CnX 索引子系统的设计与实现

学　　院　计算机科学与技术

专　　业　计算机科学与技术

年　　级　2004

姓　　名　王乐

指导教师　张坤龙

2008 年 6 月 15 日

摘　　要

CnX 索引子系统是一个完整的中文 XML 文献倒排索引的构建系统，它主要由中英文语义处理模块、倒排索引构建模块和 Okapi BM25 概率模型评分模块组成，论文设计并实现了基于 C/S 架构的多线程 CnX 索引子系统。

与普通的无结构文本文档不同，XML 文档是一种半结构化的文档，在构建 XML 文档倒排索引的时候要考虑在倒排索引中体现 XML 的结构信息。XML 文档的结构就是一棵树，它由很多的结点构成，它的结点又可以分为结构(内部)结点和叶子结点，通常认为只有叶子结点才会包含文本信息。对于每个叶子结点所包含的文本信息，可以跟检索平面文件的方式一样——直接进行简单的全文检索。

由于 CnX 索引子系统需要支持中文检索，所以在进行创建索引的过程中，先要对中文语句进行分词处理。然后根据 XML 文档的结构信息构建 tag-term 的语词对，在内存中调整 XML 树的结构，并形成一个虚拟的文档树对象。接下来通过运用前后序遍历算法处理这棵树，将 XML 文档的倒排索引存储到数据库中。最终在完成基本索引的创建之后，再使用 Okapi BM25 算法对所有索引进行评分，以供上层的核心查询程序使用。

CnX 索引子系统是一个完整 XML 信息检索系统的基础，对于 XML 信息检索系统的整体构建有着很重要的作用。

关键词：XML；中文；倒排索引；信息检索(IR)

ABSTRACT

CnX indexing subsystem is a complete indexing constructor of Chinese XML data. It is mainly composed by a Chinese-English semantic processing module, an inverted index building module and a scoring module which uses Okapi BM25 algorithm that is a kind of probabilistic model. This paper gives a design solution and an implement solution of the CnX，which is based on C/S structure of a multi-threaded subsystem.

This paper starts from the modern information retrieval technology, and considers the ways of retrievaling XML data at first, and then begin to discuss the demand analysis，the ways of design and implementing of CnX. XML is a kind of semi-structured data, how to store the structure information of a XML document must be considered when building the inverted index. The structure of a XML document just likes a tree of data structure, which is builded up by many element nodes. And the nodes it has also can be divided into inner nodes and leaf nodes, usually, the leaf nodes are considered that contain text content, and the inner nodes usually not. For the text of the leaf nodes, it can be retrievaled by a ways of full-text content, just like retrievaling a plain text file.

As CnX focuses on the Chinese indexing, the Chinese sentences need lexically analyzing at first between the full-text content retrievaling, and then builds the pair of tag-term according to the structure information of the XML document. Before building a virtual document object of a XML document, the structure of the memory tree must be adjusted to a conscious state. Through repeated handling this tree，the inverted index is stored into a database system at last. After building the index completely，CnX will score the stored index by Okapi BM 25 algorithm for the top of the core procedures to use.

CnX index subsystem is a complete XML based information retrieval system, it plays an important role in building the whole information retrieval system.

Key words：XML；Chinese Words；Inverted Index；Information Retrieval (IR)

目　　录

第一章　　绪论

XML 文档是一种半结构化数据格式文档，随着 XML 技术的发展，它已经成为了很多场合下数据交换和数据存储事实上的标准。同时，由于互联网的迅速发展和 XML 技术的广泛应用，使得互联网中 XML 文档格式的数据变得越来越多，这种数据量的日益剧增，加大了人们在互联网上查找有用信息的难度。近几年来，由于我国信息技术的高速发展以及互联网在我国的快速普及，中文 XML 文档信息也变得越来越丰富，所以就需要一个中文信息检索系统来帮助人们准确且高效地检索有用的中文信息，CnX 就是这样的一个中文信息检索系统。索引的构建对于任何一个信息检索系统都是至关重要的，CnX 同样也不例外。本论文主要的工作就是设计并实现 CnX 的索引子系统，为 CnX 信息检索系统的构建做必要的准备。

1.1　XML 信息检索技术

XML 文献集[1]就是一组或者若干组具有相同或相似结构的 XML 文档的集合，而 XML 信息检索的目的就是为了帮助用户在这些 XML 文献集中方便而快速地检索出对用户有用或相关的信息。XML 信息检索技术作为现代信息检索技术中的一个分支，它的重点依然是信息检索，而不是数据检索，因此确定信息的相关性同样是 XML 信息检索的核心。

1.1.1　XML 文档的重要性

XML 是 Extensible Markup Language（可扩展性标识语言）的简称，和通常所见的 HTML 一样，它也是一种“标记”语言。XML 牢牢植根于 SGML[2]，由于 SGML 本身因为太复杂难以移植到 web 上，所以就产生了一个该语言的缩略版。推出它的目的是为了能够在网络上按照现在 HTML 所利用的方法利用、接受和处理通用 SGML。XML 的设计宗旨是让网络应用程序的执行更容易，并且增强 SGML 和 HTML 的互动操作性能。XML 是一个严格符合 SGML 编写格式的被应用程序所利用的文档格式。XML 提供了一种结构化的数据显示，这种结构化的数据功能强大且易于使用。XML 不像 HTML 一样使用预定义的标签，而是允许设计者自行创建自己的元素及其标签，使得这些元素的标签既能够容易被人阅读，又能方便被机器阅读。

XML 通过自身的结构来描述文档的信息，它是一种创建可相互交换的结构化文档的方法[3]。当一个数据集中(某个会议的论文集)的所有文档都使用 XML 来编制时，可以定义 DTD 或者模式来创建 XML 文档，这样使得这个数据集中的所有文档都成为带有结构信息的文档——XML 文档。XML 文档的优点在于它自身是一种带有自描述标签树形结构文档，这完全不同于关系数据模型。正是因为 XML 文档的这种树结构，使得人们能够用一种更加自然的方式去直接管理数

据，而不是完全依赖于某个关系数据库软件来管理一些简单的数据。

每一个 XML 文档都是一棵树，它由很多的结点构成，而每一个结点有包含一个标签和对应的内容以及相对应的属性。在具体的处理过程当中，对于当前结点的属性，可以把它当作当前结点的子结点来看待的，这样这棵树就有了统一的结构，同时也不会破坏原 XML 文档的信息。对于树中的节点的内容，它要么是空，要么就是一段文字。通常来说，非叶子结点只是包含子结点和自己的属性(此时每个属性也作为自己的子叶子结点)，而只有叶子结点才拥有文本内容。但是对于既包含文本信息又包含子结点的结点和只包含纯文本的结点，在 XML 文档中也是存在。

XML 文档的每个结点都包含一定的文本信息，所以对于每一个结点的内容来说都可以称是全文本模型，同时可以这样定义一个 XML 元素(结点)的全文本内容：一个元素(结点)的全文本内容包括它所有子孙结点的全文本内容。这种统计思想也是源于 XML 文档的树型结构。

1.1.2 XML 信息检索的概念

XML 信息检索就是为了帮助用户在特定的 XML 文献集中准确且高效的地找出对用户有用或相关的信息，确定信息的相关性是 XML 信息检索的核心。在讨论 XML 信息检索之前，先介绍一下现代信息检索技术。

在过去的几十年中，信息检索领域已经得到了很大的发展，并且已经超越了它标记文本和在某一集合中检索出有用文献的最初目标。现代信息检索技术的研究包括建模、文献的分类和归类、系统构建、用户界面、数据可视化、信息过滤和查询语言等。

信息检索的核心是确定数据相关性，这不同于数据检索。信息检索的行为受到用户任务和检索系统所采用的文献逻辑视图的直接影响。用户的检索任务基本分为两种类型：信息或数据的检索和浏览，而往往用户的这两种类型的任务又是联系在一起的，如图 1-1 所示。

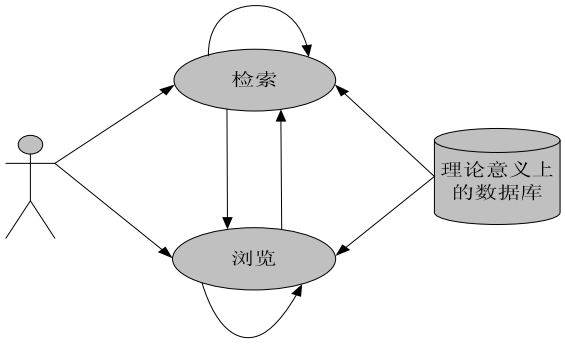

图 1-1　　用户任务间的联系[1]

现代计算机的出现使得采用完整的语词集合表示文献变为可能，在信息检索系统中可以使用全文本逻辑视图。然而对于规模很大的集合来说，采用全文本逻辑视图将会变得不是很合理，于是可以采用排除停用词、提取词干、归类同义词近义词和提取句子语义，同时可以采用压缩技术，实现逻辑视图从全文本到索引词集合的转移。

以计算机为中心和以人为中心是研究现代信息检索问题的两个基本观点，但是索引却是每个现代信息检索系统的核心。信息检索涉及到的其它实践问题有很多，比如说扫描、光学字符识别和跨语言检索等。

XML 信息检索作为现代信息检索中很重要的一个分支，现代信息技术的发展在很大的程度上推动了 XML 信息检索技术的发展。XML 信息检索大量运用现代信息检索技术所取得的成果，并结合 XML 自身的结构特性，在 XML 数据量日益剧增的背景下，XML 信息检索得到了很大的发展。

1.1.3 XML 信息检索技术的研究现状

Web 的广泛使用已经使得它成为了人类知识和文明的全球存储库，这个存储库史无前例的允许用户在无限的空间中实现思想和信息的共享，因此这个巨型知识存储库将会以一个非常快的速度无限增长。由于信息量的巨大、信息的类别多种多样，再加上 Web 自身结构的问题，使得人们很难在 Web 中检索出自己想要的信息。对于 XML 信息检索来说，关键问题还在于如何建模、标引并检索，这些关键的问题目前都是研究的热点。关于 XML 信息检索，国外由 IEEE Computer Society 赞助的 INEX 从 2002 年起，每年都要举办一次。

XML 信息检索的理论研究已经非常成熟，国外高水平论文非常之多，市场上存在的产品也非常丰富，如基于关键字查询的 Google，Yahoo！和 baidu 等。XML 技术也发展得非常健全，WEB 上的很多数据都已经以 XML 的结构方式进行存储，如维基百科的海量数据，同时，维基百科也是使用关键字来检索 XML 文献集的。

XML 文档不同于简单的平面文档，对于平面文档，可以直接对其进行全文检索，建立倒排索引文档，然后根据某种或某几种算法进行关键字检索。但是对于 XML 文档来说，应该怎样构建带有结构信息的倒排索引文档呢？又能不能扩展查询方式，而不限于关键字查询呢(比如使用定义严格的查询语句)？信息检索本身就具有模糊性，那么又应该怎样利用 XML 的结构信息来提高检索结果的准确率呢？目前有很多人都在关注这些问题，国内外也已经取得了一些 XML 信息检索技术的成果，基于 XML 的信息检索系统也已有原型，例如国外的 Max-Planck 信息研究所的 TopX[4]和国内的中国人民大学 WAMDM 实验室的 OrientX[5]等。在中文环境中研究 XML 信息检索是完全符合中文 XML 数据日益剧增的事实的，改善中文 XML 信息检索的需求也越来越受到更多的关注。

1.2 XML 信息检索系统

XML 信息检索系统就是应用 XML 信息检索技术，并结合其它的软件工程的方法和思想构建的信息检索系统软件，人们不必了解它的具体工作原理，而是直接通过合理使用它，就可以在特定的 XML 文献集中找到有用或相关的信息。本论文介绍的 CnX 就是这样的一个 XML 信息检索系统。

1.2.1 XML 信息检索系统的概念

XML 信息检索系统是一套完整的软件系统。它主要的功能包括 XML 文献集的倒排索引文档的构建和利用倒排索引文档，并使用适当的检索方法，根据用户的需求，帮助用户在特定的 XML 文献集中高效而准确的检索出有用或相关的信息。

1.2.2 XML 信息检索的基本流程

XML 信息检索系统使用 Top-k[2]查询方法，并且有一定的查询扩展能力。XML 信息检索的主体流程与一般的信息检索流程仍然大体相似，只是会在一些细节的地方侧重不同，比如说 XML 文档的倒排索引的数据模型应该是怎样的，数据结构应该如何构建以及如何解决查询处理的相关问题等。

XML 结构上的优势使得它可以比较容易的支持自己的检索的语言，而不仅只是使用在检索框中输入几个关键字的方式进行检索。下面用 XML 信息检索系统的结构图 1-2 来说明 XML 信息检索的基本流程。

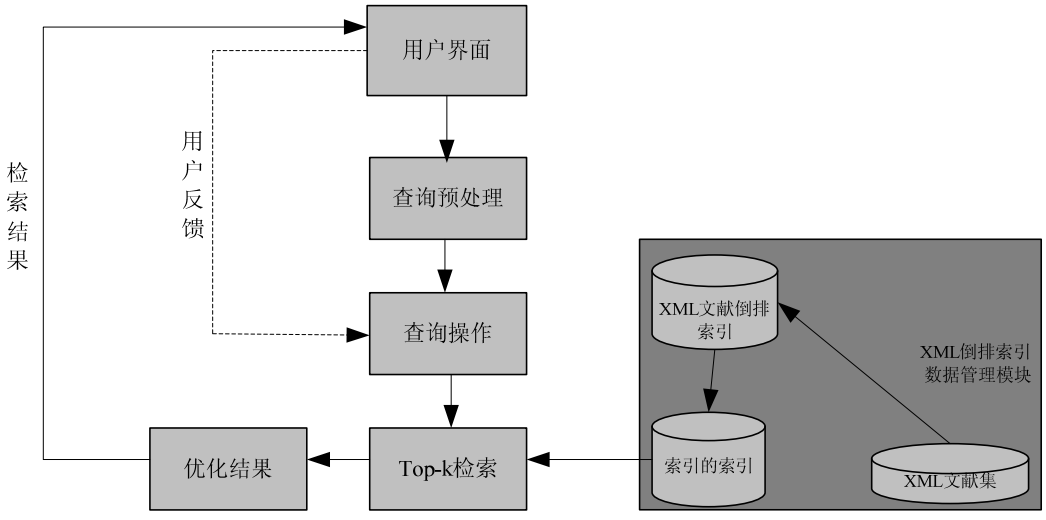

图 1-2　　XML 信息检索一般流程

在图 1-2 中可以看到，XML 信息检索需要先根据一定的数据模型构建合理的倒排索引，然后用户输入自己的查询描述，系统将会预先处理这些描述，然后根据一定的算法对检索结果进行排序，或者是根据排序得出结果，其中也可以加

入用户的反馈环节。

1.2.3 中文 XML 信息检索系统 CnX

本论文侧重优化中文 XML 文献集的检索系统(索引部分)的设计与实现，CnX 就是这样的一个侧重于中文的 XML 信息检索系统。

由于中文和英文(或其他西欧语言)的信息检索有着很大的不同，英文语句中的单词与单词之间都有空格，而中文却没有这个有利于构建倒排索引的便利条件，那么 CnX 如何对中文语句进行分词呢？这方面中科院的基于多层隐马尔可夫模型的汉语词法分析系统 ICTCLAS[6]为 CnX 提供了很好的解决方法。同时为了提高索引的效率，CnX 还对自然语句进行必要的语义处理，降低索引的冗余性。关于语义的处理，目前也有很多人正在研究这方面的课题，语义的处理将会提高索引的质量，并改善用户的信息检索体验等。

1.3 CnX 索引子系统

XML 信息检索系统 CnX 与 TopX 有着很深的渊源，CnX 使用了很多 TopX 里 XML 信息检索的思想，保留了 TopX 构建 XML 文档倒排索引的数据模型，然后使用了面向对象技术加以重新实现，并且还实现了简单的语义处理和中文索引功能等。

CnX 索引子系统是 CnX 信息检索系统的重要组成部分，其主要功能就是构建 XML 文档的倒排索引，并使用 Okapi BM25 算法[7]对倒排索引进行评分，以供上层核心查询程序使用。它的主要处理过程包括：解析中文 XML 文档，将 XML 文档映射成具有统一结构的树，构建 XML 文档的倒排索引和对倒排索引中语词对(tag-term)、文档结点元素以及文档进行全局评分等(评分结果也存储在数据库中，成为索引的一部分)。

系统分为以下几个模块，如图 1-3 所示。

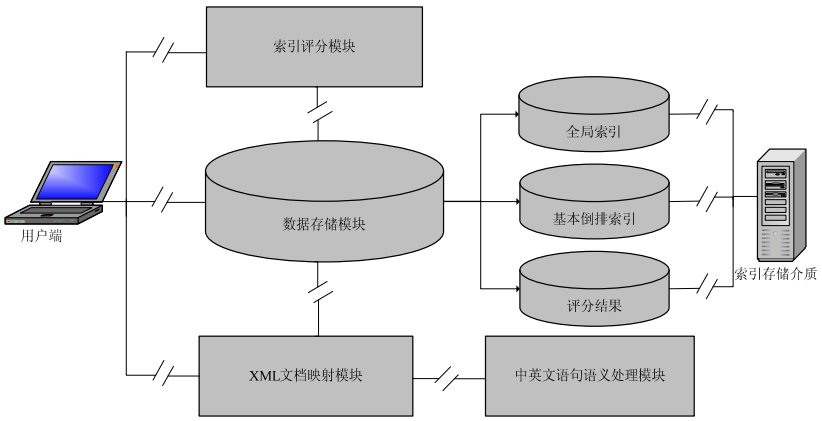

图 1-3 CnX 索引子系统的逻辑结构

CnX 索引子系统采用 C/S 架构，将倒排索引文档存储在数据库中，数据存

储采用持久化技术。其中倒排文档的构建是该系统的关键部分，这个倒排索引文档包含了 XML 的结构信息，除了去掉了那些没有语义的词和用户指定的停用词，整个倒排索引几乎是原文献的一个异构的备份。CnX 索引子系统在分析处理并完成倒排索引的构建后，可根据用户的需求对索引中语词对进行评分，为以后的检索做必要的准备。其中评分采用的是 Okapi BM25 概率评分模型，通过顺序扫描索引列表快速计算各个项的分数。系统采用 C/S 架构，这样有利于系统的分层部署，又由于系统的数据存储完全使用了持久化技术，这样就更易于把系统部署到不同的数据库系统上。同时，数据库系统在该系统中仅仅承担着数据存储的任务，系统不依赖于数据库系统。数据存储部分可以由用户另外单独实现，也可以实现持久化数据存储接口，将最后的索引数据以一个合理的方式直接保存在文件系统之上，这样系统将会具有更强的独立性。

该子系统采用多线程并行设计，人机交互接口良好。多核时代的到来，大大提高了多线程并行程序设计的重要性，本系统也采用多线程设计，可在界面上设置同时运行的最大线程数，每个单独线程将会处理一个 XML 文档，但是多线程并行访问全局索引的时候，就必须要考虑到线程间的资源共享和线程间的同步问题。通过锁的机制来控制关键区，以保证关键区在每个时刻都只有一个线程在访问，用户不必关心具体是哪个 XML 文档正在被处理，当前被处理的所有文档的内容都会实时的保存的各自的线程空间里。所有线程处理完毕，程序可以显示完成状态等

1.4 论文的研究内容和意义

本论文主要探讨了如何在 XML 文献集上高效而准确地检索信息，根据这个总体的需求，合理应用了信息检索领域里的相关理论和方法，设计并实现了一个 XML 信息检索的索引子系统。该索引子系统是整个检索系统的基础，对于检索系统的整体构建有着很重要的意义。

- 研究内容

  本论文从构建 CnX 信息检索系统的大局出发，侧重其索引子系统的设计与实现。XML 信息检索系统主要分为倒排索引的构建和信息的检索，而本论文侧重的则是构建 XML 文献的倒排索引，通过学习现代信息检索各方面的知识，并结合 XML 技术发展的现状，本论文研究的内容主要包括：如何高效构建 XML 文献的倒排文档，如何建模、标引并检索，如何高效利用 XML 的树型结构，采用何种评分模型最有利于 XML 文献的检索等；另外本论文同时注重系统本身的构建方法，在设计和实现过程中，使用了一些面向对象的技术，如 Java、JNI、持久化技术、反射技术等。本论文针还对中文 XML 文献集的检索做了不少的研究工作，主要包括如何提高中文的分词的准确率和效率和如何区分中英文词干的提取等等。

- 研究意义

中文 XML 文献集的数据量越来越大，人们在这些文献集中查找信息变得越来越困难，而 CnX 旨在帮助人们解决这个困难，所以本论文的研究具有一定的实际意义。同时，本论文在研究的过程中，结合了现代信息检索的理论和 XML 文档的结构特征以及中文的特点，开展了中文 XML 信息检索的研究工作，并取得了一些成果，为下一步的 XML 信息检索打下了很好的基础，因此又具有一定的理论意义。

1.5 论文的组织结构

第一章介绍了 XML 信息检索技术的相关概念和研究现状，简要地描述了 XML 信息检索系统 CnX 的索引子系统的体系结构，说明了论文的研究内容和研究意义。

第二章主要分析了 CnX 索引子系统的需求。

第三章在理解第二章需求的基础上，给出了 CnX 索引子系统具体的设计方案。

第四章根据第三章 CnX 索引子系统的设计，详细阐述 CnX 索引子系统的实现中使用到的技术和方法。

最后一章，即第五章，对本文的工作做出总结，并指出未来可做的工作。

第二章　CnX 索引子系统的需求分析

2.1　总体需求

CnX 索引子系统需要实现的基本功能就是能够根据一定的索引数据模型构建中文 XML 文献的倒排索引，并能够根据 Okapi BM25 算法对索引进行评分。其中在构建中文 XML 文献倒排索引的时候，需要简单处理中文语句的语义和提取英文单词的词根，并通过添加用户词典或停用词过滤技术提高 XML 文献倒排索引的质量。系统还应该能够并行处理多个 XML 文档，提高处理效率，并能实时向用户反馈处理进程信息。

总体需求应该分为以下几个部分：

①人机交互接口(程序的用户界面)：此部分的设计应该符合界面程序的标准，满足用户的需求，符合用户使用其它应用程序的习惯。

程序主要使用 Java 实现，所以界面可以采用 Java 的 Swing[8]来实现，另外要求调用本地画图接口，实现本地程序风格，例如在 Windows XP 环境下的程序应产生 XP 风格的界面等，如图 2-1 和图 2-2 所示。

图 2-1　Swing 原生态风格的界面

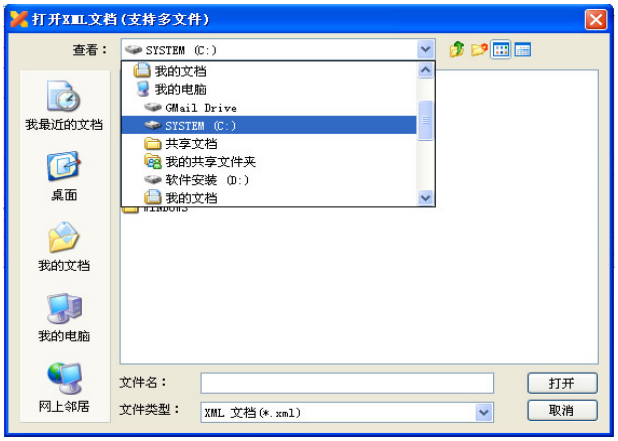

图 2-2　XP 风格的界面

另外在程序运行的过程中，程序应该能够实时地反馈必要的处理信息和处理状态给用户。

②功能需求：系统能够正确处理中文 XML 文献集中的文档、自行定义中文分词短语词典(图 2-3)和自行定义停用词词典(图 2-4)。

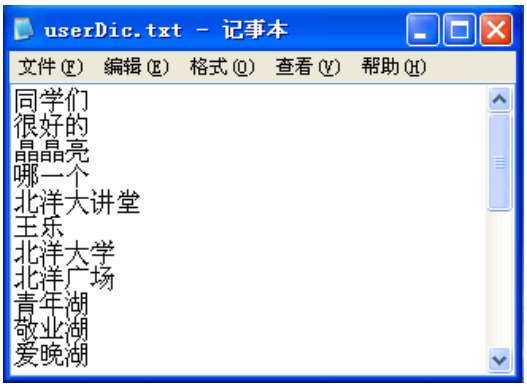

图 2-3　　用户自定义分词词典

图 2-4　　用户自定义停用词词典

系统应该把索引的构建过程和索引的评分过程分开，使得系统既能够在构建索引之后立即评分，也能够随时对索引进行评分。对于中文分词和词干提取，系统也应该是可选的，同时，对于任意一个 XML 文献集，系统都能够提取其 XML 文档的自描述信息，以供上层查询程序使用，系统还应该能在运行的过程中随时被停止。另外，对于倒排索引的存储可以使用关系数据库，并且不能依赖于某一个关系数据库等等。

③结构需求：系统需采用 C/S 架构，将数据存储和数据处理分开部署到不同的设备上，系统应采用并行化设计，能够同时处理多个 XML 文档等。

④运行环境需求：系统应具备跨平台运行能力，应该能够非常容易的就能部署到主流的平台上，如 Windows 和类 Unix 平台上等。

2.2 倒排索引结构需求

倒排索引可以直接存储在文件系统上或者存储在数据库系统里，考虑到实现上的复杂度，可以选择将索引存储在关系数据库里，然后再利用关系数据库的功能对索引进行评分处理。所以就需要先在数据库中构建存储索引的表结构，考虑到 XML 文档倒排索引的特点——需要存储 XML 的结构信息，倒排索引结构表的需求如下(以下结构依赖于构建的 XML 映射树)。

表 2-1 oneindex 表

字段名	字段含义
path	XML 结点的 XPath 路径
bucketId	全局 XPath 的唯一编号

表 2-2 xPath 表

字段名	字段含义
eid	全局元素唯一编号
Xpath	当前元素的 XPath

表 2-3 documents 表

字段名	字段含义
did	文档全局编号
fileId	文档自描述编号
author	文档作者
title	文档题目
preview	预览内容
uri	文件定位符

表 2-4 elements 表

字段名	字段含义
did	元素所属文档编号
tag	元素名
pre	前序遍历编号
post	后续遍历编号
lev	元素(结点)所属层
bucketId	结点的 oneindex 标识
cpts	结点语词总数

表 2-5 features 表

字段名	字段含义
did	元素所属文档编号
tag	元素(结点)名
term	元素包含的单个语词

pre	前序遍历编号
post	后续遍历编号
lev	元素(结点)所属层
bucketed	元素 oneindex 标识
tf	语词在元素中的出现频率

表 2-6　terms 表

字段名	字段含义
did	语词所属文档编号
term	语词
off	语词在当前元素中的位置
pre	前序遍历编号
post	后序遍历编号

在构建完 XML 映射树后，接着就将各项信息存储到上述的各个结构当中去，下面对上述存储 XML 倒排索引的结构做一些简要的说明：

- oneindex：这个结构存储了 XML 文献集的全局信息，每一个路径唯一的结点都会有一个唯一的编号，以供全局评分使用。
- xpath：整个 XML 文献集中所有的元素的 XPath 路径都存储在这个结构里，它配合 oneindex 以供全局评分使用。
- documents，elements，features，terms：在 XML 映射树中表现为 documents 包含 elements，elements 包含 terms，features 描述 elements 的特征，于是将这种关系直接保存在了这些结构中。

通过构建以上结构，XML 文献的倒排索引就可以完整的构建了，接下来就是对这些以构建的倒排索引进行评分。首先介绍定义带有标签 A 的结点 N 下的所有语词的分数表示方法：

- 全文语词出现频率，ftf(t,n)，即语词 t 在结点 n 下的所有文本中出现的总次数。
- 标签出现频率，N_A，即 A 为结点 N 的标签，标签为 A 的结点在整个文集中出现的次数。
- 元素出现频率，$ef_A(t)$,即标签为 A，并且包含语词 t 的结点在整个文集中出现的次数。

下面考虑如何计算 A//“t_1，t_2，……，t_m”的分数，其中 A 为标签名，“t_1，t_2，……，t_m”表示可能在标签为 A 的子树中出现的语词。为了避免短元素里的少数语词的高频率问题，这里使用比较高级的 Okapi BM25 评分模型(源于文本文档的概率检索模型)，以下是该模型的评分计算公式：

$$\text{score}(n,//A[t_1,t_2,\cdots\cdots;t_m]) = \sum_{i=1}^{m} \frac{(k_1+1)ftf(t_1,n)}{K+ftf(t_1,n)} \cdot \log\left(\frac{N_A - ef_A(t_i)+0.5}{ef_A(t_i)+0.5}\right) \qquad (2\text{-}1)$$

其中：

$$K = k_1\left((1-b)+b\frac{\sum_{t\in n} ftf(t,n)}{avg\left\{\sum_{t'} ftf(t',n')\middle| n'\right\}}\right) \tag{2-2}$$

n 为节点 n 下所有语词的集合，n′ 表示标签为 A 的结点。

在该子系统中，取 k_1 为 1.25，b 为 0.75。然后根据上述的 Okapi BM25 评分公式对所有文献进行全局的评分统计，最后得出每个元素(结点)的最大分数和全局语词对(tag-term)出现频率：

表 2-7　elements_maxscores 表

字段名	字段含义
tag	名为 tag 的元素
maxelementscore	最大分数

表 2-8　dfvalues 表

字段名	字段含义
tag	元素名
term	语词
df	语词对全局出现次数

评分是构建 XML 倒排索引过程中的很重要的一部分，它的构建是建立在前者(基本倒排索引)之上的，可以完全在关系数据库中完成评分的工作。到这里，XML 的倒排索引就构建需求就分析完了。

第三章　　CnX 索引子系统的设计

3.1　开发和运行环境

这里先讨论系统的开发和运行环境。

3.1.1　系统开发环境

CnX 索引子系统主体部分采用 Java 2 standard 标准开发，中文分词和中文语词提取模块是在中科院中文分词程序 FreeICTCLAS 基础上使用标准 C++语言开发完成的，英文单词词根提取模块是基于 snowball 西欧多语言词根提取程序开发而来的，数据库部分使用了标准的 SQL 语言，对于 XML 的解析则采用了十分流行的 DOM4j 和对 XPath 完全支持的 jaxen。在课题的研究以及程序开发过程中主要使用到的工具和标准如表 3-1 所示。

表 3-1　　开发环境

名称描述	具体内容
开发语言	Java Release 2 ISO C++ ISO SQL
架构模型	C/S 架构，Model 2
开发平台	Microsoft Windows XP professional 版本 2002 Service Pack 2 JDK1.6.0_03
IDE 工具	Myeclipse6.0 Visual Studio 2005 Netbeans 6.0
数据库管理系统	Mysql 5.0.45

3.1.2　系统运行环境

系统使用了 Client/Server(客户端/服务端)的设计架构，系统在逻辑上可以采用分层的部署方式。客户端可以部署在性能要求一般的个人 PC 上，但是由于客户端承担了很大一部分的运算工作，所以客户端的计算机应该尽可能的性能高点；对于服务器端，其主要的工作就索引数据的存储和评分。

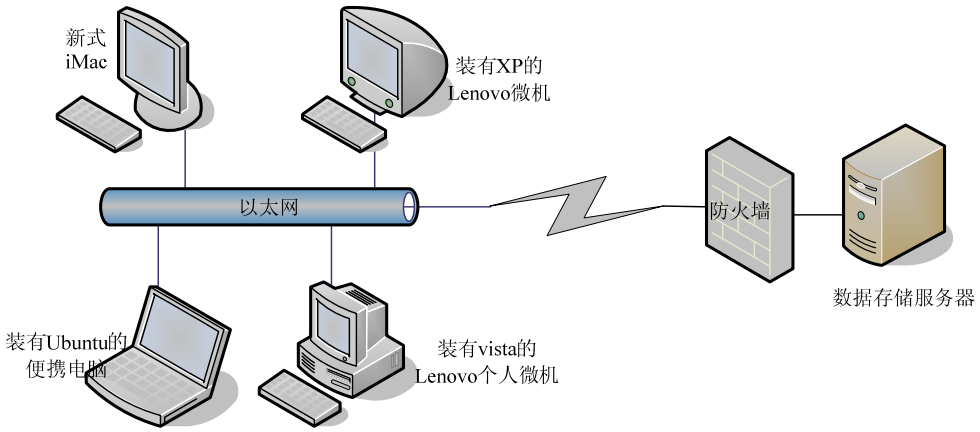

图 3-1　　系统逻辑部署图

系统的运行应该不依赖于具体的环境，系统主体部分需要采用 Java 开发，并且其它部分也应该使用 ANSI 或 ISO 标准开发，保证了系统至少在代码级别上实现跨平台。

3.2　总体架构设计

3.2.1　自顶向下的设计

■　系统宏观主体执行流程

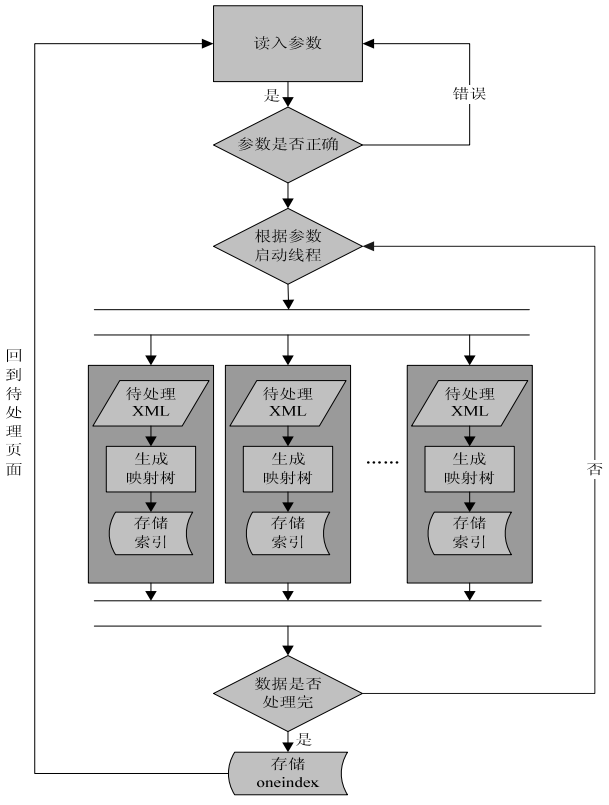

图 3-2　　CnX 主体执行流程

■ 系统的人机交互接口(参数的输入和结果的输出)

程序的图形界面采用 Swing 编写，并且具有宿主操作系统的风格。通过界面对象的传递，CnX 索引系统可以获取所有的待处理 XML 文档和相应的处理参数，同时并能够实时显示处理状态，最终可供用户查看处理结果。界面的显示和 XML 文档的处理部分都为单独的线程,界面的实现线程主要负责参数的输入和处理状态的输出，后台的处理线程从前台接受参数，处理 XML 文档，然后将结果存储到数据库中去。

■ 一个实例

下面用一个具体的例子来说明 CnX 索引子系统设计的基本工作原理。

CnX 索引子系统作为 CnX 索引子系统的重要组成部分，它主要的功能就是高效的构建中文 XML 文献的倒排索引，并根据上层核心查询的算法要求对倒排索引进行评分，这里用一个具体的例子来说明 CnX 索引子系统的功能需求，假设待处理的 XML 文档如图所示：

```
<? xml version="1.0" encoding="GBK"?>
<文章 name="do it">
  <编号>TJU-000000001 </编号>
  <作者 author="王乐">王乐 </作者>
  <标题 title="XML Data Management">XML Data Management Data </标题>
  <摘要> 管理系统 XML management systems vary widely in their expressive power.
  </摘要>
  <正文>
    <标题 1>Native XML Data Bases. </标题 1>
    <段落 1 time="2008-05-09">Native XML data base systems can store schemaless data.
    </段落 1>
    <评注>这是一简单的测试</评注>
    <标题 2>北京奥运会，王乐天津大学北洋广场 </标题 2>
    <段落 2>和平奥运，绿色奥运，人文奥运</段落 2>
  </正文>
  <参考文献> (1)文献 1 (2)文献 2</参考文献>
</文章>
```

图 3-3 XML 文档

它的数据模型如图 3-4 所示(其中的具体内容略)。

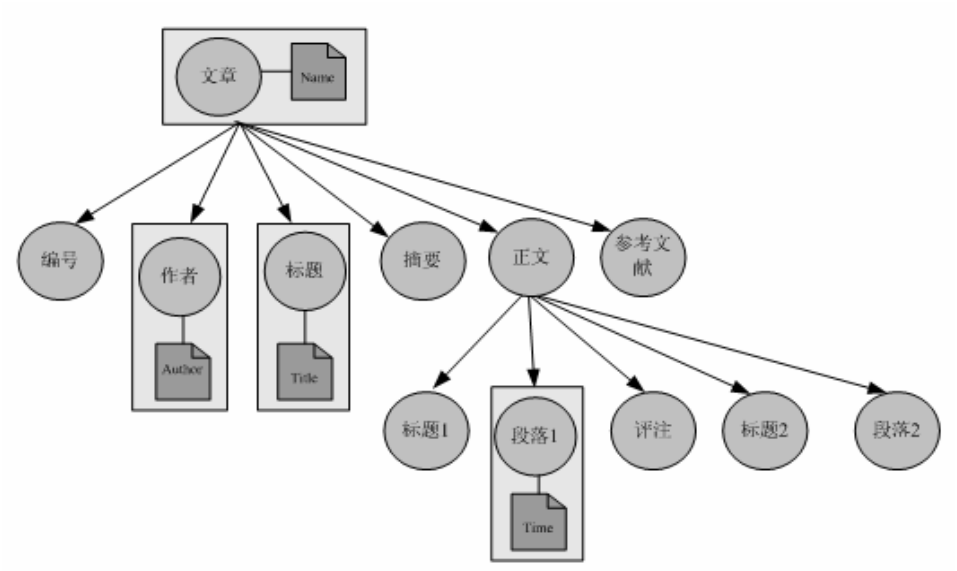

图 3-4　　XML 文档树结构

其中方框表示带有属性的结点。这种原生态 XML 的树结构不容易进行操作，为了更好更方便的构建 XML 文档的倒排索引，就需要调整该树的结构，具体做法是将结点的属性映射为该结点的直接子结点，并位于所有已存在的直接子结点之后。在完成映射树的构建之后，直接进行前后序遍历更新树的信息[9]，为构建倒排索引做必要的准备，完成此步后，映射树的样子如图 3-5 所示。

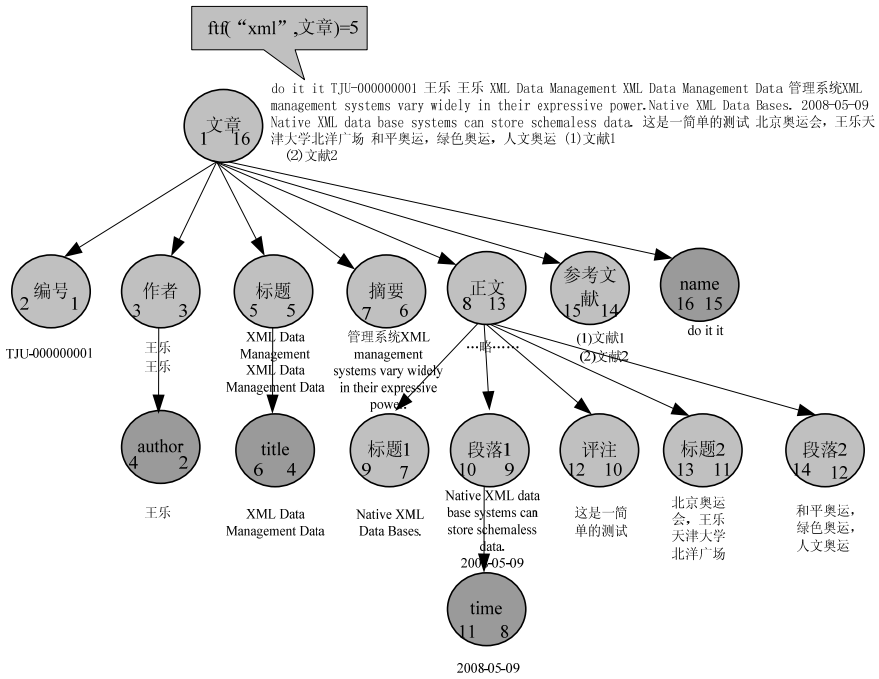

图 3-5　　带有冗余信息的 XML 结点树

结点左右侧的编号分别为前后序遍历的序号，这里创建结点的前后序编号是有原因的，上层的核心查询程序的算法实现需要判断结点的父子关系[10]，所以这里在建倒排索引时，就先为其准备条件，然后上层只需要根据公式 3-1 就可以很容易的判断出结点的父子关系：

$$v\text{是}v'\text{的子结点} \Leftrightarrow pre(v) > pre(v') \wedge post(v) < post(v') \qquad (3\text{-}1)$$

到这里，索引的基础部分就构建完成了，图 3-5 的冗余树包含了构建倒排索引所需的所有信息，接下来就是利用数据持久化技术将这颗树按照原先的倒排索引结构需求存储下来，对于索引的结果就可以直接在数据库中查看了。最后使用 Okapi BM25 算法对已构建的索引进行评分，当然评分的结果也存储在数据库中。

3.2.2 功能模块的设计

图 3-6 展示了 CnX 索引子系统的总体架构以及各个模块之间的关系，并且在一个线程空间里组织了各个模块之间的关系，阐明了设计的具体思路。

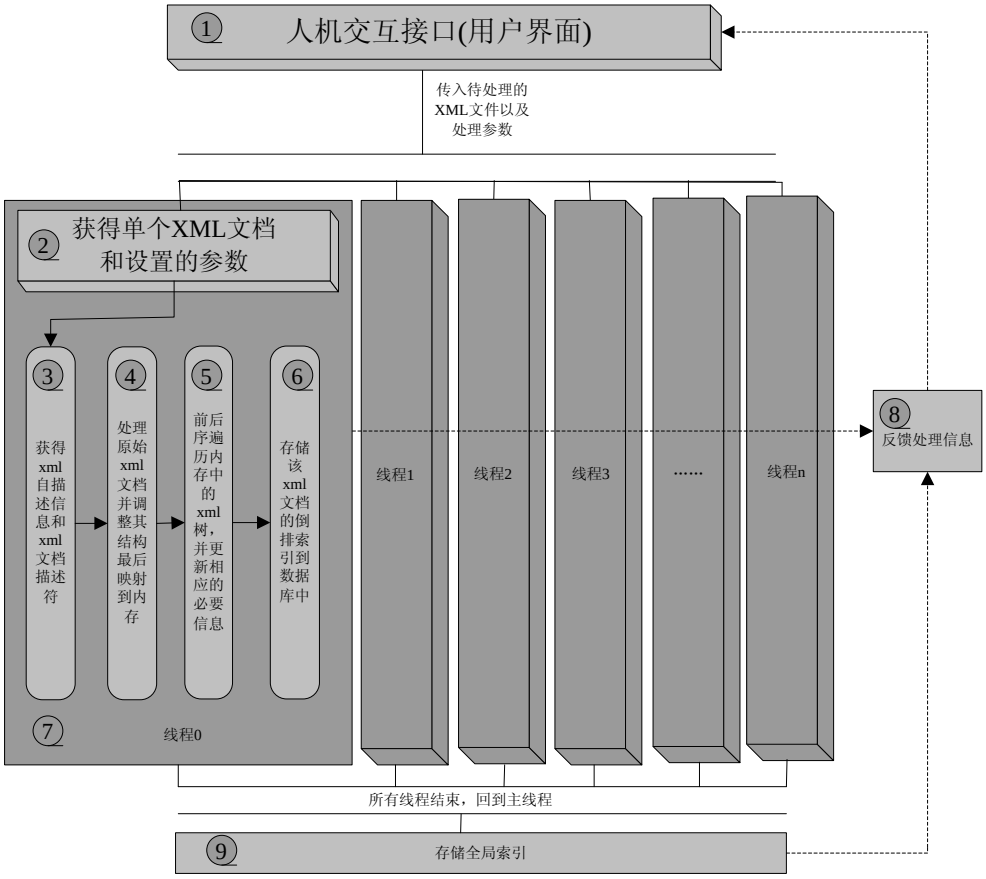

图 3-6　　CnX 索引子系统的逻辑模块

- 倒排文档模块
  - XML 的结构调整方案和倒排文档存储结构

    此模块是将原 XML 文档映射到内存中，所以需要首先定义内存中

映射文档的结构。例如 XML 文档，如图 3-3 所示，

映射到内存中时，将会把结点的属性映射为结点的子结点，上面的 XML 文档映射到内存中时，它的树结构将与图 3-5 相同。

关于倒排文档的存储结构，在第二章已经简要的说明了，所以在这里只是说明一下映射文档以及其结点的结构。

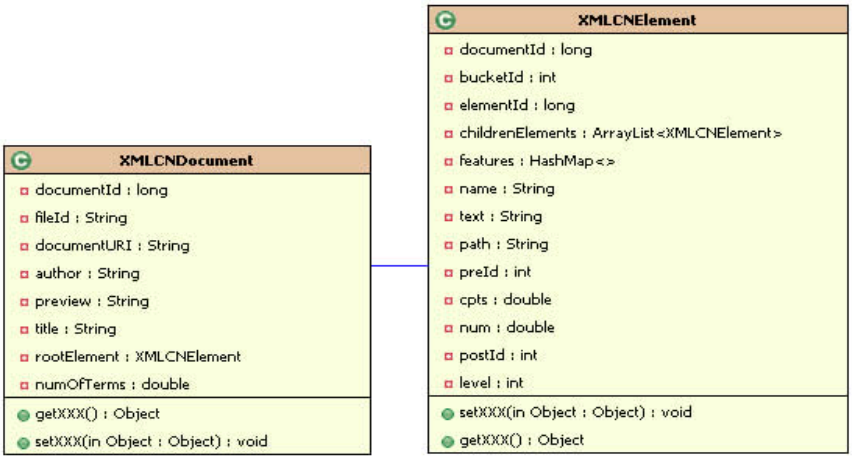

图 3-7　　映射文档以及其结点结构[11]

在映射文档的数据结构中，包含了文档本身的一些自描述的信息，最为重要的是它拥有文档的根节点，即整个文档的入口，以后的数调整的算法的实现都是从这个根结点(rootElement)开始，同时在结点的数据结构中，它通过一个链表容器来保存它所有的当前子结点，然后这样递归的保存下去，就形成了一棵完整的 XML 内存映射树。

- 中英文语义处理

这部分主要基于一些开源的程序来实现的。

对于中文分词和语词提取部分，CnX 结合了中科院的 ICTCLAS 的开源版本的代码，考虑了 CnX 索引子系统的要求，在 ICTCLAS 的基础上需要添加以下静态方法接口：

FreeICTCLAS_ImportUserDic：加载用户字典

FreeICTCLAS_Init：中文分词模块的初始化

getResultOfParagraph：获得带有词性的分词结果

此部分使用 JNI 技术实现，考虑到多线程同步问题，需要在本地方法前使用 synchronized 注明为同步方法。

对于英文处理部分，主要功能就是提取单词的词根，而且此部分在中文分词过程中执行，因此需要将其与中文分词部分整合在一起，提供一个统一的调用接口，并能够由用户自行选择是否进行提取语义和过滤

停用词，以下为需提供的接口：

ParagraphProcess(

boolean isCnStem,

boolean isEnStem,

boolean isFilterStopWords,

StopWordsFilter stopWordsFilter)：对外的初始化接口

getResultOfPara：最终的处理文本方法，返回结果

- 全局语词对评分模块

此模块在索引构建完成并成功存储之后运行，它主要就是分析已经构建索引并将索引的分数存储到数据库中，于是会有大量的数据库操作，因此这部分使用 JDBC 直接实现，以提高程序的效率。由于 Okapi BM25 算法使用的数学模型不是很复杂，可以直接使用 SQL 脚本来进行计算，于是可以把这部分全部用 SQL 脚本实现，然后通过 JDBC 调用即可。

- 持久化模块(数据存储模块)

索引的存储使用 Hibernate 持久化框架，Myeclipse 的 Hibernate 插件非常自动化，可以自动生成 O/R-Mapping 配置文件、POJO 类和 DAO 方法，但是考虑到它自动生成的代码为了通用性而降低了执行性能，于是加入了以往项目中使用的 Hibernate 的接口，以提高程序的性能。

3.3 其它的设计

- 全球化设计

程序应该使用全球化设计方法。在 CnX 索引子系统中，应该单独编写资源文件的管理方法，由于程序主要就是界面上需要全球化，所以可以动态的加载那些资源文件。

在 CnX 中的做法与其它的方法一样，现将那些信息放在一个文本文件中，有求一行一个标识，且属性与值之间用“=”连接：

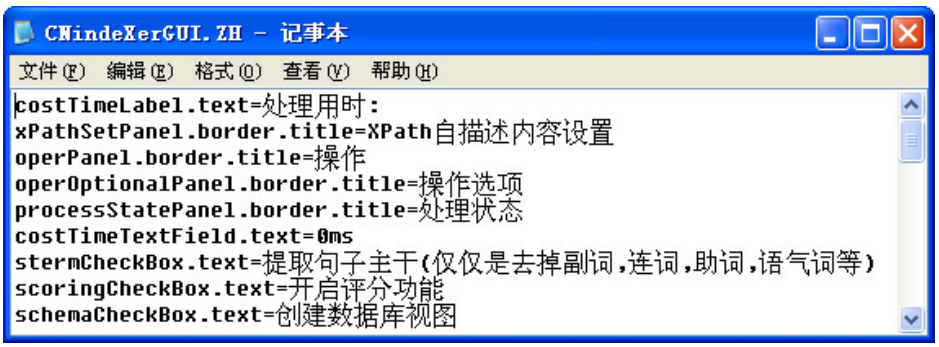

图 3-8　　资源属性与值

对于这种结构的资源文件，可以很容易的想到使用哈希表来作为它的容器，资源属性与值及对应着哈希表的键—值。系统应该提供一个统一的方法，供其他对象调用。

■ 跨平台和分布式

CnX 子系统主体部分采用 Java 开发，中文分词和中文语义处理部分采用标准的 C++开发，然后通过 JNI 技术调用中文分词和中文语义处理模块 DLL(在类 UNIX 系统上则为 so)。由于系统主体采用了跨平台的 Java 技术，因此只需针对某种平台改动少许代码就可以实现跨平台。对分布式计算技术的支持，系统也应该适当考虑，以便于充分利用身边可用的计算资源。CnX 索引子系统采用多线程设计，只要在线程并发上适当考虑分布式即可，可将主要的运算环节转移到计算性能较为强大的分布式计算设备上，将客户端设计为真正意义上的终端。

■ 提高系统健壮性

对于使用 Java 构建的应用程序，系统本身就已经具有很强的健壮性，但是无论语言有多么的强大，它也不能在逻辑的角度保证应用程序的健壮性。所以在程序的设计过程中，一定要首先尽可能的保证程序的在逻辑上正确性，然后再在细节方面保证程序的正确性。

在程序的设计过程中，要充分考虑程序的自身的问题，比如说 Java 和 C++的异常捕捉处理，C++的空指针和地址访问越界问题，缓存区溢出问题等等，类似的这些问题都需要在程序设计和实现过程考虑到并应该合理的解决。同时还应该考虑到程序运行的环境问题，比如说 Java 虚拟机的崩溃、操作系统的崩溃、数据库系统的崩溃或者文件系统的崩溃等等，这些都是无法在程序运行时预知的，但是通过预先考虑这些环节并进行合理解决将会大大提高系统的健壮性。

■ 测试用数据的设计

在这里简要的说一下测试用例的设计。CnX 索引子系统的主要功能即使处理 XML 文档，这里需要准备一定的 XML 测试文档，这些文档的结构应该满足 XML 的标准定义。其中 XML 中结点的内容应该包含中英文，或者只包含中文或者英文，同时为了提高程序性能的总体效果，用户还应该设计一个好的字典和停用词表等

第四章　CnX 索引子系统的实现

4.1　人机交互接口的实现

对用户来说，程序参数的设定应该非常方便，并且程序的设计也应该非常适应用户的习惯；对程序来说，程序应该能够实时向用户展现自己的运行状态并接受用户的命令等，这就是一个比较好的人机交互方式。所以在 CnX 索引子系统的实现过程中应该也要满足这些要求，在该系统中，直接将整个界面对象传入到各个线程中，这样各个线程就可以方便的获取并直接显示处理状态等。

4.1.1　参数的设置(输入)

在 CnX 索引子系统中，参数的输入主要分为三部分：

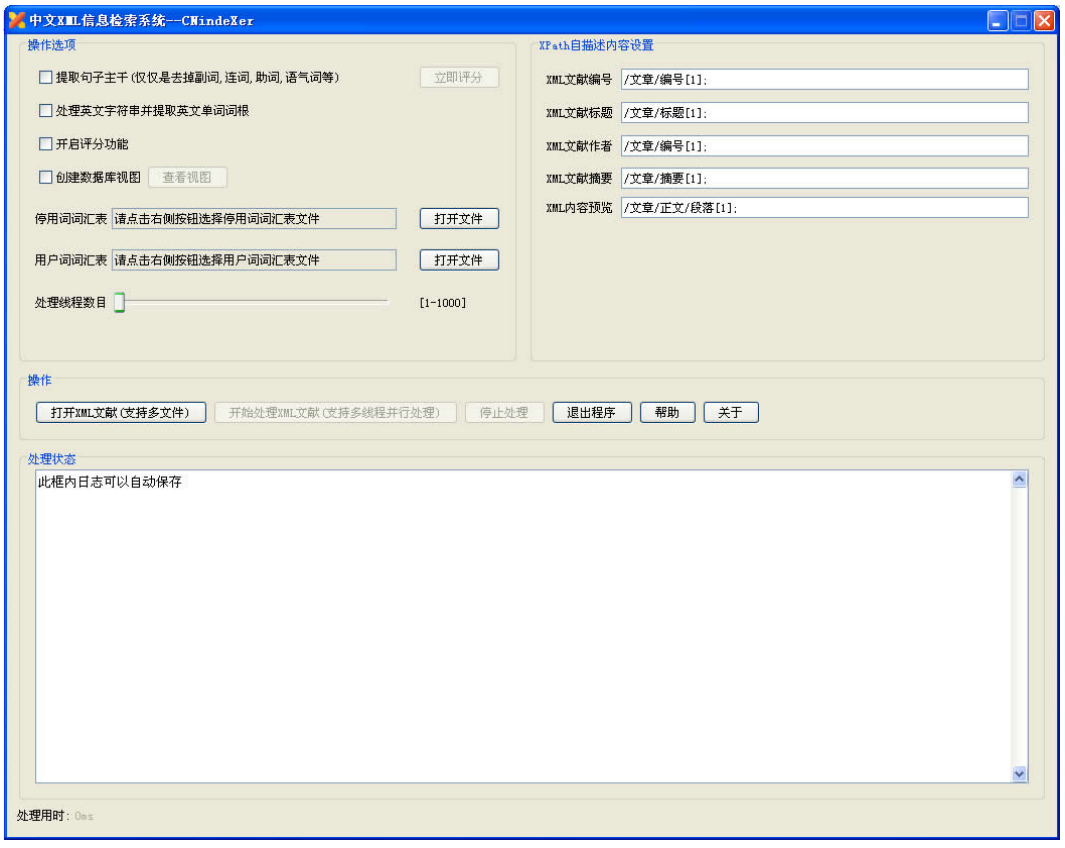

图 4-1　CnX 索引子系统主界面

- 操作选项参数：主要设置程序的初始化参数，包括是否提取语义、是否过滤停用词、是否即时评分和同时运行的最大线程数等
- XPath 自描述参数：此参数获取标准的 XPath 语句，提取 XML 文档的信息到 XMLCNDocument 文档对象中去。
- XML 文档：获取所有待处理的 XML 文档，并作为参数的形式传到各个

线程空间中去。

4.1.2 获取程序运行状态(输出)

如图 4-2 所示，运行状态的输出如日志动态文本框、进度条、处理时间以及一些按钮的变化等。

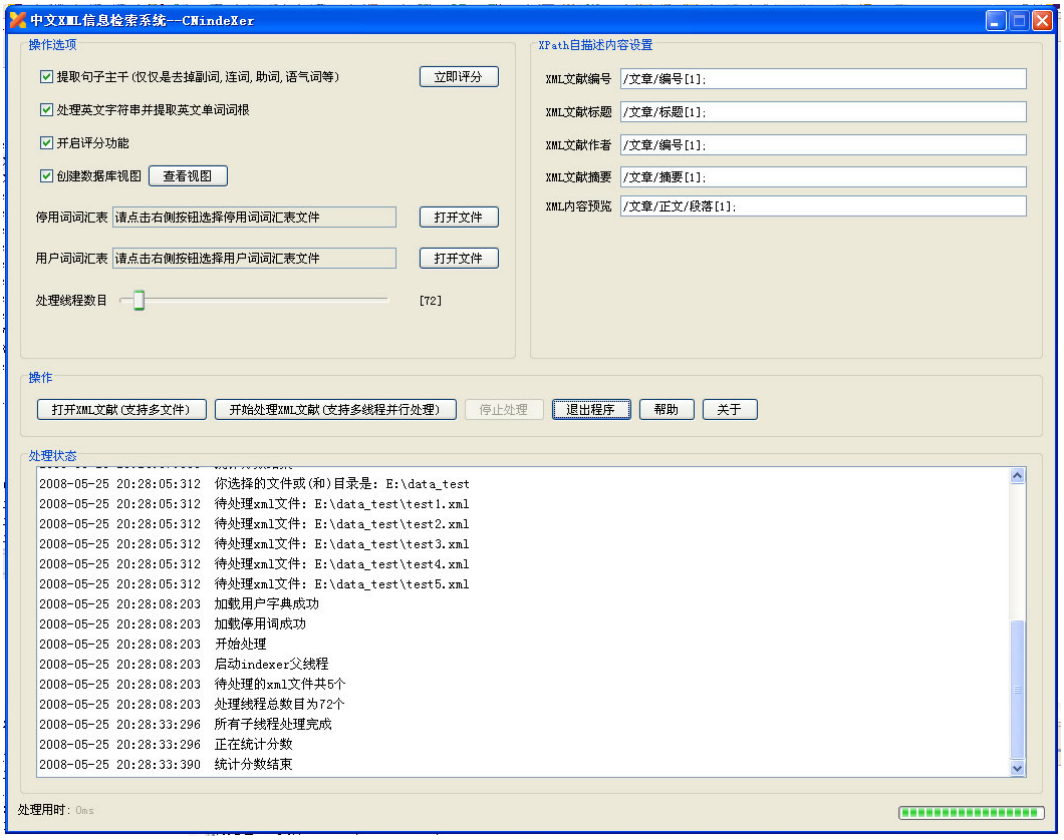

图 4-2 CnX 索引子系统处理状态显示

4.2 关键技术

CnX 索引子系统主要功能就是能够构建 XML 文献的倒排文档并能够根据 Okipa BM25 算法对索引项进行评分。系统的设计与实现过程中将会使用到一些关键的技术知识，比如核心的 XML 半结构化数据的倒排文档的构建技术、面向对象和 Java 技术、XML 技术、关系数据库、持久化技术，多线程分布式并行计算技术、中文分词技术、中文语义提取技术以及英文单词词根和停用词过滤技术等等。系统主体部分使用 Java 技术构建，然后各个模块之间的通信也是通过与 Java 相关的一些技术实现的。对于系统的设计为什么要使用这些技术也是有根据的，比如说，为什么要使用 Java 而不是用 C#呢？对于类似的这些问题都可以在下面的文章中找到相应的解答。以下将会详细介绍该系统运行环境以及系统实现过程中所采用的各种技术，并同时解答类似上述那样的问题。

4.2.1 面向对象的处理方法

- Dom4j

Dom4j 是 dom4j.org 出品的一个开源 XML 解析包[12]，它是一个易用的、开源的库，用于 XML，XPath 和 XSLT。它应用于 Java 平台，采用了 Java 集合框架并完全支持 DOM，SAX 和 JAXP。DOM4J 最大的特色是使用大量的接口，使得它使用起来变得非常灵活，目前使用 Dom4j 的项目已经越来越多。Dom4j 的主要接口都在 org.dom4j 包中，Dom4j 能够读取 XML 文档，返回一个 Document 对象，根据获得的 Document 对象获得该文档的根结点，并能使用迭代器或使用快速遍历方法对该文档进行遍历，还提供对 XPath 和 XSLT 的支持。最令人兴奋的是 Dom4j 对 Visitor 的支持，这样可以大大缩减代码量，并且清楚易懂。了解设计模式的人都知道，Visitor 是 GOF 设计模式之一。其主要原理就是两种类互相保有对方的引用，并且一种作为 Visitor 去访问许多 Visitable[13]。

- XPath 和 jaxen

XPath 是为 XML 定义的查询语言，它提供了在文档中选择结点子集的简单语法。通过 XPath 和指定类似于目录的路径（即名称）以及路径中的条件，可以检索元素集合。XPath 对 XSLT 和 XML 文档对象模型都很重要，Dom4j 中对 XPath 的支持是使用 jaxen 实现的。

- 面向对象(Object Oriented, OO)和 Java

面向对象[14]是当前计算机界关心的重点，它是上个世纪 90 年代软件开发方法的主流。面向对象的概念和应用已超越了程序设计和软件开发，扩展到很宽的范围。如数据库系统、交互式界面、应用结构、应用平台、分布式系统、网络管理结构、CAD 技术、人工智能等领域。

作为一种程序语言，Java 拥有许多重要的特征：简单的、面向对象的、网络的、解释的、健壮的、安全的、可移植的、高性能的，这些特性使得 Java 变得十分的流行。然而在 Java 的发展历史的各个不同的时间点上，Java 这个名词却有着不同的意义。

随着时间的推移，Java 这个名词不再只是表示一种程序语言，而是一种开发软件的平台，更进一步地说，Java 也是一种开发软件的标准与架构的统称。Java 语言只是 Java 蓝图中极小的一部分，更多的时候是在学习如何应用 Java 所提供的资源与各种标准，以开发出更好的软件。

- Java 本地接口(JNI)

Java Native Interface 技术允许在 Java 应用中集成 C 或 C++构建的模块。JNI 本身就是 Java 平台的一部分，而 Java 平台又是一个由 Java 虚拟机(JVM)和 Java 应用开发接口(API)组成的开发环境。在 CnX 索引子系统中，中文分

词和中文语义提取模块就是使用 C++开发的，需要在通过 Java 来访问这个模块，于是会使用到 JNI 技术。

当 Java 应用被部署到宿主环境中时(如 Windows 或者 Linux)，就必须要考虑到它将与宿主环境中的其它非 Java 语言编写的应用的协同工作，现在人们已经广泛的接受了 Java，但是从商业的角度上来说，Java 与类似 C 或者 C++这样的本地代码还要共存很长一段时间。JNI 是一个双向接口，它既允许 Java 应用调用本地库，又允许将一个 Java 虚拟机嵌入到一个本地应用中。图 4-3 展示了 JNI 在整个应用中的角色。

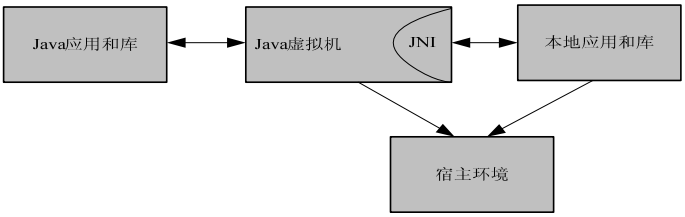

图 4-3　　JNI 的角色[15]

在实现中具体的做法是先编写带有 native 方法的 Java 类(图 4-4)，编译成字节码文件后，再使用 javah 工具生成对应的 C++头文件，然后实现 C++头文件中的方法，编译发布为 DLL 文件。Java 的程序只要在这个动态库加载后就可以直接使用带有 native 方法的 Java 类中定义的方法了。

```
package tjucs.util.ictclas;
public class FreeICTCLAS
{
    static{
        System.loadLibrary("FreeICTCLAS");
    }
    public native boolean FreeICTCLAS_ImportUserDic(String
userWordsFile);
    public native boolean FreeICTCLAS_Init(int OperateType, int
OutputFormat);
    public native String FreeICTCLAS_ParagraphProcess(String
sParagraph);
}
```

图 4-4　　带有 native 方法的 Java 类

- Java 反射机制

Java 提供的反射机制允许程序在运行时动态加载类、查看类信息、生成对象或操作生成的对象，很多的框架应用中都使用到了 Java 的反射机制，大大提高程序的代码质量和通用性。在 CnX 索引子系统中，也使用了 Java 的反射机制，通过动态加载类来大大优化代码的结构，提高系统的质量[9]。具体的例子位于包 tjucs.indeXer.indexer 中的类 XMLCNProcessor 的方法 getSelfIntroInfo()。

■ Java 多线程

Java 在语言级别上支持管程，这样就可以比较简单地解决 RPC 问题。但是在 CnX 索引子系统中，却不是使用这样的方法来解决线程间的通讯问题的，因为涉及到资源共享的内容比较少，就可以通过锁的机制直接使用共享内存的方式来实现。当今处理器资源丰富，所以需要尽可能的利用剩余的 CPU 资源，提高程序的执行效率。

4.2.2 数据存储

■ 关系数据库

上个世纪 70 年代，IBM 公司的 San Jose 研究实验室推出了一种新的关系数据模型的数据表达框架，到了上个世纪的 80 年代，关系数据库便巩固了它作为主导 DBMS 模式的地位，并且数据库系统已经被广泛的使用。在 CnX 索引子系统中使用了 Mysql 数据库来存储索引信息，通过论文后面的持久化技术的分析，该系统不一定要使用数据库或者可以很轻易的使用另一种关系数据库代替原先的 Mysql 数据库。

■ 持久化技术

Java 是面向对象的语言，因此对象十分重要。在企业级应用中开发，一个主要的部分就是维护和创建与数据库交互的应用。通常与数据库交互都是用 SQL 语句，这对熟悉面向对象技术的程序员来说是一个麻烦。因为无法用对象将数据表之间的关系描述清楚，或者需要花很大的力气在 SQL 语句上，才有可能用对象来描述数据表间的关系。通过开源社区的先驱们不断的思索和探讨，终于开发出了一个新的规范，即利用一组对象关系映射，并运用 Java 的反射机制将数据库中表结构以及表的关系完全反映到一组简单的 Java 对象中来，业务逻辑层只要对这组简单 Java 对象操作，就可以完成与数据库交互的工作，这就是所说的持久化技术[16]。

持久化技术有一些重要的名词：

- O/R Mapping：对象关系映射，利用一组简单的只有 get/set 方法的 Java 类来描述一张张数据库中的表，并通过 XML 来描述这些 Java 对象之间存在于数据库中的表之间的关系。
- POJO：简单的 Java 对象，可以理解为持久层对象，即该对象将与

数据库中表的一项对应。

- DAO：在持久化层的会话(session)中操作 POJO 的对象。

持久层框架大都属于开源框架，他们活跃于开源社区，颇具代表性的持久化框架包括 Hibernate 和 iBATIS，另外，SUN 官方发布的 JDO 和 EJB3.0 标准也属于持久化技术。在 CnX 索引子系统中使用了 Hibernate 持久层框架来实现索引数据的持久化存储。

4.2.3 语词处理

英文句子单词之间有天然的分隔符，于是不需要特别处理就可以很简单的得到最小语素——单词。CnX 索引子系统侧重于中文的处理，所以先处理中文，在处理中文的过程中发现英文语句，然后再去处理英文语句。在中文句子分词的过程中是使用 HMM 算法查字典来实现中文分词的，具体的分词效果至少达到 MS Office Word(在文本中按住 Ctrl，再按向左向右键即可看到它的分词功能)的分词效率和准确水平。

- 中英文混合分词
  - 中文语词处理

中文句子语词的提取主要还是基于语词词典来实现的，在中文语句分词的过程中，会去查阅分词模块指定的字典，得到一些词的信息，比如说词的词性，长度和使用频率等。同时分词词典还应该支持用户自定义的词库，提高人的干预度，从而可以提高分词的准确性和灵活性。对于用户字典，可以是一个简单的文本文件，其结构只是要求文本文件的一行为一个词，而且这个词的汉字数目不能超过 24 个。构建的字典(包括用户字典)结构如图 4-5 所示。

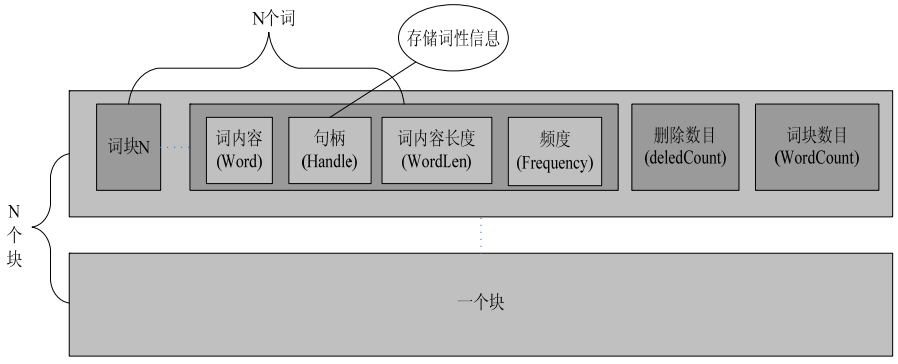

图 4-5　　字典结构[17]

对于根据字典提取词性，需要在句柄中构建相应的词性信息，词性信息含义如表 4-1 所示。

表 4-1　词性信息代表意义

后缀名	代表含义	举例
/w	标点符号	，；。！
/n	名词	茶杯
/m	数词	一个
/a	形容词	漂亮
/u	助词	的
/nx	英文字符串	Tianjin University
/d	副词	高兴地
/p	介词	到
/z	状态词	活生生
/t	时间词	2008 年
/f	方位词	东方
/r	代词	你
/e	叹词	哎
/y	语气词	呢
/c	连词	在
/l	用户词	用户字典中的词

- 英文单词词根提取

英文单词词根的提取主要是使用了 snowball 的 English-stemming 包，由于英文单词的词性的规则变化(少数是不规则的)，可以根据英文单词的前缀或者后缀的变化规律来处理英文单词，得到单词的词根(也不一定是词根，但这是无关紧要的)，通过处理英文单词，可以得到一个所有词根的合集(无冗余)，实现了索引的压缩。

- 停用词过滤

需要用户首先提供一个自定义而且较为合理的停用词表，这个表的结构可以用一个简单的文本来表示，要求一行一个停用词。在 CnX 子系统的实现过程中，首先要加载这些通用词，将这些停用词加载到一个哈希容器中，这个哈希容器中的所有停用词都不会有重复，而且在哈希容器中，可以对中英文词进行排序，中文词首字的拼音字母和英文词的首字母都可以按照英文的 alpha 字母表来排序，于是这样就可以使用分枝裁剪的算法提高效率。

4.3　功能模块的实现

4.3.1　构建倒排文档模块的实现

在构建倒排文档的过程中，最主要就是构建 XML 映射树和处理中英文语句语义，下面将一一介绍。

- XML 映射树的实现

XML 映射树就是把真实存在的 XML 文件读入内存，然后在这个 XML 文件

的基础之上构建一个合理的树结构，它的实现分为三个步骤：

- 构建基本的映射树框架

使用 Dom4j 包，首先读入 XML 文档，紧接着获得 XML 文档的根节点，此根结点也将作为映射树根结点的一部分。采用深度遍历的方式，以先访问结点的子结点再访问该结点属性的顺序，构建一颗映射树。例如，图 3-3 所示的 XML 文档映射成树以后应该是图 3-5 的样子。

- 前序遍历映射框架更新 bucketId、preId 和 elementId

在构建完成映射树以后，这部分的操作将会是非常的简单，考虑到 bucketId 与 oneIndex 相关，elementId 又与 documentId、preId 相关，而 oneindex 和 documentId 都已经在构建映射树框架的时候生成，于是 CNXMLElement 对象的这几个属性很适合在前序遍历中进行更新。

其中 elementId = (documented << 16) + preId ,这里要注意运算符的优先级。

- 后序遍历映射框架更新 features、cpts 和 postId

这个部分可以同前面的那个部分同步执行，同样，此部分的操作也十分简单。这里简要的介绍下这几个变量的含义，features 为一个结点的特征值，即每个结点以及其所有根结点所包含的语词信息，结构为一个哈希表，键值是这个结点里的任一个语词，值为这个语词在这个结点里出现的次数，由于需要先知道子结点的 features 信息才能构建父结点的 features 信息，所以此属性适合使用后序遍历的方式更新。相同的道理，cpts 表示该结点下所有语词的个数，也是需要先知道子结点的 cpts 才能更新父结点的 cpts，于是 CNXMLElement 对象的这两个属性可以在使用后序遍历的方式更新 postId 的时候进行更新。

■ 中英文语句语义处理模块的实现

- 中文分词以及中文语义的处理

中文分词使用 JNI 调用 C++编写的动态库来实现的。由于字典中存有中文词的词性和使用频率等信息，在中文分词的过程中在词的词尾加上表 4-1 中对应的信息，并将带有词性后缀的词以一个空格分隔组成新的句子。通过分析句子中的中文词后缀的信息提取语义，如过滤掉助词(后缀/u)，叹词(后缀/e)等。提取其中的后缀为/nx 的词，因为在字典中把英文字符串的词性标识定义为/nx，此部分在交由英文词根提取模块处理，最后返回结果。关于 freeICTCLAS.Dll 的构建，需要实现 C++中 char*到 Java 的 String 类型的转化，具体实现方法请参照函数 chartoJstring 和 jstringTochar 的实现。

- 英文词根提取

英文单词词根的提取也是基于字典的算法，不过由于英文单词的词性变化很有规律，可以在程序里直接构建字典数组，例如-ing,-ed,-ment，-tion，

-ly 等等，通过第一步中文语义的分析获得英文字符串，然后再根据上述算法构建只包含词根的英文语句，并以一个空格分隔，最后将结果传递到中文语义处理模块，再统一交由上层调用。

- 过滤停用词

此子模块在上述的两个子模块中都会被使用，这个部分很容易实现，将停用词存放在哈希容器中，以空格切割原句子得到单词，再从哈希容器中进行关键字匹配，从而过滤停用词。

4.3.2 全局语词对评分模块的实现

在前面的设计中已经提到，此部分使用 JDBC 执行 SQL 脚本的方式来实现。由于评分是一个全局的冗余计算的过程，所以在评分之前要先将所有的评分存储相关的都清空，使用类似 delete from element_frequencies 的语句即可。由于是使用 JDBC 调用一整段 SQL 脚本而不是通过 JDBC 执行存储过程，所以要在每个单独的 SQL 语句后以一个特定的字符标注 SQL 语句的结束，在本系统中使用一个“@”来标识，先以“@”分割 SQL 脚本，然后再依次执行 SQL 语句，最后得到索引的评分结果。下面介绍一下 SQL 脚本的具体实现[18]。

- 清空评分将会使用的表，如图 4-6 所示。

```
delete from element_frequencies@
delete from element_statistics@
delete from features_temp@
delete from element_maxscores@
delete from featurestagview@
delete from elementstagview@
delete from dfvalues@
```

图 4-6　　清空表的 SQL 代码

- 构建元素频率表和元素统计表，如图 4-7 所示。

```
insert into element_frequencies (select tag, term, count(pre) as ef from features group by tag, term)@

insert into element_statistics (select tag, count(did) as elements, avg(cpts) as avgcpts, 1.25 as k1, 0.75 as b from elements group by tag)@
```

图 4-7　　生成元素频率表和元素统计表的 SQL 代码

■ 构建特征临时表，这里实现了 Okapi BM25 算法，如图 4-8 所示。

```
insert into features_temp (
select
   f1.did,
   f1.tag,
   f1.term,
   f1.pre,
   f1.post,
   f1.lev,
   f1.bucketId,
   (log(2, 1 + (es.elements - ef.ef + 0.5)/(ef.ef + 0.5)) * (((es.k1 + 1)*f1.tf)/(es.k1*((1 - es.b) + (es.b * e1.cpts / es.avgcpts)) +
   f1.tf))) as localscore
from features f1, elements e1, element_frequencies ef, element_statistics es
where f1.did = e1.did and f1.tag = e1.tag and f1.pre = e1.pre  and f1.tag = ef.tag and f1.term = ef.term and f1.tag = es.tag
)@
```

图 4-8　　实现 Okapi BM25 算法 SQL 代码

■ 构建元素的最大分数表，如图 4-9 所示。

```
insert into element_maxscores(
 select tag, max(localscore) as maxelementscore from features_temp group by tag
)@
insert into element_maxscores(
   select '*' as tag, max(localscore) as maxelementscore from features_temp
)@
```

图 4-9　　生成元素的最大分数表的 SQL 代码

■ 统计 term-tag 词对在不同文档中的出现频率，如图 4-10 所示。

```
insert into dfvalues(select tag, term, count(distinct did) from featurestagview group by tag, term)@
```

图 4-10　　生成 term-tag 频率表的 SQL 代码

4.3.3 数据存储模块的实现

CnX 索引子系统数据存储使用了 Hibernate 持久化框架，由于索引的存储不依赖于具体的存储方式，所以实现部分还给出了其它实现方案的思考。

■ 使用(关系数据库)持久化技术

持久化技术将数据库的表映射为一个 Java 类，而表中的每一条记录将会在 Hibernate 的 session 中以一个该表对应的类的对象的形式存在，通过操作这个对象就可以达到操作表的目的。

持久化层的一些方法基本上采用了以往的项目中使用的方法，然后再单独实现自己的 DAO 类即可。

■ 存储模块的其他实现方案

倒排索引不依赖于数据库，可以直接以文件的形式存在文件系统上。由于 Oneindex 的全局性，所以可以直接将其内存对象存储下来，在需要的时候再加载就可以，而其它的索引则可以以块的形式存储到文件系统上，这样也可以实现索引的存储。不过对于索引的管理，就必须要重新设计一个模块来进行处理，而不是直接使用数据库去管理了。

4.4 系统实现的特点

CnX 的主要设计思想都源于 TopX 系统，但是 CnX 在自身的设计与实现过程中也根据自身的实际情况做了很多的改进和完善。CnX 主要的主要功能和优点都已经在前文中介绍过了，这里主要对比 TopX 的索引子系统分析一下 CnX 在构建过程中的显著特点。

TopX 索引子系统采用 Java 语言开发，但是它的实现过程中却没有用到面向对象的特点，它的各个系统之间的耦合度高而内聚度低，很不适合代码的二次应用。在其实现过程具有相同结构的代码重复出现多次，大大增加了代码的复杂度，TopX 的这些的不足在 CnX 中都得到了好的解决。

TopX 在构建和更新内存映射树的时候算法的实现逻辑比较混乱，让人难以琢磨，CnX 则根据对象属性的特征，将属性进行合理的分类，再分别以直观的前序和后续的树遍历方式来实现树的更新。

TopX 大量使用了 Oracle 数据库的特性，增加了跨平台移植的难度，CnX 则采用数据存储持久化技术，大大降低了对某种特定数据库的依赖等。

第五章　　结论

近年来，在 XML 数据集上进行信息检索变得十分流行。本文没有从全局上实现整个 XML 信息检索系统，只是实现了 XML 信息检索系统的索引子系统部分，但它为后面的研究工作作了必要的准备。

由于 CnX 是一个中文 XML 信息检索系统，它的索引子系统就需要对中文 XML 文档进行构建索引的相关工作。由于中文语句结构特点不同于西欧语句，于是需要针对中文做一些必要的处理，这是 CnX 索引子系统的特色之处。

论文在构建 CnX 索引子系统应用的过程中，做了以下工作：

- 运用 XML 技术：该系统是针对某一个或某几个 XML 文献集的检索系统，同一个文献集中的 XML 文档都会有相同的结构，然后使用 XPath 读取 XML 文档的某个节点或属性的具体信息，针对索引的数据模型调整 XML 文档结构等。
- 实现中文分词和英文词根提取：CnX 索引子系统的中文分词技术是基于 ictclas 的 free 版本实现的。在中文分词之前先加载字典，根据一些基本的标点符号进行断句，然后再分别处理每个小的部分。在处理每个小的部分过程中，先查询字典，找到已经在字典中存在的词，生成一个词组的邻接表，最后求出最优分词路径。同时，在 CnX 索引子系统中，可以添加用户字典，增加了人工干预的程度，提高了分词的灵活性和准确率。针对中英文混合的语句，该系统先是处理中文语句，并在处理中文语句的过程中发现英文语句，然后处理英文语句，最后再统一输出结果。这就是 CnX 索引子系统的特色之处。
- 面向对象的程序设计：程序主体部分使用 Java 语言编写，中文分词子模块使用 C++编写，在整个程序中所有的单个 XML 文档都会以一个对象的形式存在，对于系统性能瓶颈的地方，采用 C++编写，然后使用 JNI 技术进行调用，倒排索引采用面向对象的持久化存储。通过面向对象的程序设计，提高程序源码的可读性和重用性等。

CnX 索引子系统目前还只是一个简单的原型，还有很多的地方需要完善，比如说，CnX 支持的 XML 格式只是 XML 标准的一个简单的子集，对于 Xlink 和 XSL 等等暂时都还没有支持，这些都需要在以后的工作中进行改进和完善。另外，今后还应该开展上层查询算法的研究，逐步完善 CnX 信息检索系统。

参考文献

[1]Ricardo Baeza-Yates，Berthier Ribeiro-Neto 等著，王知津，贾福新，郑红军等译．现代信息检索[M]．北京：机械工业出版社，2006．1—96．

[2]World Wide Web consortium. XML Path Language (XPath) Version 1.0[EB/OL]. http://www.w3.org/TR/1999/REC-xpath-19991116,W3C Recommendation, 16 November 1999.

[3]Chuck White，Liam Quin，Linda Burman 著，周生炳，宋浩，肖伟等译．XML 从入门到精通(黄金版)[M]．北京：电子工业出版社，2002．5—12，653—661．

[4]Martin Theobald, Holger Bast, Debapriyo Majumdar *et al*. TopX:efficient and verdatile top-k query processing for semistructured data[J]. Max-Planck Institute for Informatics, VLDB Jounal(2008): 81—92.

[5]OrientX [EB/OL]. http://idke.ruc.edu.cn/OrientX/.

[6]中科院计算所汉语词法分析系统 ICTCLAS[EB/OL]. http://www.nlp.org.cn/project/project.php?proj_id=6．

[7]Martin Theobald. TopX : efficient and verdatile top-k query processing for semistructured data[D]. Saarbrücken : Max-Planck Institute for Informatics, 2006.

[8]林信良编著．Java 学习笔记[M]．北京：清华大学出版社，2007．239—253，329—410，458—486．

[9]Ellis Horowitz，Sartaj Sahni，Suan Anderson-Freed 著，李建中，张岩，李志军译．数据结构(C 语言版)[M]．北京：机械工业出版社，1997．118—200．

[10]Udi Manber 著，黄林鹏，谢瑾奎，陆首博等译．算法引论[M]．北京：电子工业出版社，2005．44—61，133—139．

[11]Russ Miles，Kim Hamilton 著，汪青青译．UML 2.0[M]．北京：清华大学出版社，2007．1—109．

[12]dom4j cookbook[EB/OL]. http://www.dom4j.org/cookbook.html.

[13]Steven John Metsker 著，龚波，冯军，程群梅等译．设计模式 Java 手册[M]．北京：机械工业出版社，2006．229—238．

[14]面向对象语言[EB/OL]．http://www.itisedu.com/phrase/200603010948085.html, 2008.

[15]Sheng Liang. The Java™ Native Interface Programmer’s Guide and Specification[M]. ADDISON-WESLEY, 1999. 1—59.

[16]Hibernate 中文帮助文档[EB/OL]．http://www.redsaga.com/hibernate_book.html，2006．

[17]ictclas4j 中文词系统文档[EB/OL]．http://code.google.com/p/ictclas4j/, 2008.

[18]Raghu Ramakrishman，Johannes Gehrke 著，周立柱，张志强，李超等译．数据库管理系统原理与设计[M]．北京：清华大学出版社，2004．681—704．

致　　谢

本论文的完成得益于很多人的指导，关心和帮助。

首先感谢张坤龙老师，给予我论文各个方面极大的指导和帮助，他教会了我做科研的方法，并且细心指导我论文的大纲结构和格式要求，我们经常在一起探讨人生的真谛，感悟学问的灵魂。他认真的工作态度和严谨的科研作风深深地影响了我，使我终身受益。

感谢信息与网络中心的老师们对我的指导和关心，给了我许多宝贵的意见，还提供了无比优越的工作和学习环境，让我能更好地学习知识，并完成毕业设计任务。

感谢吴宗远同学，我们一起阅读论文，一起讨论问题，一起研究问题，在阅读英文论文的时候他给了我很大的帮助。

感谢曹玮同学，我们经常谈着人生的理想，我们还时常交流技术问题，他给了我很大的支持和帮助。

感谢和我朝夕相处的室友，特别是与我同年同月同日生的孙恺，若不是天天都能见到他带着酒窝的灿烂笑容，我恐怕不能这么高效地完成自己的毕业设计。

最后衷心感谢我的家人和朋友，是他们对我的关心、支持和帮助才使我能够安心地做毕设，才能够顺利地完成本论文。