，选修人数越多

推荐排名越高。

4.3 功能框架

如图 4.3-1 所示，专业选取决策工具由已修课程、推荐课程和推荐专业三个模块组成。

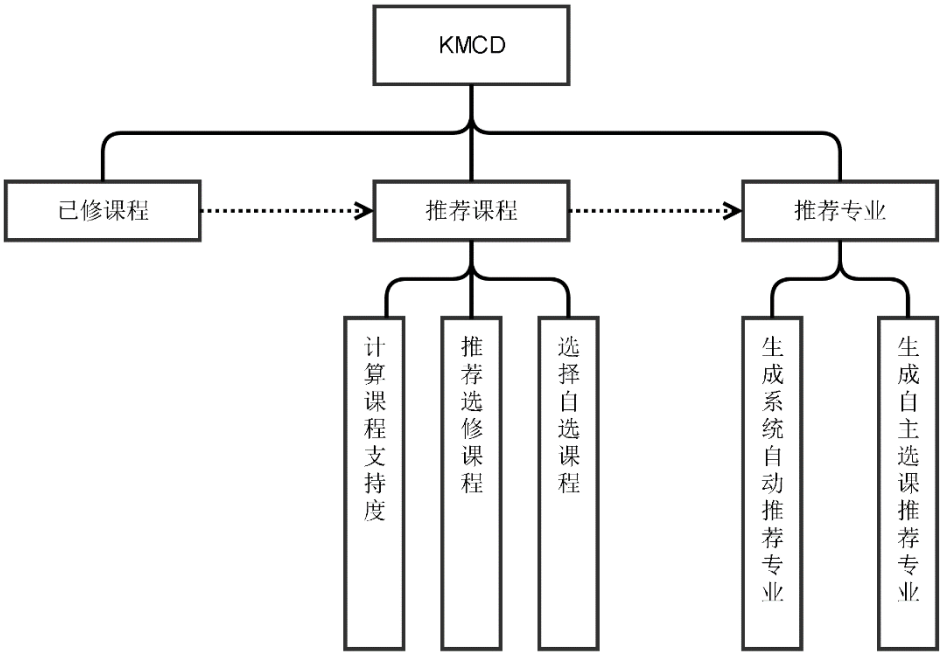

图 4.3-1 KMCD 功能框架

1) 已修课程模块从数据库中取出并显示当前学生已修课程数据，将当前学生及其已修课程数据发送给推荐课程模块；
2) 推荐课程模块根据数据库数据计算并显示各门课程支持度及各门专业推荐课程，并提供选课功能供当前学生选择自己喜欢的课程。再将系统推荐课程及用户自选课程发送给推荐专业模块；
3) 推荐专业模块分别计算并显示系统自动推荐专业与根据自主选课计算的推荐专业。

5 系统验证及数据分析

为验证系统的正确性与可用性，本节以暨南大学信息科学技术学院计算机系电子信息工程专业及计算机科学与技术专业 11 届毕业生数据进行实验。数据库中共有 2 门专业，104 门课程，91 位毕业生，2151 条课程成绩记录。当前用户已修 2 门课程。

5.1 确定系统参数

系统可以两个指标作为性能衡量的参考标准：相似毕业生人数占总毕业生人数的比例，表示用于计算推荐度的样本大小；以及推荐课程/专业所得推荐度间的标准差，表示系统对课程/专业的推荐倾向性。

5.1.1 确定成绩浮动值δ

成绩浮动值δ影响系统所得当前用户相似毕业生人数。本文希望得到相似毕业生人数占总毕业生人数的 25%左右。若δ设置得过大，KMCD 系统选取相似毕业生过多，得到的相近毕业生之间的学习能力差异较大，无法准确反映当前学生学习能力；若δ设置得过小，所得相似毕业生过少，则提供计算课程支持度的人数过少。两种情况都导致最终结果不够准确。由于本次实验所有成绩样本均为整数，δ可只考虑整数。取δ的区间为[0,10]，δ在此区间中变动时所得相似毕业生占总毕业生人数比例如表格 5.1-1 与图 5.1-1。

表格 5.1-1 成绩浮动值δ与相似毕业生人数的关系

成绩浮动值 δ	相似毕业生人数	占总毕业生人数百分比
0	2	2.20%
1	14	15.38%
2	21	23.08%
3	31	34.07%
4	42	46.15%
5	48	52.75%
6	51	56.04%
7	57	62.64%
8	61	67.03%
9	69	75.82%
10	72	79.12%

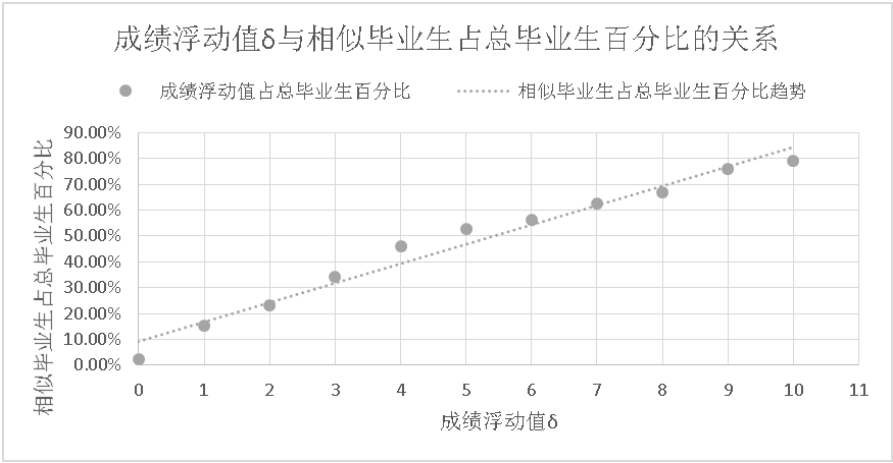

图 5.1-1 成绩浮动值δ与相似毕业生占总毕业生人数百分比的关系

由结果可知，相似毕业生占总毕业生人数百分比随δ的增大而增大，且趋势呈正相关线性关系。δ=2 时，相似毕业生人数占总毕业生人数百分比最接近 25%。因此，选取 2 为系统成绩浮动值。

5.1.2 确定合格分数线γ

合格分数线影响推荐度之间标准差，即系统对推荐课程/专业的倾向性。推荐度标准差越大，所得结果倾向性越强。本文希望推荐度之间的标准差尽量大。设合格分数线为γ，所得结果见附录 B 与图 5.1-2、图 5.1-3。

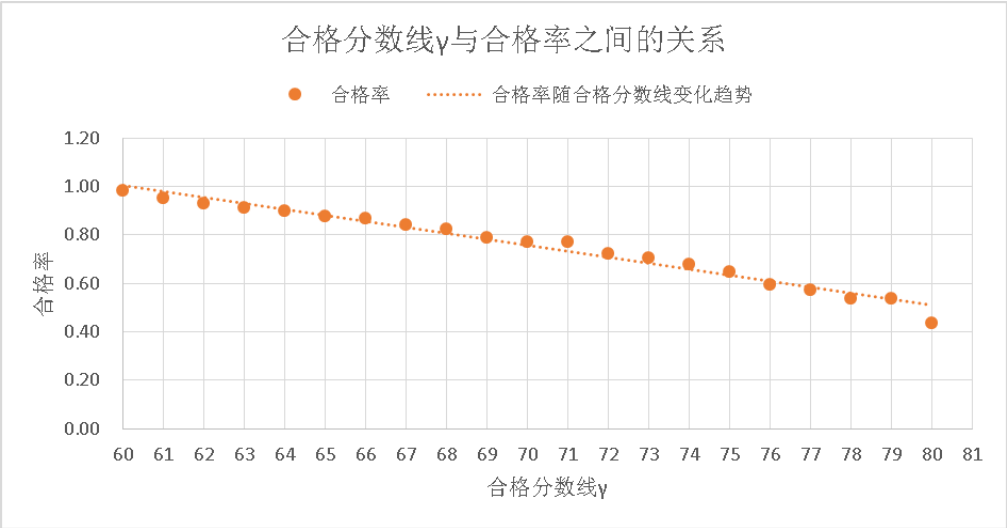

图 5.1-2 合格分数线 γ 与合格率间的关系

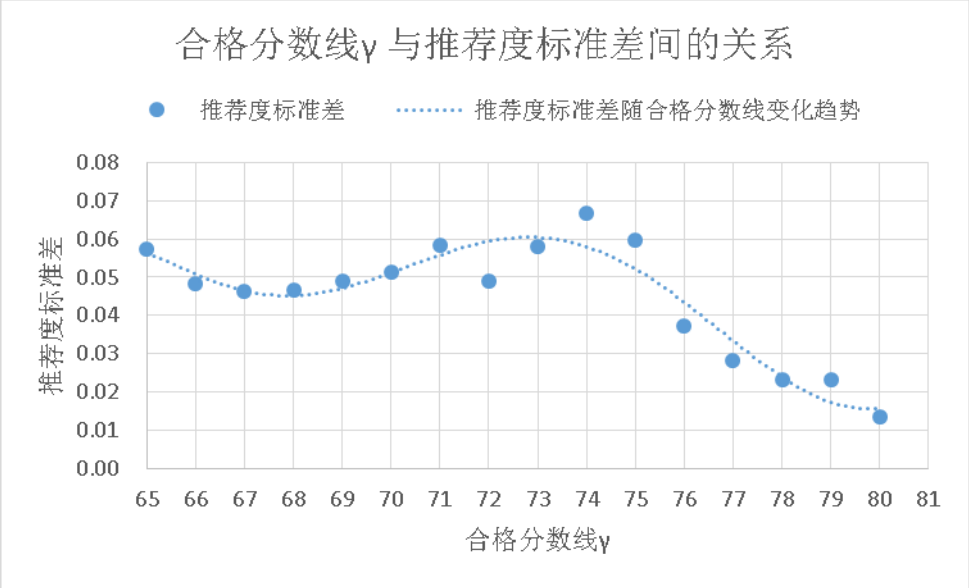

图 5.1-3 合格分数线 γ 与推荐度标准差间的关系

由结果可知。γ 在大于 80 的区间内合格率低于 0.5，不考虑 γ 在该区间中取值。而 γ 在[60,64]区间内时合格率高于 0.9，系统将失去针对性，故也不选择此区间作为 γ 取值。合格率在[0.5,0.9]区间内、推荐度标准差偏大、推荐度标准差变化趋势与 γ 呈较明显多项式关系的 γ 取值落在[65,80]区间。在此区间中，推荐度标准差首先呈下降趋势，在 γ 取值为 67 后随 γ 增大而增大，直到 γ 取值为 74 左右时达到此区间中的最大值，之后回落。因此，

可认为γ取值为74时系统推荐性能最佳。

5.2 实际分析结果

图 5.2-1 Prolog 程序询问推荐专业运行结果

```
1 ?- choose_major('2011051742',M).
M = 电子信息工程.
```

图 5.2-2 系统自动推荐专业结果

表格 5.2-1 系统自动推荐选修课

知识群名称	群内要求学分	课程编号	课程名称	课程学分
基础选修	2	8062063	数字系统设计	2
信号分析与信息处理	4	8060232	计算方法与程序设计	2
		8061135	单片机原理及应用	2
专业设计实践	6	8061138	虚拟仪器技术	2
		8062064	DSP应用系统综合设计	1
		7020106	传感技术	2
微电子技术与集成电路	4	8062059	超大规模集成电路技术基础	2
		8061026	集成电路器件电子学	5
通信网络与嵌入式技术	4	8060072	计算机网络	3
		8062027	卫星通信系统	2

图 5.2-3 自主选课推荐专业结果

表格 5.2-2 系统自动推荐选修课

知识群名称	群内要求学分	课程编号	课程名称	课程学分
基础选修	2	8062063	数字系统设计	2
信号分析与信息处理	4	8060232	计算方法与程序设计	2
		8061135	单片机原理及应用	2
专业设计实践	6	8061138	虚拟仪器技术	2
		8062064	DSP应用系统综合设计	1
		7020106	传感技术	2
微电子技术与集成电路	4	8061026	集成电路器件电子学	5
通信网络与嵌入式技术	4	8060072	计算机网络	3
		8062027	卫星通信系统	2

1) 对比图 5.2-1 与图 5.2-2，可知线上系统所得最佳推荐专业（即推荐排名为 1 的专业）与 Prolog 程序运行所得结果一致，证明了系统的正确性；

2) 图 5.2-2 与图 5.2-3 反映出学生选择选修课时，会在一定程度上参考系统的推荐排名，因此所得推荐专业结果与系统自动推荐结果较为接近。

3) 对比表格 5.2-1 与表格 5.2-2，可发现在“微电子技术与集成电路”这一知识群中，群内要求学分为 4，系统根据课程排名依次选择了排名度最高的“超大规模集成电路技术基础”（2 学分），因还未达

到学分要求，继续选择推荐排名第二的“集成电路器件电子学”（5学分），即一共选修了两门课程。而由课程学分可知，其实只需选择排名第二的“集成电路器件电子学”课程便可达到学分要求，即当前学生最终做出的选择。由此可见，KMCD 系统仅仅给出作为选择课程的理论建议，不一定是实际的最优解，具体情况还需用户结合实际分析。

结论

本次毕业设计由我在余芳老师的指导下独立完成。开发过程的第一步是理解老师所提供的推理模型。为此，我回顾了人工智能和专家系统课程中所学到的知识。接着，为开发出能实现此模型的系统，我自学了 Prolog 语言，了解了 Prolog 语言的特性，成功地使用它开发出了 KMCD 系统，并在此过程中领悟了 Prolog 语言与常用的过程性语言的区别。秉承着精益求精的信念，我在系统功能实现完毕后对代码进行多次优化及重构，力图得到最佳的性能与效果。

在线下系统开发完毕后，我又自学了 PHP 与 html 5 网页开发，成功地将 KMCD 系统移植到网站上，让其成为一个真正的、具有实用性的线上系统。在线上系统的基本功能实现完毕后，我又对系统进行多次功能与界面上的改动，力求让系统满足用户友好、更符合现实情况等的要求。

KMCD 可作为一个网络服务提供给不能与专业人员面对面交流的学生，帮助他选择适合他学习及他感兴趣的专业。学生即使不采纳 KMCD 的推荐，也可以根据其提供的信息作出有利于自己的选择。当然，当学生需要选择专业的时候，还有很多本文未列出的因素需要考虑，例如该专业的就业前景、学生的个人兴趣等。本文所讨论的 KMCD 系统仅仅根据学生本科一年级的成绩考虑学生从专业中毕业的概率。但其他考虑因素只要也可以通过一阶逻辑系统表现，日后使用这些因素拓展系统功能是可能的。

在整个开发过程中，我学习了很多不同的开发工具，加强了通过项目自学技术的能力。相信这些经历会在日后的学习和工作中发挥出很大的作用。

致谢

首先要感谢本次毕业设计的导师，余芳老师。余老师为人严谨，待学生尤其严格，私底下却又非常和蔼。早在其他同学还未开始设计之前，余老师就已敦促我要抓紧时间完成设计，以免在答辩前手忙脚乱。在系统开发过程中，余老师对其中的问题和不足之处及时给出了修改和指正。至此在论文完成之际，对余老师表示衷心的感谢和诚挚的敬意！

另外，感谢中山大学数学系 2011 级苏馨同学为第 3.3.1 节数学证明提供帮助。

在这次设计中，由于水平有限，加之仓促，难免有不足之处，恳请各位老师批改指正。

附录A Prolog 源码

```
open_odbc:-
%% Create a new ODBC connection to data-source DSN and return a handle to
this connection in Connection.
    odbc_connect('cpl_kmcd',_,
        [ user('root'),
        password(''),
        alias('kmcd_connection'),
        open('once')]
    ).

close_odbc:-
    odbc_disconnect(kmcd_connection).

:-open_odbc.

%% ------------------------------------------------
%% KEY FACTS to KMCD read from SQL
%% ------------------------------------------------
:- dynamic current_stu/2.
current_stu_get(CSTU_ID,CSTU_NAME):-
    odbc_query(kmcd_connection,'
        select distinct stu_id,stu_name
        from cpl_kmcd_stu stu
        where stu.stu_prospect="unknown"
        ',
        row(CSTU_ID,CSTU_NAME))
    .

:- dynamic former_stu/2.
former_stu_get(FSTU_ID,FSTU_NAME):-
    %% odbc_current_connection(Connection,_),
    odbc_query(kmcd_connection,'
        select distinct stu_id,stu_name
        from cpl_kmcd_stu stu
        where stu.stu_prospect="graduated"',
        row(FSTU_ID,FSTU_NAME)
        ).

:- dynamic major/2.
major_get(MAJOR_ID,MAJOR_NAME) :-
```

```
    %% odbc_current_connection(Connection,_),
    odbc_query(kmcd_connection,'
        select distinct major.major_id, major.major_name
        from cpl_kmcd_major major',
        row(MAJOR_ID,MAJOR_NAME)
        ).

:- dynamic cstu_course_id/2.
cstu_course_id_get(CSTU_ID,CSTU_COURSE_ID) :-
    %% odbc_current_connection(Connection,_),
    odbc_query(kmcd_connection,'
        select stu_id,course_id
        from cpl_kmcd_completed_course cc',
        row(CSTU_ID,CSTU_COURSE_ID)
        ).

:- dynamic optional_future_course/2.
optional_future_course_get(CSTU_ID,OPTIONAL_COURSE_ID) :-
    %% odbc_current_connection(Connection,_),
    odbc_query(kmcd_connection,'
        select distinct stu_id,course_id
        from cpl_kmcd_future_courses',
        row(CSTU_ID,OPTIONAL_COURSE_ID)
        ).

:-dynamic get_score/2.
get_score_get(STU_ID,COURSE_ID,COURSE_SCORE) :-
    %% odbc_current_connection(Connection,_),
    odbc_query(kmcd_connection,'
        select stu_id,course_id,course_score
        from cpl_kmcd_stu_scoreofcourse',
        row(STU_ID,COURSE_ID,COURSE_SCORE)
        ).


%% courses' score of current students
cstu_get_score(CSTU_ID,CSTU_COURSE_ID,CSTU_COURSE_SCORE) :-
    current_stu(CSTU_ID,_)
    ,get_score(CSTU_ID,CSTU_COURSE_ID,CSTU_COURSE_SCORE).

%% future courses which has been token by previous students.
%%
```

```
fstu_take_course(FSTU_ID,OPTIONAL_COURSE_ID,COURSE_STATUS)
%% COURSE_STATUS=1 means pass, COURSE_STATUS=0 means fail.
fstu_take_course(FSTU_ID,OPTIONAL_COURSE_ID,P_F):-
    get_score(FSTU_ID,OPTIONAL_COURSE_ID,COURSE_SCORE ),
    ((COURSE_SCORE >= 74, COURSE_SCORE =< 100) -> P_F = 1; P_F = 0).



%% elective courses groups
:-dynamic courses_group_id/3.
courses_group_id_get(CSTU_ID,GROUP_ID,COURSE_ID,R_E) :-
    odbc_query(kmcd_connection,'
    select stu_id,group_id,course_id,r_e
    from cpl_kmcd_future_major_courses',
    row(CSTU_ID,GROUP_ID,COURSE_ID,R_E)
    ).

:-dynamic group_id/2.
group_id_get(CSTU_ID,GROUP_ID):-
    odbc_query(kmcd_connection,'
    select distinct stu_id,group_id
    from cpl_kmcd_future_major_courses',
    row(CSTU_ID,GROUP_ID)
    ).

:-dynamic major_group_id/4.
major_group_id_get(CSTU_ID,MAJOR_ID,GROUP_ID,R_E) :-
    %% odbc_current_connection(Connection,_),
    odbc_query(kmcd_connection,'
    select stu_id,major_id,group_id,r_e
    from cpl_kmcd_future_major_courses
    group by stu_id,major_id,group_id',
    row(CSTU_ID,MAJOR_ID,GROUP_ID,R_E)
    ).

initialize:-
    forall(current_stu_get(CSTU_ID,CSTU_NAME),assert(current_stu(CSTU_I
D,CSTU_NAME)))
    ,forall(former_stu_get(FSTU_ID,FSTU_NAME),assert(former_stu(FSTU_I
D,FSTU_NAME)))
    ,forall(cstu_course_id_get(CSTU_ID,CSTU_COURSE_ID),assert(cstu_cour
se_id(CSTU_ID,CSTU_COURSE_ID)))
```

```
    ,forall(optional_future_course_get(CSTU_ID,OPTIONAL_COURSE_ID),as
sert(optional_future_course(CSTU_ID,OPTIONAL_COURSE_ID)))
    ,forall(get_score_get(STU_ID,COURSE_ID,COURSE_SCORE),assert(get_
score(STU_ID,COURSE_ID,COURSE_SCORE)))
    ,forall(courses_group_id_get(CSTU_ID,GROUP_ID,COURSE_ID,R_E),ass
ert(courses_group_id(CSTU_ID,GROUP_ID,COURSE_ID,R_E)))
    ,forall(group_id_get(CSTU_ID,GROUP_ID),assert(group_id(CSTU_ID,GR
OUP_ID)))
    ,forall(major_get(MAJOR_ID,MAJOR_NAME),assert(major(MAJOR_ID,
MAJOR_NAME)))
    ,forall(major_group_id_get(CSTU_ID,MAJOR_ID,GROUP_ID,R_E),assert
(major_group_id(CSTU_ID,MAJOR_ID,GROUP_ID,R_E)))
    .
:-initialize.

%% ------------------------------------------------
%% KEY RULES to KBMC
%% ------------------------------------------------

%% -----------
%% Tfc
%% -----------
/*Find out former students FStu who have similar capabilities
  with the current student CSTU_ID.*/
similar_fstu(CSTU_ID,FSTU_ID):-
    cstu_get_score(CSTU_ID,CSTU_COURSE_ID,CSTU_COURSE_SCORE)
    ,get_score(FSTU_ID,CSTU_COURSE_ID,FSTU_COURSE_SCORE)
    ,FSTU_COURSE_SCORE=<CSTU_COURSE_SCORE+2,FSTU_COURS
E_SCORE>=CSTU_COURSE_SCORE-2
    .



/*Find out future courses TC that those former students FSTU_ID
have been token.*/
similar_fstu_take_course(CSTU_ID,FSTU_ID,FSTU_COURSE_ID,P_F):-
    setof(FSTU_ID,similar_fstu(CSTU_ID,FSTU_ID),FSTU_ID_LIST),
    member(FSTU_ID,FSTU_ID_LIST),
    fstu_take_course(FSTU_ID,FSTU_COURSE_ID,P_F)
    .
```

```
/*Count the number BhfcT and BhfcP of those former student who have been
taken or passed course OFC as a support for the current student CSTU_ID which
are brought by both the hypothesis and the evidence.*/
bhfc_take_pass(CSTU_ID,OFC,BhfcT,BhfcP):-
    current_stu(CSTU_ID,_),
    optional_future_course(CSTU_ID,OFC),
    findall(P_F,similar_fstu_take_course(CSTU_ID,_,OFC,P_F),P_F_L)
    ,length(P_F_L,BhfcT)
    ,sum_list(P_F_L,BhfcP)
    .


/*Calculate the supporting degree Tfc of the future courses OFC.*/
tfc(CSTU_ID,OFC,Tfc_FLOAT,BhfcT,BhfcP):-
    bhfc_take_pass(CSTU_ID,OFC,BhfcT,BhfcP),
    (BhfcT=\=0 ->
    Tfc = BhfcP/BhfcT;
    Tfc = 0),
    Tfc_C = integer(Tfc*100) + BhfcT/200,
    Tfc_FLOAT is float(Tfc_C)
    .

/*Calculate the supporting degree of courses in a gourp GROUP_ID.*/
tfc_for_group(CSTU_ID,GROUP_ID,OFC,Tfc_FLOAT,BhfcT,BhfcP):-
    current_stu(CSTU_ID,_)
    ,courses_group_id(CSTU_ID,GROUP_ID,OFC,_)
    ,tfc(CSTU_ID,OFC,Tfc_FLOAT,BhfcT,BhfcP)
    .

/*Calculate the max supporting degree of courses in a gourp GROUP_ID.*/
tfc_max(CSTU_ID,GROUP_ID,TfcM):-
    current_stu(CSTU_ID,_)
    ,group_id_get(CSTU_ID,GROUP_ID)
    ,findall(Tfc_FLOAT,tfc_for_group(CSTU_ID,GROUP_ID,_,Tfc_FLOAT,_,
_),Tfc_FLOAT_L)
    ,max_member(TfcM,Tfc_FLOAT_L)
    .

/*Find all recommended course's id whose Tfc equals to the maximum Tfc in the
group.*/
choose_course_list(CSTU_ID,GROUP_ID,TfcM,OFC_L):-
    tfc_max(CSTU_ID,GROUP_ID,TfcM)
```

```
    ,findall(OFC,tfc_for_group(CSTU_ID,GROUP_ID,OFC,TfcM,_,_),OFC_L)
    .

/*Get other information of the recommended course for further calculation.*/
choose_course(CSTU_ID,GROUP_ID,OFC,BhfcT,BhfcP,TfcM):-
    choose_course_list(CSTU_ID,GROUP_ID,TfcM,OFC_L),
    OFC_L=[OFC|_],
    tfc(CSTU_ID,OFC,_,BhfcT,BhfcP)
    .

%%Set recommended courses' information as evidence to raise the speed of
searching.
:-dynamic choose_course_fast/6.
savecc:-
    forall(choose_course(CSTU_ID,GROUP_ID,OFC,BhfcT,BhfcP,TfcM),asser
t(choose_course_fast(CSTU_ID,GROUP_ID,OFC,BhfcT,BhfcP,TfcM)))
    .
:-savecc.

%% -----------
%% Tm
%% -----------
/*list out all groups GROUP_ID and its max BhfcP/BhfcT in a major
MAJOR_ID*/
tfc_for_major(CSTU_ID,MAJOR_ID,GROUP_ID,BhfcT,BhfcP):-
    major(MAJOR_ID,_),
    major_group_id_get(CSTU_ID,MAJOR_ID,GROUP_ID,_),
    choose_course_fast(CSTU_ID,GROUP_ID,_,BhfcT,BhfcP,_)
    .


bhm_take_pass(CSTU_ID,MAJOR_ID,BhmT,BhmP):-
    current_stu(CSTU_ID,_),
    major(MAJOR_ID,_),
    findall(BhfcT,tfc_for_major(CSTU_ID,MAJOR_ID,_,BhfcT,_),BhfcT_L),
    findall(BhfcP,tfc_for_major(CSTU_ID,MAJOR_ID,_,_,BhfcP),BhfcP_L),
    sum_list(BhfcT_L,BhmT),
    sum_list(BhfcP_L,BhmP)
    .

tm(CSTU_ID,MAJOR_ID,BhmT,BhmP,Tm_FLOAT):-
    current_stu(CSTU_ID,_),
```

```
    major(MAJOR_ID,_),
    findall(BhfcT,tfc_for_major(CSTU_ID,MAJOR_ID,_,BhfcT,_),BhfcT_L),
    findall(BhfcP,tfc_for_major(CSTU_ID,MAJOR_ID,_,_,BhfcP),BhfcP_L),
    sum_list(BhfcT_L,BhmT),
    sum_list(BhfcP_L,BhmP),
    (BhmT=\=0 ->
    Tm=BhmP/BhmT;
    Tm=0),
    Tm_C = integer(Tm*100) + BhmT/200,
    Tm_FLOAT is float(Tm_C)
    .

tm_max(CSTU_ID,TmM):-
    findall(Tm_FLOAT,tm(CSTU_ID,_,_,_,Tm_FLOAT),Tm_L),
    max_member(TmM,Tm_L).

choose_major(CSTU_ID,MAJOR_NAME):-
    current_stu(CSTU_ID,_),
    tm_max(CSTU_ID,TmM),
    Tm_FLOAT is TmM,
    tm(CSTU_ID,MAJOR_ID,_,_,Tm_FLOAT),
    major(MAJOR_ID,MAJOR_NAME).

%% -----------
%% User Friendly Interface
%% -----------

cstu(CSTU_ID) :-
    write('Current
Students:\nstudent_id\tstudent_name\n==========================
==================\n'),
    forall(current_stu(CSTU_ID,CSTU_NAME),writef('%w\t%w\n',[CSTU_ID,
CSTU_NAME])),nl,nl.
cstu:-cstu(CSTU_ID).

fc:-
    write('Future
Courses:\ncourse_id\tcourse_name\n==========================
==================\n'),
    forall(optional_future_course(CSTU_ID,OPTIONAL_COURSE_ID),writef(
'%w\t%w\n',[CSTU_ID,OPTIONAL_COURSE_ID])),nl,nl.
```

```
m :-
    write('Optional
Major:\nmajor_id\tmajor_name\n================================
=============\n'),
    forall(major(MAJOR_ID,MAJOR_NAME),writef('%w\t\t%w\n',[MAJOR_I
D,MAJOR_NAME])),nl,nl.

cstu_gs :-
    write('Current courses & scores of current
students:\nstudent_id\tcourse_id\tscore\n============================
==================\n'),
    forall(cstu_get_score(CSTU_ID,CSTU_COURSE_NAME,CSTU_COURSE
_SCORE),writef('%w\t%w\t%w
\n',[CSTU_ID,CSTU_COURSE_NAME,CSTU_COURSE_SCORE])),nl,nl.

sfstu(CSTU_ID) :-
    write('Similar former
students:\ncstudent_id\tfstudent_id\n================================
================\n'),
    forall(similar_fstu(CSTU_ID,FSTU_ID),writef('%w\t%w\n',[CSTU_ID,FST
U_ID])),nl,nl.
sfstu :- sfstu(CSTU_ID).

sfst_tc_take :-
    write('Future courses that similar former students has
taken:\ncstu_id\t\tfstu_id\t\tcourse_id(take)\n=======================
======================\n'),
    forall(similar_fstu_take_course(CSTU_ID,FSTU_ID,FSTU_COURSE_ID,_
),writef('%w\t%w\t%w\n',[CSTU_ID,FSTU_ID,FSTU_COURSE_ID])),nl,nl.

bhfc(CSTU_ID) :-
    write('The number of similar former students who took/passed future
courses:\nstudent_id\tcourse_id\tBh1(fc)\tBh2(fc)\n==================
============================\n'),
    forall(bhfc_take_pass(CSTU_ID,OFC,BhfcT,BhfcP),writef('%w\t%w\t%w\t
%w\n',[CSTU_ID,OFC,BhfcT,BhfcP])),nl.
bhfc :- bhfc(CSTU_ID).

tfcall(CSTU_ID) :-
    write('The supporting degrees of the future elective courses
fc:\nstudent_id\tcourse_id\tT(fc)\n================================
==============\n'),
```

```
    forall(tfc(CSTU_ID,OFC,Tfc_FLOAT,_,_),writef('%w\t%w\t%w\n',[CSTU_
ID,OFC,Tfc_FLOAT])),nl.
tfcall :- tfcall(CSTU_ID).

tfcfg :-
    write('The supporting degrees of every future elective courses in
groups:\ngroup_id\tcourse_id\tT(fc)\n==============================
=======================================\n'),
    forall(tfc_for_group(_,GROUP_ID,OFC,Tfc_FLOAT,_,_),writef('%w\t%w\t
%w\t\n',[GROUP_ID,OFC,Tfc_FLOAT])),nl.
tfcmax :-
    write('The recommended courses for group
is:\ngroup_id\tT(fc)\n=======================================
==\n'),
    forall(tfc_max(_,ECGNum,TfcM),writef('%w\t\t%w\n',[ECGNum,TfcM])),
nl.

ccl :-
    write('The recommended courses for elective courses
group:\nstu_id\t\tgroup_id
T(fc)\tcourse_id\n=========================================
================\n'),
    forall(choose_course_list(CSTU_ID,GROUP_ID,TfcM,OFC_L),writef('%w
\t%w\t%w\t\t%w\n',[CSTU_ID,GROUP_ID,TfcM,OFC_L])),nl.

cc(CSTU_ID) :-
    write('The recommended courses for elective courses
group:\nstu_id\t\tgroup_id\tcourse_id\n==========================
==============\n'),
    forall(choose_course(CSTU_ID,GROUP_ID,OFC,_,_,_),writef('%w\t%w\t\t
%w\n',[CSTU_ID,GROUP_ID,OFC])),nl.
cc :- cc(CSTU_ID).

tfcfm:-
    write('the supporting degree of choosing the major
m:\nmajor_id\tT(m)\n=======================================
==\n'),
    forall(tfc_for_major(CSTU_ID,MAJOR_ID,GROUP_ID,BhfcT,BhfcP),writ
ef('%w\t\t%w\t%w\t%w\n',[CSTU_ID,MAJOR_ID,GROUP_ID,BhfcT,BhfcP]))
,nl.
%% csec :-
%%    write('The recommended courses for group
```

```
is:\ngroup_id\tcourse_id\tT(fc)\n==========\n'),
%%
    forall(cstu_select_elective_course(_,ECGNum,SC,Tfc),writef('%w\t\t%w\t%w\n',[ECGNum,SC,Tfc])),nl.

bhmtp(CSTU_ID) :-
    write('The number of similar former students who took future courses in majors:\nstudent_id\tmajor_id\tB1h(m)\tB2h(m)\n==========\n'),
    forall(bhm_take_pass(CSTU_ID,MAJOR_ID,BhmT,BhmP),writef('%w\t%w\t\t%w\t%w\n',[CSTU_ID,MAJOR_ID,BhmT,BhmP])),nl.
bhmtp :- bhmtp(CSTU_ID).

tm(CSTU_ID) :-
    write('the supporting degree of choosing the major m:\nstudent_id\tmajor_id\tT(m)\n==========\n'),
    forall(tm(CSTU_ID,MAJOR_ID,_,_,Tm_FLOAT),writef('%w\t%w\t\t%w\n',[CSTU_ID,MAJOR_ID,Tm_FLOAT])),nl.
%% tml :- forall(tm_list(_,TmL),writef('Tm = %w\n',[TmL])),nl.
%% tmm :- forall(tm_max(_,TmM),writef('TmMax = %w\n',[TmM])),nl.
tm :- tm(CSTU_ID).

cm(CSTU_ID) :-
    write('The current student is recommended to choose major:\nstudent_id\tmajor_name\n==========\n'),
    forall(choose_major(CSTU_ID,M),writef('%w\t%w\t\n',[CSTU_ID,M])),nl.
%% cml :- forall(choose_major_list(CStu,ML,Tm),writef('%w is recommended to choose major %w \n T%w = %w\n',[CStu,ML,MAJOR_NAME,Tm])),nl.
```

附录B 合格分数线变动对系统性能影响一览表

*注 1：相似学生的成绩浮动值设为 2。

*注 2：“选择”一列中，EE 代表“电子信息工程”专业，CS 代表“计算机科学与技术”专业。

合格分数线γ	被选课程总数	被选课程合格总数	合格率	B_{1h}(CS)	B_{2h}(CS)	T(CS)	B_{1h}(EE)	B_{2h}(EE)	T(EE)	推荐度标准差	选择
60	463	457	0.99	143	142	0.99	173	173	1.00	0.00	EE
61	463	442	0.95	140	131	0.94	165	165	1.00	0.03	EE
62	463	432	0.93	139	124	0.89	165	165	1.00	0.05	EE
63	463	424	0.92	139	122	0.88	165	163	0.99	0.06	EE
64	463	417	0.90	139	119	0.86	165	162	0.98	0.06	EE
65	463	408	0.88	139	118	0.85	165	159	0.96	0.06	EE
66	463	402	0.87	139	118	0.85	165	156	0.95	0.05	EE
67	463	391	0.84	139	116	0.83	165	153	0.93	0.05	EE
68	463	382	0.83	139	113	0.81	160	145	0.91	0.05	EE
69	463	366	0.79	139	106	0.76	165	142	0.86	0.05	EE
70	463	358	0.77	139	103	0.74	160	135	0.84	0.05	EE
71	463	358	0.77	133	95	0.71	160	133	0.83	0.06	EE
72	463	335	0.72	133	95	0.71	160	130	0.81	0.05	EE
73	463	327	0.71	133	91	0.68	160	128	0.80	0.06	EE
74	463	315	0.68	133	87	0.65	160	126	0.79	0.07	EE
75	463	300	0.65	133	83	0.62	160	119	0.74	0.06	EE
76	463	277	0.60	135	82	0.61	173	118	0.68	0.04	EE
77	463	266	0.57	132	79	0.60	168	110	0.65	0.03	EE
78	463	250	0.54	132	74	0.56	168	102	0.61	0.02	EE
79	463	250	0.54	132	74	0.56	168	102	0.61	0.02	EE
80	463	203	0.44	136	63	0.46	159	78	0.49	0.01	EE
85	463	118	0.25	131	46	0.35	178	48	0.27	0.04	EE
90	463	60	0.13	134	32	0.24	190	19	0.10	0.07	CS

参考文献

[1] Joseph Psotka, Sharon A. Mutter (1988). Intelligent Tutoring Systems: Lessons Learned. Lawrence Erlbaum Associates. ISBN 0-8058-0192-8.

[2] P.Brusilovsky, Adaptive and Intelligent Technologies for Web-based Education, KI - Kuenstliche Intelligenz, 1999, vol.13, no.4, 19-25.

[3] Zhou, Qing, and Fang Yu. "Knowledge-Based Major Choosing Decision Making for Remote Students." Computer Science and Software Engineering, 2008 International Conference on. Vol. 5. IEEE, 2008.

[4] 周立柱.Prolog 逻辑程序设计及应用 .北京:清华大学出版社,1991.191-192

[5] Blackburn, Patrick; Bos, Johan; Striegnitz, Kristina (2006). Learn Prolog Now!. ISBN 1-904987-17-6.

[6] Ivan Bratko, PROLOG Programming for Artificial Intelligence, 2000, ISBN 0-201-40375-7.

[7] William F. Clocksin, Christopher S. Mellish: Programming in Prolog: Using the ISO Standard. Springer, 5th ed., 2003, ISBN 978-3-540-00678-7. (This edition is updated for ISO Prolog. Previous editions

[8] Lloyd, J. W. (1984). Foundations of logic programming. Berlin: Springer-Verlag. ISBN 3-540-13299-6.

[9] Hyry．Prolog 和人工智能[EB/OL]．[2015-05-09] http://hyry.dip.jp/tech/book/index/prolog

[10] 周立柱.Prolog 逻辑程序设计及应用 .北京:清华大学出版社,1991.10

[11] 周立柱. Prolog 逻辑程序设计及应用. 北京：清华大学出版社，1991．29

[12] 周立柱．Prolog 逻辑程序设计及应用．北京：清华大学出版社，1991．1

[13] VanHelsing．七种 Prolog 解释器/编译器[EB/OL]．(2010-09-21) [2015-05-11] http://blog.sina.com.cn/s/blog_494e45fe0100lh1v.html

[14] Jan Wielemaker. SWI-Prolog ODBC Interface[EB/OL]. [2015-05-11] http://www.swi-prolog.org/pldoc/doc_for?object=section(%27packages/odbc.html%27)

[15] Jan Wielemaker. SWI-Prolog Reference manual[EB/OL]. (2014-05-23)[2015-05-11] http://www.swi-prolog.org/pldoc/doc_for?object=manual

[16] 李大方， 何守才．提高关系数据库与 PROLOG 之间的接口效率．见：第九届全国数据库学术会议．上海．1990-09-07． 第九届全国数据库学术会议论文集(下)．上海：何守才，复旦大学出版社，1990-08．1