

JZSearch 精准搜索引擎技术白皮书



自然语言处理与信息检索共享平台
Natural Language Processing & Information Retrieval Sharing Platform

<http://www.nlpir.org/>

2015-6

For the latest information about JZSearch, please visit <Http://www.nlpir.org/>

Document Information

Document ID	JZSearch-WHITEPAPER	Version	V1.0
Security level	Public 公开	Status	Creation and first draft for comment
Author	张华平	Date	May 31, 2015
Publisher	/	Approved by	

Version History

Note: The first version is "v0.1". Each subsequent version will add 0.1 to the exiting version. The version number should be updated only when there are significant changes, for example, changes made to reflect reviews. The first figure in the version 1.x denotes current review status by. 1. x denotes review process has passed round 1 etc .Anyone who create, review or modify the document should describe his action.

Version	Author/Reviewer	Date	Description
V1.0	Kevin Zhang	2011-8-21	first complete draft for comment.
V1.1	Kevin Zhang	2011-9-15	增加了缓存机制的说明
V2.0	Kevin Zhang	2011-12-5	
V2.5	Kevin Zhang	2012-1-5	增加可视化界面部分，及案例篇
V2.6	Kevin Zhang	2012-5-24	增加多编码支持，增加情感扫描识别，新增1.5；新增3.1.12，修改4.2.1 修改4.3.1
V2.7	Kevin Zhang	2012-5-28	增加了Complex运算符号
V2.7	Kevin Zhang	2012-5-28	语法中：列出所有的索引内容
V2.8	Kevin Zhang	2012-6-8	增加同义词联想功能
V2.9	Kevin Zhang	2013-9-28	增加搜索聚类、比较功能，见3.1.17-3.1.19
V3.0		2015-5-31	增加了Unique去重语句
V3.1		2015-6-21	增加了JZOR条件或语句

目录

JZSearch 精准搜索引擎技术白皮书.....	1
目录	3
一、 入门篇：了解基本知识.....	6
1.1、JZSearch 精准搜索引擎介绍.....	6
1.2、JZSearch 精准搜索引擎的优势与特色	9
1.3、JZSearch 精准搜索引擎性能测试.....	12
1.4、字段类型定义.....	12
1.5、精准搜索排序策略.....	14
1.6、支持的字符编码.....	14
二、 通用篇：五步配置搭建搜索服务.....	14
2.1、采用可视化界面配置（适用于 Windows,安装包比较大，需要可另外索取下载）15	
2.1.1、运行 star.Bat 出现初始界面，配置数据库信息.....	15
2.1.2、选择要建立索引的表，点击“执行”、“创建索引”	16
2.1.3、弹出如下界面：配置好字段信息:点“创建索引文件”	16
2.1.4:选择索引服务，然后选择项目，点击建立索引：	17
2.1.5：索引建立成功之后，即可开启索引服务，测试索引：	18
2.2、采用手工配置（适用于各种操作系统）	18
2.2.1 第一步：配置数据库读取参数.....	19
2.2.2 第二步：数据库字段信息列表导入.....	20
2.2.3 第三步：建立字段信息文件	20
2.2.4 第四步：自动建立索引	20
2.2.5 第五步：启动搜索服务.....	21
2.2.6 第六步：启动客户端服务.....	21
2.2.7 其他脚本.....	21
2.2.8 Linux 环境	21
三、 进阶篇：了解内核.....	22
3.1、JZSearch 检索语法说明	22
3.1.1 搜索与运算符 AND	22
3.1.2 搜索或运算符 OR	22
3.1.3 搜索非运算符 NOT.....	23
3.1.4 搜索邻近运算符 NEAR.....	23
3.1.5 搜索智能模糊运算符 FUZZY	23
3.1.6 搜索范围运算符 RANG	23
3.1.7 搜索最小值运算符 MIN	24
3.1.8 搜索最大值运算符 MAX	24
3.1.9 搜索前缀运算符 PREFIX	24
3.1.10 搜索精准运算符 PRECISION	25
3.1.11 搜索排序运算符 SORT.....	25
3.1.12 复合运算符 COMPLEX.....	26
3.1.13 列出所有索引的结果内容 LISTALL	26
3.1.14 正负面情感搜索.....	26
3.1.15 删除显示运算符 Deleted	27

3.1.16 两表关联搜索语法.....	27
3.1.17 搜索聚类运算符 Cluster	28
3.1.18 分组运算符 Groupby.....	29
3.1.19 搜索结果比较运算符 Compare Between With.....	29
3.1.20 搜索结果自动去重处理 Unique	30
3.1.21 搜索结果条件或处理 JZOR	30
3.2 JZSearch 搜索后台服务系统搭建.....	31
3.3 JZSearch 客户端搭建与管理指南.....	32
3.3.1 客户端管理命令语法.....	32
3.3.2 命令行方式.....	32
3.3.3 C 语言 API 方式.....	33
3.3.4 JAVA 语言调用搜索客户端.....	34
3.3.5 调用搜索服务的协议说明.....	34
四、 高级篇：API 定制开发	36
4.1 字段定义接口.....	36
4.1.1 JZIndexer_FieldAdd 添加一个字段.....	36
4.1.2 JZIndexer_FieldSave 保存已经设置的字段信息.....	37
4.1.3 JZIndexer_FieldLoad 读取已经设置的字段信息文件	38
4.2 索引接口.....	38
4.2.1 JZIndexer_Init 精准索引器初始化	38
4.2.2 JZIndexer_Exit 精准索引器系统退出	39
4.2.3 CJZIndexer 精准索引器类.....	40
4.2.3.1 CJZIndexer::CJZIndexer 精准索引器类构造函数	41
4.2.3.2 CJZIndexer::MemIndexing 精准索引器类内存索引函数	42
4.2.3.3 CJZIndexer::FileIndexing 精准索引器类文件索引函数.....	43
4.2.3.4 CJZIndexer::BigFileIndexing 精准索引器类大文件索引函数.....	45
4.2.3.5 CJZIndexer::IdIndexing 精准索引器类 ID 索引函数	46
4.2.3.6 CJZIndexer::IntIndexing 精准索引器类整型索引函数	47
4.2.3.7 CJZIndexer::LongIndexing 精准索引器类 64 位长整型索引函数	48
4.2.3.8 CJZIndexer::FloatIndexing 精准索引器类浮点型索引函数.....	50
4.2.3.9 CJZIndexer::AddDoc 精准索引器类文档添加函数	51
4.2.3.10 CJZIndexer::Save 精准索引器类保存函数	52
4.2.3.11 CJZIndexer::Merge 精准索引器类索引合并函数.....	53
4.2.3.12 CJZIndexer::Export 精准索引器类索引导出函数	54
4.3 检索接口.....	55
4.3.1 JZSearch_Init 精准搜索器初始化	55
4.3.2 JZSearch_Exit 精准搜索器系统退出	56
4.3.3 JZSearch_Reload 精准搜索器系统增量加载.....	57
4.3.4 JZSearch_Export 精准搜索器系统索引内容导出函数	58
4.3.5 JZSearch_Merge 精准搜索器系统索引归并优化函数.....	59
4.3.6 搜索结果的数据记录结构.....	59
4.3.7 CJZSearcher 精准搜索器类	60
4.3.7.1 CJZSearcher::CJZSearcher 精准索引器类构造函数.....	61
4.3.7.2 CJZSearcher::Search 精准索引器类搜索函数	62

4.3.7.3 CJZSearcher::Search 精准索引器类搜索函数	63
4.3.7.4 CJZSearcher::DocDelete 精准索引器类索引文档删除函数	64
4.4 利用 JZSearch 开发程序，搭建搜索引擎服务指南	65
4.5 索引数据辅助分析	66
五、 案例篇	66
5.1 中国邮政集团名址信息中心首页的邮址垂直搜索	66
5.2 河北标准化研究院的标准搜索	66
5.3 中国对外承包工程商会的知识搜索门户	67
5.4 富基融通的商品比价搜索	68
5.5 WBK 微博人物搜索	69
六、 问答篇：FAQ 及小技巧	70
6.1：环境问题	70
6.1.1 JZSearch 支持 Linux 吗？	70
6.1.2 Linux 环境使用太不方便，有什么技巧？	70
6.2：索引问题	70
6.2.1 在 Windows 下，针对 MySQL 数据库的索引老是创建不成功，访问不了数据库，为什么？	70
6.2.2 采用 bigfile 字段，为什么？	70
6.2.3 数据库增删改如何适应，为什么？	70
6.3：搜索问题	70
6.3.1 老是搜索不到结果或者搜索到的结果老是不变的，怎么回事？	70
七、 作者篇	71

一、入门篇：了解基本知识

1.1、JZSearch 精准搜索引擎介绍

JZSearch 精准搜索引擎由北理工副教授张华平博士精心设计，具有专业精准、高扩展性和高通用性的特点。可支持文本、数字、日期、字符串等各种数据类型的高效索引，支持丰富的查询语言和查询类型，支持少数民族语言的搜索。目前已经应用于中国邮政搜索引擎、河北省标准搜索引擎、富基融通（纳斯达克上市公司：EFUT）商品搜索。

同时，全文搜索中间件通过可视化界面，可以快速地配置相关参数，启动搜索服务，并提供测试程序，可以无缝地与现有数据库系统融合，实现全文搜索与相关的数据库管理应用系统。

其主要特性包括：

- ✧ 可以按照任意指定字段的排序，支持指定字段的搜索，也可以搜索多个字段，以及复杂表达式的综合搜索；
- ✧ 支持精确匹配以及模糊匹配，默认为精确匹配，忽略字母大小写进行模糊匹配；
- ✧ 内嵌正负面情感等极性分析，也可以支持类别搜索；
- ✧ 语义联想搜索：如搜索“马铃薯”可以同时返回“土豆”的内容，搜索“北京市”可以返回“北京”或者“首都”的内容；语义联系词表用户可以根据业务需要进行定制；
- ✧ 支持增量索引：系统可以在搜索服务不停的前提下，继续索引新的

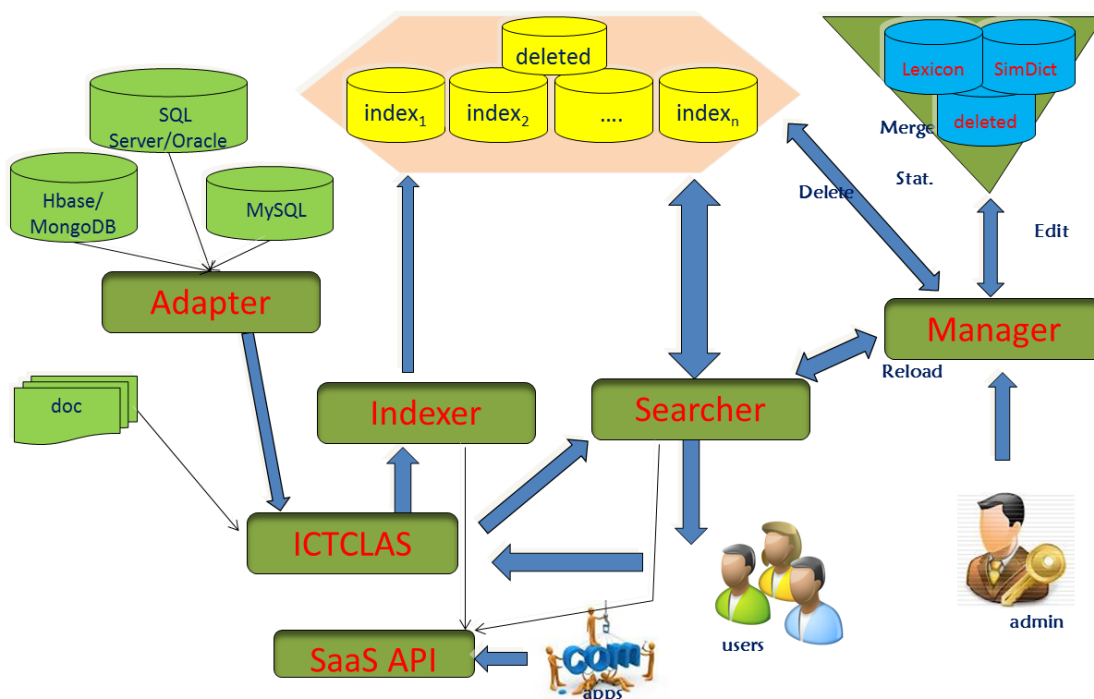
数据，索引完成后，可以搜索新的数据；

- ✧ 自动备份与恢复机制，在建立索引和自动优化之前，系统会将已有的索引文件自动备份；在当前索引文件被破坏无法搜索的前提下，系统将自动恢复上次搜索正常的备份文件；
- ✧ 自动缓存机制：系统自动保存最近常用的搜索条件与结果，再次搜索时将直接推送搜索结果内容，可以将搜索响应速度提升 30% 以上；缓存会随着新的索引数据自动更新，不存在缓存延迟问题；
- ✧ 自动优化机制：在系统索引碎片较多时，系统会自动优化归并；
- ✧ 实现的是多线程搜索服务；
- ✧ 兼容当前所有厂商的数据库系统，其中 SQL Server, Oracle, MySQL, DB2 等。
- ✧ 支持 Windows/Linux/FreeBSD 等操作系统，支持 C/C++/C#/Java 二次开发。

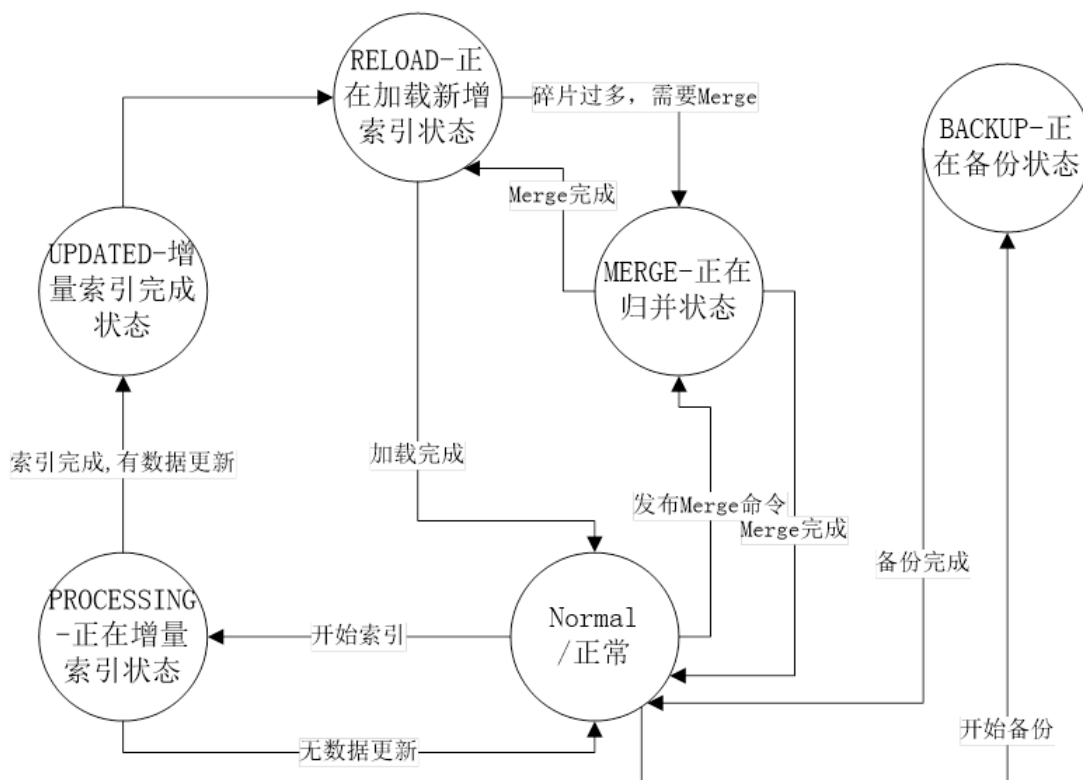
JZSearch 大数据精准搜索引擎的技术架构如下图所示：

JZSearch 大数据精准搜索引擎的技术架构如下图所示：

JZSearch Big Data Search Engine Architecture



JZSearch 大数据精准搜索引擎的状态转移图如下图所示：



1.2、JZSearch 精准搜索引擎的优势与特色

下面，我们结合已经大规模公开应用的中国标准图书馆搜索（<http://121.28.48.213:9999/byweb/search!list.action>）为例，进行说明。

示例一：搜索“马铃薯”，系统会自动搜索出标准数据库里面的标题以及各类信息，左边会自动对搜索结果进行统计分析。

The screenshot shows the JZSearch Standard Library search interface. The sidebar on the left lists various standard categories, with a red box highlighting the '标准分类' (Standard Classification) section. The main search area shows the keyword '马铃薯' (Potato) and a search button. Below the search bar, there is a table of search results. The table has columns for '标准号' (Standard Number), '名称' (Name), and '中标分类名称' (Standard Classification Name). The results are filtered by '国内标准' (Domestic Standard). A red text overlay on the table states: '对搜索结果自动分类统计' (Automatic classification and statistics for search results) and '自动搜索数据库里面的内容，数据库更新10秒内，会直接反映在搜索结果里。' (Automatic search of database content, database updated within 10 seconds, will be directly reflected in search results). The table lists several standards related to potato cultivation and processing, such as '旱地膜覆盖栽培技术规程' (Technical specifications for dryland film-covered cultivation) and '马铃薯有机栽培技术规程' (Technical specifications for organic potato cultivation).

标准号	名称	中标分类名称
DB13/T 396.7-1999	旱地膜覆盖栽培技术规程	种籽与育种
DB13/T 398.7-1999	旱地膜覆盖栽培技术规程	农林技术
DB13/T 865-2007	马铃薯有机栽培技术规程	瓜果、蔬菜种植与产品
DB1302/T 156-2001	无公害马铃薯生产技术规程	瓜果、蔬菜种植与产品
DB1304/T 071-2001	无公害农产品生产技术规程 马铃薯	农林技术
DB1304/T 102-2001	马铃薯脱毒种薯繁育技术规程	农林技术
DB1304/T 136-2004	棉花与土豆(马铃薯)间套复种生产技术规程	瓜果、蔬菜种植与产品
DB1306/T 11-2005	无公害马铃薯生产技术规程	瓜果、蔬菜种植与产品
DB1308/T 007-1999	马铃薯茎尖脱毒技术规程	豆类、薯类作物与产品
DB1308/T 008-1999	脱毒马铃薯微型种薯生产技术规程	豆类、薯类作物与产品
DB1308/T 009-1999	脱毒马铃薯基础种薯生产技术规程	豆类、薯类作物与产品
DB1308/T 010-1999	脱毒马铃薯合格种薯生产技术规程	豆类、薯类作物与产品
DB1308/T 011-1999	马铃薯脱毒种薯质量标准	豆类、薯类作物与产品
DB1308/T 012-1999	马铃薯商品薯(块茎)质量标准	豆类、薯类作物与产品
DB1308/T 088-2005	食用马铃薯淀粉	豆类、薯类作物与产品

图 1. 语义搜索功能示例

示例二：高级搜索，JZSearch 精准搜索引擎可以针对任意类型的字段进行各类综合搜索。复杂的搜索响应时间均在 100 毫秒左右。

标准图书馆 www.bzsb.info

加入收藏 设为首页 帮助 关于我们

网站 | 登录 | 注册 | 返回首页

高级检索

标准号: gb

标准名称: 土豆

主题词:

适用范围:

附录:

发布日期: 从 到 日期格式(年-月-日): ####-##-##

实施日期: 从 到 日期格式(年-月-日): ####-##-##

废止日期: 从 到 日期格式(年-月-日): ####-##-##

引用标准:

采用标准:

检索范围: ☒ 国内标准 ☐ 国际标准

查 询

系统可以针对给定的任意类型字段(文本或者数据)进行搜索

示例三：搜索“土豆”，系统自动搜索出“马铃薯”等信息

标准图书馆 www.bzsb.info

加入收藏 设为首页 帮助 关于我们

网站 | 登录 | 注册 | 返回首页

标准分类

- 农药管理与使用方法(6)
- 畜禽饲料与添加剂(3)
- 农机具(3)
- 植物检疫、病虫害防治(2)
- 种籽与育种(2)
- 标志、包装、运输、贮存(2)
- 食品加工与制品综合(2)
- 粮食、饲料作物综合(1)
- 豆类、薯类作物与产品(1)

标准号 gb 关键词: 土豆 检索 ☒ 按国内标准 ☐ 按国外标准

分类检索 高级检索

标准号 名称 中图分类号

GB 10395.16-2010	农林机械 安全 第16部分:马铃薯收获机	农机具
GB 18133-2000	马铃薯病毒种薯	豆类、薯类作物与产品
GB 4406-1984	种薯	种籽与育种
GB 7331-2003	马铃薯种薯产地检疫规程	种籽与育种
GB 7413-2009	甘薯种薯产地检疫规程	植物检疫、病虫害防治
GB/T 15683-2008	大米 直链淀粉含量的测定	粮食、饲料作物综合
GB/T 17980.133-2004	农药 田间药效试验准则(二) 第133部分:马铃薯脱叶干燥剂试验	农药管理与使用方法
GB/T 17980.137-2004	农药田间药效试验准则(二) 第137部分:马铃薯和芽病试验	农药管理与使用方法
GB/T 17980.15-2000	农药田间药效试验准则(一) 杀虫剂防治马铃薯等作物蚜虫	农药管理与使用方法
GB/T 17980.34-2000	农药田间药效试验准则(一) 杀菌剂防治马铃薯晚疫病	农药管理与使用方法
GB/T 17980.37-2000	农药田间药效试验准则(一) 杀线虫剂防治胞囊线虫病	农药管理与使用方法
GB/T 17980.52-2000	农药田间药效试验准则(一) 除草剂防治马铃薯地杂草	农药管理与使用方法
GB/T 20194-2006	饲料中淀粉含量的测定 旋光法	畜禽饲料与添加剂
GB/T 23620-2009	马铃薯甲虫疫情监测规程	植物检疫、病虫害防治
GB/T 25417-2010	马铃薯种植机 技术条件	农机具

示例四：搜索“GB 1003”，系统不会给出 GB 10030 等不准确的结果。

实现了精准的匹配与理解。

标准图书馆 www.bzsb.info

加入收藏 设为首页 帮助 关于我们

网站 | 登录 | 注册 | 返回首页

标准分类

- 低压电器综合(1)

标准号 GB 1003 关键词: 检索 ☒ 按国内标准 ☐ 按国外标准

分类检索 高级检索

标准号 名称 中图分类号

GB 1003-2008	家用和类似用途三相插头插座 型式、基本参数和尺寸	低压电器综合
--------------	--------------------------	--------

共 15 条记录, 合 1 页

河北省标准化研究院标准文献中心
技术支持: 灵致中科(北京)软件有限公司
京ICP备 05014994号
Copyright © 2004-2012

精准匹配GB1003, 而不输出右下方的结果

GB 10030-2006	团头鲂鱼苗、鱼种
GB 1003-2008	家用和类似用途三相插头插座 型式、基本参数和尺寸
GB 10035-2006	气囊式体外反搏装置

与传统的开源搜索引擎 Lucence 与 Sphinx 系统, 以及 TRS 等已有的搜索厂商, 通过以上的例子, 我们不难看出 JZSearch 精准搜索引擎的优势与特色在于以下五点:

1. **语义精准搜索:** 系统不是简单的关键词匹配, 会自动根据语义知识进行联想, 搜索用户真正需要的信息; 并能实现数字、字母的精准搜索, 而不是模糊匹配(搜索 100, 而不会给出 1001 等不当结果)。
2. **与业务无缝衔接:** JZSearch 精准搜索引擎能够兼容现有的所有数据库, 无需开发, 即可实现与现有业务数据库的无缝衔接, 实现任意字段的搜索分析, 为数据库提供毫秒级的搜索服务; 而新增或者修改的数据在 10 秒内, JZSearch 搜索引擎就能捕获并反映在搜索结果中。非常适合企业业务数据的垂直搜索过滤。
3. **更强搜索功能:** JZSearch 可以便利的搜索负面信息, 对搜索结果自动统计分析, 还可以实现指定文字范围内的搜索, 如: 搜索招标公告中的公司名称附件的法人信息, 而不是泛泛地搜索, 可以实现信息的精准定位。
4. **系统自主维护:** JZSearch 搜索引擎可以处理 TB 级别的大数据业务, 索引速度 10MB/s, 搜索响应 1 秒以内; 整个过程无需人工干预, 系统部署后, 自动索引、自动搜索、自动增量更新、自动优化、自动备份与恢复, 实现 7*24 小时在线运行。

5. 各类成功案例：目前，基于 JZSearch 公开运行的搜索引擎已经有 3 年多的历史，各类政府、企业及互联网的典型成功案例包括：中国邮政集团的邮政搜索引擎（所有数据 2 个亿，每年搜索次数超过 1 个亿）；中国证监会的舆情搜索服务（覆盖所有财经网站）；河北省标准化研究院的标准搜索引擎（同时在线服务 10 万家的企业）；纳斯达克上市公司富基融通的商品搜索引擎（服务中国 70% 的零售业）；缔元信网站统计数据搜索服务（每天新增数据 500GB）。

1.3、JZSearch 精准搜索引擎性能测试

项目	参数
测试机器配置	ThinkPad x220i 内存 2G， Intel Core i3-2310M CPU @2.10GHz 2.10GHZ
测试数据	NLPIR 新闻语料库（中文 3,686,479KB 英文 2,249,607KB）
使用内存	1GB
占用 CPU	25%
索引时间	1,115 秒
索引速度	5,323KB/s
搜索速度	<1 秒可控制在毫秒级别

注：测试在个人使用的笔记本电脑运行，在服务器端的速度可以超过 10MB/s。

1.4、字段类型定义

JZSearch 兼容当前基本的数据类型，如下表。

类型名	类型值	说明
FIELD_TYPE_TEXT	1	文本类型
FIELD_TYPE_INT	2	整型（32 位整型数据）
FIELD_TYPE_LONG	3	长整型（64 位整型数据）
FIELD_TYPE_DATETIME	4	日期时间型（目前等同于 FIELD_TYPE_INT）

FIELD_TYPE_FLOAT	5	浮点型
FIELD_TYPE_BIGTEXT	6	存储在大文件中

示例如下：

序号	数据库字段名称	索引名称	数据类型	是否需要索引	是否需要存储原文	是否需要通配索引	是否需要摘要并红显	备注信息
1	bigfilename@offset@length@e:\语料库资源\corpus\新闻语料库	content	bigtext	1	1	1	1	0
2	id	id	text	0	1	0	0	0
3	title	title	text	1	1	1	1	0
4	url	url	text	0	1	0	0	0
5	publishtime	publishtime	date	1	1	0	0	0
6	click	click	int	1	1	0	0	0

说明如下：

- 数据库字段名：**为数据库中的实际字段名称，必须一致，否则无法访问数据库；如不访问数据库，该字段可以不设置。
- 索引名：**为后续检索的字段名称，由用户自定义，必须和提交搜索的名称一致，否则解析错误，无法正确搜索；
- 数据类型：**支持数据库的常见格式，其中要求非字符型字段不用有索引，只用于过滤或者存储；参见数据字段类型表；
- 是否需要索引：**表示的是搜索系统在索引过程中是否保留原文，并在检索结果中是否出现；
- 是否需要存储原文：**存储在文件中，方便搜索结果显示该字段；
- 是否为通配符索引：**在不指定字段名称进行统配搜索（搜索 *）时，是否覆盖该字段。
- 是否需要摘要并红显：**在结果显示时，需要对该字段进行动态摘要，并红显。
- bigtext 类型：**主要解决大文件存储多个小文本的情况，适合于海量文本的处理，对应数据库的时候，需要依次按照顺序指定如下信息：bigfilename@offset@length@e:\语料库资源\corpus\新闻语料库 分别对应的是 大文件名，偏移量，文本内容长度 即指定的文件路径名，中间采用“@”链接。注：路径名最好是绝对路径。

1.5、精准搜索排序策略

精准搜索引擎提供四种排序方式：

类型名	类型值	说明
docid	1	按照 docid 顺序排序，即先建索引的文档优先
relevance	2	按照相关度排序
reverse_docid	3	按照 docid 逆序排序，即后建索引的文档优先
<field_name>descend/ascend	4	按照指定的数值型字段进行排序，同时可以指定从小到大，或者从大到小（默认为从大到小）

1.6、支持的字符编码

类型名	类型值	说明
INDEX_ENCODING_GBK	0	默认为 GBK 编码，其中只有简体字
INDEX_ENCODING_UTF8	1	UTF8 编码
INDEX_ENCODING_BIG5	2	BIG5 编码
INDEX_ENCODING_GBK_FANTAI	3	GBK 编码，但是简体字繁体字混杂，选择此项，系统将实现繁简体跨界搜索。

二、通用篇：五步配置搭建搜索服务

本篇介绍如何利用 JZSearch 精准搜索引擎工具包，不进行程序定制开发，搭建一套针对特定数据库的垂直搜索引擎服务。

2.1、采用可视化界面配置(适用于 Windows,安装包比较大,需要可另外索取下载)

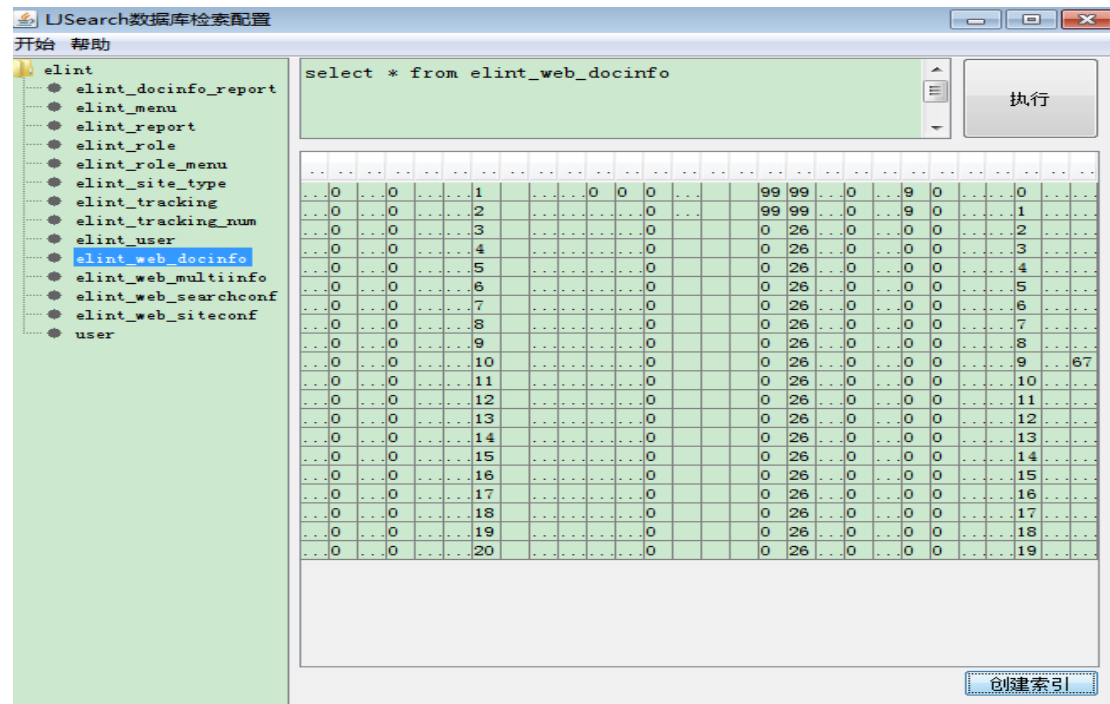
2.1.1、运行 star.Bat 出现初始界面，配置数据库信息



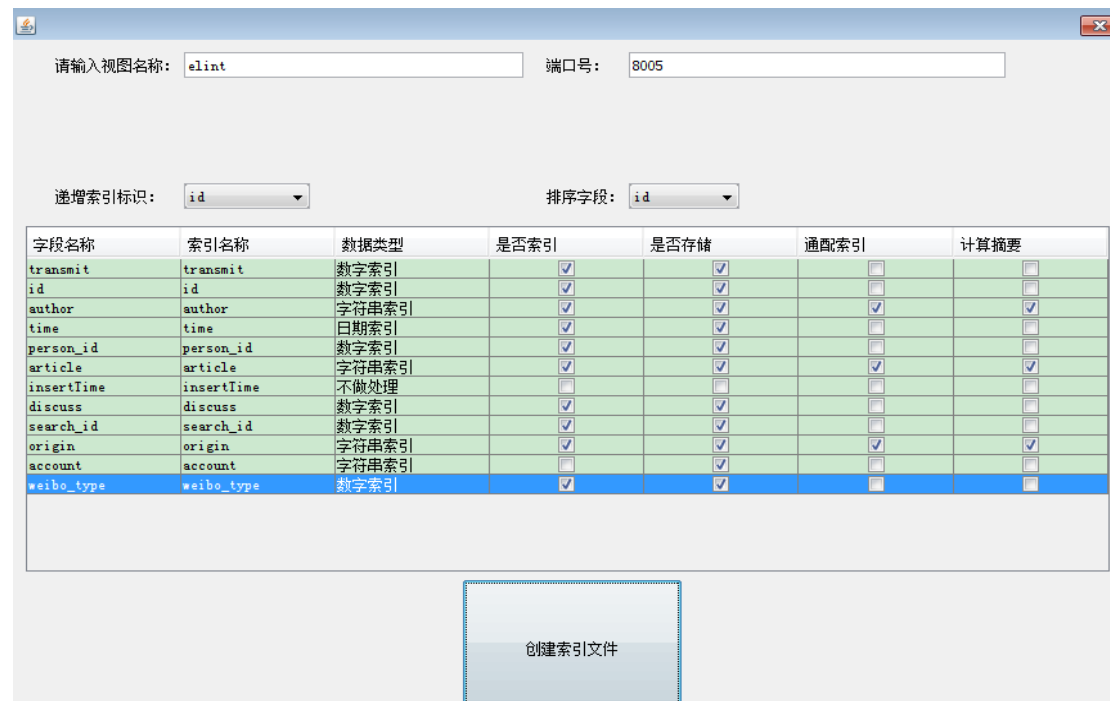
The screenshot shows a Windows-style configuration window for JZSearch. It contains the following fields and controls:

- 主机IP:** 192.168.1.201
- 端口:** 3306
- 用户名:** root
- 密码:** root
- 类型:** MYSQL (dropdown menu)
- 库名称:** elint (dropdown menu)
- 数据库列表:** A button to view the database list.
- 连接:** A large button to connect to the database.

2.1.2、选择要建立索引的表，点击“执行”、“创建索引”



2.1.3、弹出如下界面：配置好字段信息:点“创建索引文件”



2.1.4:选择索引服务，然后选择项目，点击建立索引：

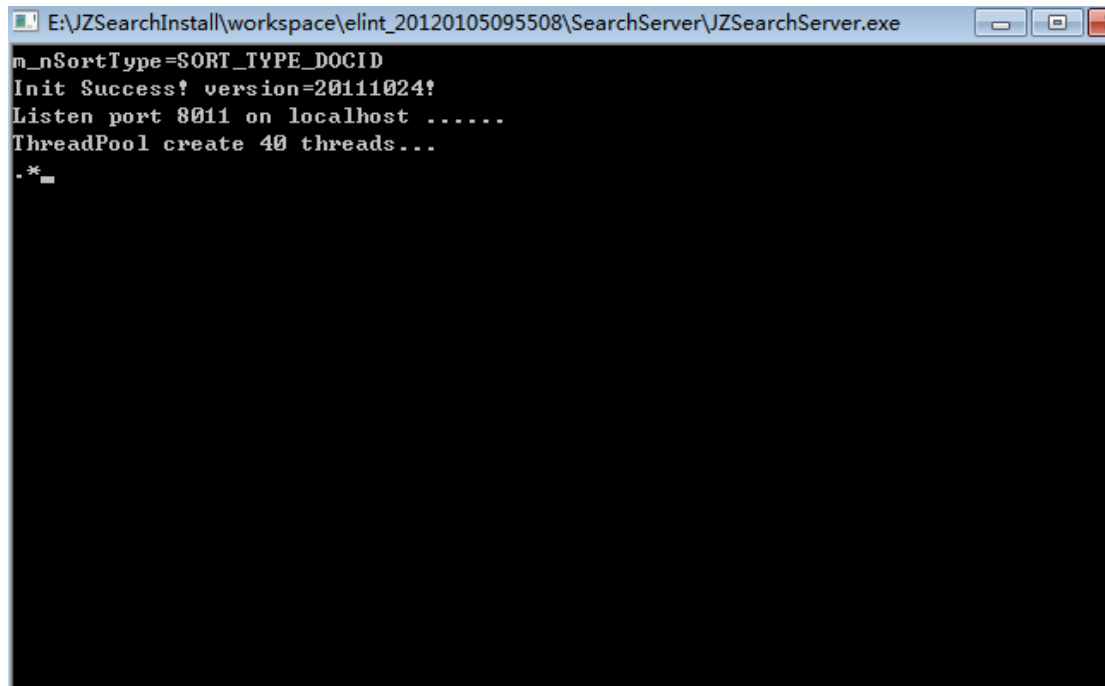
The screenshot shows two windows from the JZSearch application.

The top window, titled "JZSearch数据库检索配置", has a sidebar with a tree view of database schemas. The "索引服务" (Index Service) option is selected. The main area displays a SQL query: `select * from doc`. Below the query is a table of results with columns: t..., id, au..., time, pe..., a..., in..., d..., se..., o..., ac..., w... The table contains 18 rows of data.

The bottom window, titled "索引服务", displays a table of index services with columns: 索引名称 (Index Name), 端口号 (Port), 建立索引 (Build Index), 开启索引服务 (Start Index Service), 在线增量 (Online Incremental), 重建索引 (Rebuild Index), 关闭服务 (Close Service), and 测试索引 (Test Index). The table lists five services, with the one named "elint_201201..." and port "8011" highlighted in blue.

索引名称	端口号	建立索引	开启索引服务	在线增量	重建索引	关闭服务	测试索引
doc_20111206...	8007	建立索引	开启索引服务	在线增量	重建索引	关闭服务	测试索引
elint_201201...	8005	建立索引	开启索引服务	在线增量	重建索引	关闭服务	测试索引
elint_201201...	8008	建立索引	开启索引服务	在线增量	重建索引	关闭服务	测试索引
elint_201201...	8011	建立索引	开启索引服务	在线增量	重建索引	关闭服务	测试索引
person_20111...	8008	建立索引	开启索引服务	在线增量	重建索引	关闭服务	测试索引

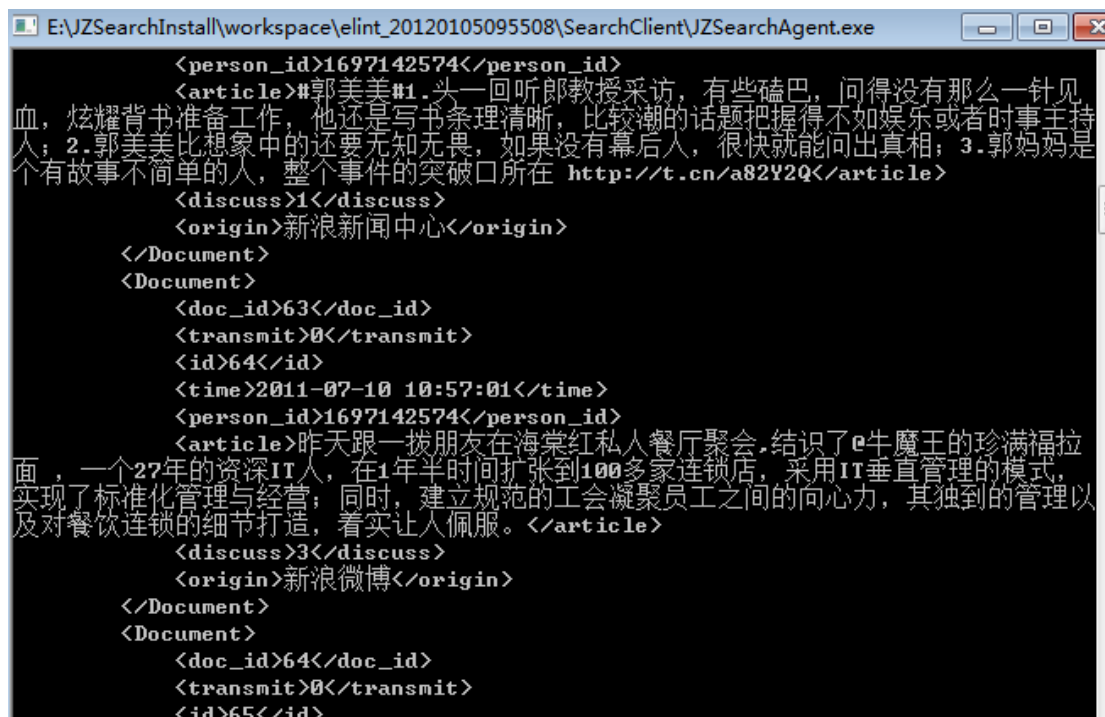
2.1.5: 索引建立成功之后，即可开启索引服务，测试索引：



```

E:\JZSearchInstall\workspace\elint_20120105095508\SearchServer\JZSearchServer.exe
m_nSortType=SORT_TYPE_DOCID
Init Success! version=20111024!
Listen port 8011 on localhost .....
ThreadPool create 40 threads...
.*

```



```

E:\JZSearchInstall\workspace\elint_20120105095508\SearchClient\JZSearchAgent.exe
<person_id>1697142574</person_id>
<article>#郭美美#1.头一回听郎教授采访，有些磕巴，问得没有那么一针见
血，炫耀背书准备工作，他还是写条理清晰，比较潮的话题把握得不如娱乐或者时事主持
人；2.郭美美比想象中的还要无知无畏，如果没有幕后人，很快就能问出真相；3.郭妈妈是
个有故事不简单的人，整个事件的突破口所在 http://t.cn/a82Y2Q</article>
<discuss>1</discuss>
<origin>新浪新闻中心</origin>
</Document>
<Document>
  <doc_id>63</doc_id>
  <transmit>0</transmit>
  <id>64</id>
  <time>2011-07-10 10:57:01</time>
  <person_id>1697142574</person_id>
  <article>昨天跟一拨朋友在海棠红私人餐厅聚会，结识了e牛魔王的珍满福拉
面，一个27年的资深IT人，在1年半时间扩张到100多家连锁店，采用IT垂直管理的模式，
实现了标准化管理与经营；同时，建立规范的工会凝聚员工之间的向心力，其独到的管理以
及对餐饮连锁的细节打造，着实让人佩服。</article>
  <discuss>3</discuss>
  <origin>新浪微博</origin>
</Document>
<Document>
  <doc_id>64</doc_id>
  <transmit>0</transmit>
  <id>65</id>

```

2.2、采用手工配置（适用于各种操作系统）

利用 DBIndex 目录下对应的可执行文件 JZIndexer，搭建数据库搜索引擎服

务的步骤（Windows 下支持 ODBC 方式读取数据库，Linux 下目前支持 MySQL 与 Oracle）。

2.2.1 第一步：配置数据库读取参数

注：如果不是直接读取数据库建立索引，则略过此步。

配置数据库信息，结果写入 index\dbConf.xml

配置信息如下所示：

```
<DatabaseConf>
  <DBConnectionString>DRIVER={MySQL          ODBC          3.51
Driver};SERVER=192.168.1.202;PORT=3306;DATABASE=elint;UID=root;Pwd=ro
ot;</DBConnectionString>    //mysql
  <Host>192.168.1.202</Host>#数据库服务器 IP
  <Username>root</Username>
  <Password>root</Password>
  <Database>elint</Database>
  <Table>elint_web_docinfo</Table>
  <SortField>inserttime</SortField>//用于排序的字段
  <IncrementField>inserttime</IncrementField>
  //用于做增量更新的字段,必须保证后续新增加的字段该值大于前面字段的值
  <MaxMem>524000000</MaxMem>//最大内存数
  <Compare>>=</Compare>
  <ICTCLAS>on</ICTCLAS>
  <RecordPerPage>100000</RecordPerPage>//每批处理的记录数
  <Language>Chinese</Language>//少数民族语言 Minor,
  <Encoding>GBK</Encoding>//编码：GBK,UTF8,BIG5,GBK-FANTI,sqlserver
版本上拷过来的
  <KeyField></KeyField>//关键字索引字段名称列表，必须有一个文本类型的
字段，用于分析判断记录是否已经存在，需要更新数据
</DatabaseConf>
```

其中：

DBConnectionString 为连接数据库的字符串，各个不同的数据库系统的字符串都不太一致，如：

MYSQL:

```
DRIVER={MySQL          ODBC          3.51
Driver};SERVER=192.168.1.103;PORT=3306;DATABASE=elint;UID=root;Pwd
=wysxzw;
```

ACCESS:

```
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=E:\JZSearch\db1.mdb
```

SQL Server:

```
Provider=SQLOLEDB; Server=127.0.0.1;Database=LOS; uid=sa; pwd=sa;
```

Username 为连接数据库的用户名；

Password 为连接数据库的密码；

Database 为指定的数据库名；

Table 为需要建立全文搜索的表名，也可以是视图名称；

SortField 为数据库建立索引时的记录排序字段名称；在本例中，是按照数据库入库的时间字段 `inserttime` 来排序的。

IncrementField 用于增量索引时区分新旧记录的字段，要求后入库记录的该字段值必须大于先入库记录的值；在本例在，是按照数据库入库的时间字段 `inserttime` 来区分先后增量索引的。

MaxMem 为系统建立索引时可以分配的最大内存，内存越大，建立索引越快，32 位环境下最大不能超过 1GB；需要比较大的内存，建议采用 64 位环境下对应的程序，同时，需要结合机器特点，内存小的机器，分配大内存往往导致建立索引不成功；另外，系统需要建立索引的字段。

Compare 增量索引时采取的比较字符串，一般采用 `>=`，采用 `=` 时，只更新上批次最后的记录；

ICTCLAS 为系统建立索引时是否采用 ICTCLAS 智能分词系统；

Language 处理的语言，默认为中英文混杂，选择 **CHINESE**，少数民族语言设置为 **Minor**；

RecordPerPage 为系统索引时，每批加载的记录数；默认为 100 万，如果每天记录较多，建议修改为 10 万，从而提高速度。

KeyField: 关键字索引字段名称列表，必须有一个文本类型的字段，用于分析判断记录是否已经存在，需要更新数据

注：为保障数据库连接成功，需要在索引服务器（可以和数据库服务器不是同一台机器）上，在“系统配置/管理工具”中建立连接该数据库的 ODBC。

2.2.2 第二步：数据库字段信息列表导入

配置字段信息，结果写入 `index\FieldInfo.txt`

`FieldInfo.txt` 内容，参见 1.3 中的示例表。

2.2.3 第三步：建立字段信息文件

执行 `bin\buildTable.bat` 建立字段信息，结果写入 `index\Field.dat`

2.2.4 第四步：自动建立索引

注：如果不是直接读取数据库建立索引，则略过此步，需要自己调用 API 编程，自行建立索引。

执行 `bin\buildIndex.bat` 建立索引

2.2.5 第五步：启动搜索服务

自己建立的索引，也可以启动本服务。

执行 `bin\startServer.bat` 启动搜索服务；

2.2.6 第六步：启动客户端服务

执行 `bin\startClient.bat` 启动客户端服务；

2.2.7 其他脚本

- 1.buildOnlineIndex:建立在线索引
- 2.cleanIndex:清楚索引垃圾信息；
- 3.RebuildAllIndex:重新建立索引，不进行增量更新；
- 4.stopServer: 停止搜索服务；
- 5.update:更新升级程序

2.2.8 Linux 环境

- 1.新将 linux-packet 下面的 install.sh 执行下面命令：
`chmod +x install.sh`
- 2.执行安装命令
`./install.sh`
- 3.其他过程类似于 Windows，不同的是这里面调用的是 `linux-packet\bin` 下面的脚本

三、进阶篇：了解内核

3.1、JZSearch 检索语法说明

JZSearch的检索语法类似于SQL语句，由下面格式的语句组成

```
{[FIELD] fieldname/* [搜索运算符] value1 value2 ... Value_n}*| [SORT] sort_type
```

说明如下：

1. [FIELD] 为搜索系统保留字；
2. Fieldname为系统设置的字段名称，该字段必须在3.2 JZSearch_FieldAdd中定义过，否则非法操作，系统不接受查询请求；其含义为在该字段上进行搜索运算；
3. Filedname也可以为“*”，含义为通配符搜索，不需要指定特定字段，所有涵盖在通配搜索范围的字段都纳入搜索的范畴；
4. 搜索运算符包括：
AND/OR/NOT/NEAR/RANG/MIN/MAX/PREF/PREC/FUZZY，具体含义见后面的详细描述；
5. Value1 Value2等为参数列表
6. 大括号的语句可以反复叠加使用，各个语句之间是逻辑与的关系；
7. 各个部分之间参与空格或者<TAB>键分开，无论是保留字还是参数均不区分大小写；
8. [SORT] sort_type 用于设定当前搜索结果的排序方式

3.1.1 搜索与运算符 AND

表示的含义：后续参数列表的逻辑与关系，举例说明如下：

```
[FIELD] title [AND] 基金 公司
```

含义：搜索 title 字段中同时包含“基金”与“公司”的文档记录；

3.1.2 搜索或运算符 OR

表示的含义：后续参数列表的逻辑或关系，举例说明如下：

```
[FIELD] title [OR] 基金 公司
```

含义：搜索 title 字段中包含“基金”或“公司”的文档记录；

```
[FIELD] * [OR] 基金 公司
```

含义：搜索任意字段中包含“基金”或“公司”的文档记录；

3.1.3 搜索非运算符 NOT

表示的含义：后续参数列表的逻辑非关系，举例说明如下：

[FIELD] *[AND] 基金 [FIELD] title [NOT] 公司 中心

含义：搜索任意字段中包含“基金”，但是标题中不包含“公司”也不包含“中心”的文档记录；

3.1.4 搜索邻近运算符 NEAR

1) 格式为：[FIELD] fieldname/* [NEAR] value1 value2 value3

2) 后续参数为 3 个，value1 value2 value3；前两个参数 value1 value2 为搜索内容，第三个参数 value3 为邻近的距离值；只有两个参数时，默认第三个参数 value3 为 10。

3) 表示的含义：搜索内容 value1 value2 必须同时存在，而且距离不能超过 value3 个词，类似 AND，但比 AND 严格。

举例说明如下：

[FIELD] * [NEAR] 尚福林 卖国贼 9

含义：搜索任意字段中同时包含“尚福林”和“卖国贼”的文档，同时要求两者的距离不能超过 9 个词，采用该语法信息过滤将更有针对性；

3.1.5 搜索智能模糊运算符 FUZZY

1) 表示的含义：搜索方式为模糊匹配，尽可能地匹配搜索串，但模糊处理，不做严格的匹配。

举例说明如下：

[FIELD] * [FUZZY] 张华平 kevinzhang 精准搜索内核

含义：搜索任意字段中“张华平 kevinzhang 精准搜索内核”的文档，按照相关度自动排序；

3.1.6 搜索范围运算符 RANG

1) 格式为：格式为：[FIELD] fieldname/* [RANG] value1 value2

2) 后续参数为 2 个，value1 value2

3) 表示的含义：字段 fieldname 的值必须在 value1 value2 之间的范围内；fieldname 字段必须为数值型字段，否则非法；

4) 注意：如果字段为 datetime 型，其表示形式为：2012/06/20_00:57:17，需要在日期与时间之间加上“_”

举例说明如下：

[FIELD] price [RANG] 1.0 9.0 [FIELD] name [AND] 牛奶 儿童
价格在 1.0~9.0 之间, 且商品名包含 “儿童”、“牛奶” 的记录
[field] title [and] 北 京 [field] publishtime [range]
2012/06/20_00:57:17 2011/06/22_14:34:55 [sort] publishtime desc
含义: 搜索标题包含 “北京”, 且发表时间在 2012/06/20_00:57:17
2011/06/22_14:34:55 之间的搜索结果, 按照 publishtime 倒序排列。

3.1.7 搜索最小值运算符 MIN

- 1) 格式为: 格式为: [FIELD] fieldname/* [MIN] value1
- 2) 后续参数为 1 个, value1
- 3) 表示的含义: 字段 fieldname 的值必须大于或等于 value1; fieldname 字段必须为数值型字段, 否则非法;
- 4) 注意: 如果字段为 datetime 型, 其表示形式为: 2012/06/20_00:57:17, 需要在日期与时间之间加上 “_”

举例说明如下:

[FIELD] price [MIN] 1.0 [FIELD] name [AND] 牛奶 儿童
价格不低于 1.0, 且商品名包含 “儿童”、“牛奶” 的记录

3.1.8 搜索最大值运算符 MAX

- 1) 格式为: 格式为: [FIELD] fieldname/* [MAX] value1
- 2) 后续参数为 1 个, value1
- 3) 表示的含义: 字段 fieldname 的值必须不大于 value1; fieldname 字段必须为数值型字段, 否则非法;
- 4) 注意: 如果字段为 datetime 型, 其表示形式为: 2012/06/20_00:57:17, 需要在日期与时间之间加上 “_”

举例说明如下:

[FIELD] price [MAX] 9.0 [FIELD] name [AND] 牛奶 儿童
价格不高于 9.0, 且商品名包含 “儿童”、“牛奶” 的记录

3.1.9 搜索前缀运算符 PREFIX

- 1) 格式为: 格式为: [FIELD] fieldname/* [PREFIX] value1
- 2) 后续参数为 1 个, value1
- 3) 表示的含义: 字段 fieldname 的内容必须以 value1 开头; fieldname 字段必须为字符型字段, 否则非法;

举例说明如下:

[FIELD] name [PREFIX] 张
姓名字段 name 必须以 “张” 作为前缀开头

3.1.10 搜索精准运算符 PRECISION

- 1) 格式为：格式为：[FIELD] fieldname/* [PREC] value1
- 2) 后续参数为 1 个，value1
- 3) 表示的含义：字段 fieldname 的内容必须精准匹配 value1；fieldname 字段必须为字符型字段，否则非法；

举例说明如下：

[FIELD] name [PREC] abc

姓名字段 name 必须以“abc”精准匹配，如“abcd”不作为匹配结果；

[FIELD] id [PREC] 123

姓名字段 id 必须以“123”精准匹配，如“1234”或者“0123”均不作为匹配结果；

3.1.11 搜索排序运算符 SORT

- 1) 格式为：[SORT] sort_type 用于设定当前搜索结果的排序方式
sort_type参见下表：

类型名	类型值	说明
docid	1	按照 docid 顺序排序，即先建索引的文档优先
relevance	2	按照相关度排序
reverse_docid	3	按照 docid 逆序排序，即后建索引的文档优先
<field_name>descend/ascend	4	按照指定的数值型字段进行排序，

- 2) [SORT]可以出现在搜索语句的起始、中间以及末尾；如果不指定的话，系统会按照系统配置的排序方式进行排序；不做任何指定的前提下，系统按照DOCID排序；

- 3) sort_type不区分大小写，只需要输入前五个字符即可。

- 4) <field_name>descend/ascend field_name必须是数值型，并设置为信息过滤。

- 5) 示例如下：

[FIELD] price [RANG] 1.0 9.0 [FIELD] name [AND] 牛奶 儿童 [SORT] docid
价格在 1.0~9.0 之间，且商品名包含“儿童”、“牛奶”的记录，结果按照 docid 排序

[FIELD] price [RANG] 1.0 9.0 [FIELD] name [AND] 牛奶 儿童 [SORT] relev
价格在 1.0~9.0 之间，且商品名包含“儿童”、“牛奶”的记录，结果按照相关度排序

[FIELD] price [RANG] 1.0 9.0 [FIELD] name [AND] 牛奶 儿童 [SORT] rever
价格在 1.0~9.0 之间，且商品名包含“儿童”、“牛奶”的记录，结果按照 docid 逆序排序

3.1.12 复合运算符 COMPLEX

1) 格式为： [FIELD] fieldname/* [COMPLEX] keyword1|| keyword2||...||keywordN word1|| word2||...||wordMdist

2) 后续参数至少 2 组：

keyword1|| keyword2||...||keywordN:关键词列表(中间用||隔开)

word1|| word2||...||wordM:关键词列表(中间用||隔开)

关键词列表至少需要两组，可以多组兼容

dist: 表示距离，不设置则不考虑之间的距离

3) 表示的含义：

{ keyword1 OR keyword2 OR...OR keywordN} AND {word1 OR word2OR...OR wordM}

如有距离，则表示集合{ keyword1 OR keyword2 OR...OR keywordN}与集合 {word1 OR word2OR...OR wordM}，出现的词必须在 dist 范围以内。

举例说明如下：

[field] title [complex] 台湾||解放军||孕妇 甲流||人事

含义：要求 title 字段内，同时出现 {台湾||解放军||孕妇} {甲流||人事}

[field] content [complex] 统计局||中国统计局||CPI 骗人||砖家 10

含义：要求 content 字段内，同时出现 {统计局||中国统计局||CPI } {骗人||砖家}，且两者之间的距离必须在 10 个词内。

3.1.13 列出所有索引的结果内容 LISTALL

1) 格式为： [CMD] LISTALL

2) 表示的含义：

不给定搜索条件，要求列出所有的索引内容；此时 field 及其他 and or 操作无效。

3) 可以对结果按照 3.1.11 的要求排序

举例说明如下：

[cmd] listall

含义：列出所有的索引内容，默认按照 docid 从小到大排序

[cmd] listall [sort] docid_reverse

含义：列出所有的索引内容，按照 docid 从大到小逆序

3.1.14 正负面情感搜索

正负面搜索采用特殊的搜索词，可以用前面所述的任何一种操作符号
特殊的搜索词参见下表：

类型名	说明
-----	----

##负面 JZSearch##	搜索负面信息
##正面 JZSearch##	搜索正面信息
##色情 JZSearch##	搜索色情信息：目前还没有开放
##政治反动类 JZSearch##	搜索政治反动类信息：目前还没有开放
##毒品类 JZSearch##	搜索毒品类信息：目前还没有开放
##赌博类 JZSearch##	搜索赌博类信息：目前还没有开放
##法轮功类 JZSearch##	搜索法轮功类信息：目前还没有开放
##民运类 JZSearch##	搜索民运类信息：目前还没有开放
##反日类 JZSearch##	搜索反日类信息：目前还没有开放

示例如下：

`[field] content [near] 张华平 ##负面JZSearch## 12`

表达的意思为：在content字段搜索 “张华平” 和负面信息相隔12个词范围内的文档

`[field] content [and] 张华平 ##正面JZSearch## [sort] publishtime desc`

表达的意思为：在content字段搜索 包含“张华平” 和正面信息的文档，且按照发表时间倒序排列。

3.1.15 删除显示运算符 Deleted

1) 格式为： `[deleted] on/off`

2) 表示的含义：

搜索结果中包含删除文档的部分是否需要显示出来，如果为 `on` 将显示，否则不显示。

举例说明如下：

`[FIELD] title [AND] 基金 公司 [deleted] on`

含义：将所有基金公司的信息找出来，如果删除文档中有相关的内容也要显示出来。

3.1.16 两表关联搜索语法

`<index>`

`<no>1</no> //说明：索引在搜索服务中的编号`

`<main>1</main> //主索引，搜索结果全部由该索引产生`

`<join>person_id_int</join> //关联字段名，外键用于关联附属表`

`<cmd> [cmd] list all [sort] publishtime desc</cmd> //搜索语句`

`</index>`

`<index>`

`<no>2</no> //说明：索引在搜索服务中的编号`

`<main>0</main> //从索引`

```

    <join>url_id_long</join>//关联字段名，与上面的 person_id_int 对应
    <cmd>                                [field]                groupName                [or]
ORGAN_6E1F9622F90B5668D76F8E01E824C00D_GROUP</cmd>//搜索语句
</index>

```

执行过程：在第二个索引中执行 [field] groupName [or] ORGAN_6E1F9622F90B5668D76F8E01E824C00D_GROUP 后，生成符合要求的 person_id_int 列表，作为查询条件之一，过滤第一个索引中搜索出来的结果。

3.1.17 搜索聚类运算符 Cluster

- 1) 格式为：格式为：[CLUSTER] fieldname1 fieldname2 ...fieldnameN [Count] Num
- 2) 表示的含义：将搜索结果分别按照字段 fieldname1fieldname2 ...fieldnameN 的内容聚类；fieldname 字段必须为字符型字段且系统已经保存，否则非法；
- 3) Num 表示输出的聚类结果数目，不设置默认为 10.

举例说明如下：

```

[field] * [and] 安全 [cluster] keywords people location [count] 10

```

搜索“安全”，将搜索结果分别按照 keywords people location 字段聚类；这些字段里面的内容按照“#”分割，如 keywords 为“<keywords>举报#信息#工作#奖励#互联绿色情#</keywords>”

搜索结果会增加最后一段：

```

<SearchCluster>
<fieldcount>3</fieldcount>
<Cluster>
<fieldname>keywords</fieldname>
<value>中国信息安全/12#中国安全/8#信息安全/6#技术/5#安全/5#HotDB/4#discussion/4#processing/4#conferences/4#北京大学/4#</value>
</Cluster>
<Cluster>
<fieldname>people</fieldname>
<value>戴浩/8#方安全/3#哈尔/2#罗锋/2#郭建宇/2#赵泽良/2#胡啸/2#舒曼/2#吴琳琳/2#叶红吴亚非刘春林司张胜赵战生赵晓春刘品新吕欣/2#</value>
</Cluster>
<Cluster>
<fieldname>location</fieldname>
<value>中国/21#中国中华人民共和国/12#北京/7#中国江苏阜宁/4#大都/3#中国湖北省宜昌市/3#安定门/2#美国/2#台中/2#中国北京北京市江苏阜宁/2#</value>
</Cluster>
</SearchCluster>

```

其中每个值由 [word]/[freq]#组成。

3.1.18 分组运算符 Groupby

- 1) 格式为: 格式为: [Groupby] fieldname1 fieldname2 ...fieldnameN
 - 2) 表示的含义: 将搜索结果分别按照字段 fieldname1fieldname2 ...fieldnameN 的值进行分组; fieldname 字段必须为数值型字段, 否则非法;
- 举例说明如下:
- [field] * [and] 安全 [groupby] type_id
- 搜索“安全”, 将搜索结果分别按照 type_id 字段分组;

搜索结果会增加最后一段:

```
<Groupby>
<fieldcount>1</fieldcount>
<group>
<fieldname>keywords</fieldname>
  <groupcount>3</groupcount>
  <item>
    <value>4</value><count>3</count>
  </item>
  <item>
    <value>23</value><count>2</count>
  </item>
  <item>
    <value>8</value><count>1</count>
  </item>
</group>
</Groupby>
```

3.1.19 搜索结果比较运算符 Compare Between With

- 1) 格式为: 格式为: [Compare] fieldname1 [between] search_cmd1 [with] search_cmd2
- 2) 表示的含义: 将[field]fieldname2 value1 {其他命令}与[field]fieldname2 value2 {其他命令}的搜索结果进行比较, 比较对应的 fieldname1 字段值的异同。

举例说明如下:

```
[Compare] keywords [between] [field] * [and] 华为 [field] type [and] news
[with] [field] * [and] 联想 [field] type [and] news
搜索新闻中“华为”“联想”的文章搜索结果中, 比较对应的 keywords 字段的
异同,
[Compare] keywords [between] [field] * [and] 华为 [field] publishtime
[range] 2012/1/1 2012/12/31 [with] [field] * [and] 华为 [field]
publishtime [range] 2013/1/1 2013/12/31
```

搜索结果格式如下所示：

<compare>

<same>信息/15/3#网络/3/15#标准/8/3#技术/7/1#信息安全/5/1#互联网/2/4#电子信息/3/1#中国/2/2#服务/2/2#服务信息中心标/2/2#</same>

<diff1>中国信息安全/12#信息技术/5#HotDB/4#conferences/4#机/4#北京大学/4#标准化/4#研究/4#discussion/4#researchers/4#</diff1>

<diff2>高校/5#人才/5#军/5#网罗/5#打造/5#薛蛮子/5#科技/5#民警/3#风范/3#微博/3#</diff2>

</compare>

3.1.20 搜索结果自动去重处理 Unique

1) 格式为：格式为：[Unique] fieldname1 fieldname2 .. fieldnameN

2) 表示的含义：将搜索结果按照字段 fieldname1 fieldname2 .. fieldnameN 对搜索结果进行去重处理，也就是说不会重复出现 <fieldname1 fieldname2 .. fieldnameN> 对应完全一致的搜索结果。

举例说明如下： [field] * [and] 图 [unique] key

表述的是：统配搜索"图"，搜索结果自动 key 必须都不一样

3.1.21 搜索结果条件或处理 JZOR

1) 格式为：格式为：[查询语句 1] [JZOR] [查询语句 2]... [JZOR] [查询语句 n]

2) 表示的含义：将[查询语句 1]、[查询语句 2]、[查询语句 n]的搜索结果进行或运算。这里的或运算与 [OR] 存在差异，后者是对同一个字段实现或运算，而[JZOR]更加宽泛。

举例说明如下： [FIELD] content [AND] "福泽全球" [jZor] [FIELD] title [OR] "一个伟大民族" [jZor] [FIELD] paragraph [OR] "人民"

表述的是：将如下三条搜索语句的搜索结果进行或运算

- 1) 搜索 content 字段，要求包含"福泽全球"
- 2) 搜索 title 字段，要求包含"一个伟大的民族"
- 3) 搜索 paragraph 字段，要求包含"人民"

3.2 JZSearch 搜索后台服务系统搭建

1. 可以支持 Windows、Linux、FreeBSD 等多种环境，支持普通 PC 机器即可运行，目前支持 C/C++调用，索引程序不建议采用非 C 程序实现。搜索前端目前已经有 C/C++/Php/Java 等实现版本；

2. 基于 JZSearch 已经实现了后台的搜索服务，可以与数据库无缝融合。

3. 命令行：JZSearchServer.exe [搜索目录 1] [搜索目录 2] ... [搜索目录 n]

4. 说明：搜索服务需要寻找指定目录下的 SearchServer.xml，根据 SearchServer.xml 的内容提供搜索服务

5. SearchServer.xml 示例如下：

```
<?xml version="1.0" encoding="GB2312"?>
<SearchServerConf>
  <ReleaseVersion>
    <version>3.0</version>
    <lastModify>2011-8-28</lastModify>
  </ReleaseVersion>
  <Config>
    <ServerIP>127.0.0.1</ServerIP>##检索服务器 IP 地址
    <Port>8003</Port>##检索服务进程的端口号
    <dictpath>D:\NLPIR\release\JZSearchInstall\dict</dictpath>##词典等目标文件
    <ICTCLAS>on</ICTCLAS>
    <index>
      <indexfile>D:\NLPIR\test\uyghan\JZSearch</indexfile>## 原始标准数据及索引
      结果文件
      <FieldInfo>D:\NLPIR\test\uyghan\fieldinfo.dat</FieldInfo>##字段信息文件
      <SORT>DOCID</SORT>
      ##排序方式：
      ##DOCID=按照 DOCID 顺序排序（即文档建索引的顺序），默认方式；DOCID=
      按照 DOCID 排序
      ##REVERSE_DOCID=按照 DOCID 逆序排序（即后建索引的结果排名靠前）
      ##RELEVANCE=按照相关度排序
```

```

    <Encoding>UTF8</Encoding>//编码: GBK,UTF8,BIG5,GBK-FANTI
    <GroupMaxCount>50</GroupMaxCount>//分组最大条目数
    <MexMem>1024000000</MexMem>//最大内存数
    <Language>Minor</Language>//少数民族语言
  </index>
</Config>
</SearchServerConf>

```

3.3 JZSearch 客户端搭建与管理指南

3.3.1 客户端管理命令语法

客户端的命令都是采用搜索服务控制命令实现的,服务端控制命令类似于搜索语法,具体细节如下:

命令格式	说明
[CMD] RELOAD	表示重新加载新增的索引内容,系统会自动维护,不建议使用
[CMD] MERGE	表示需要对索引数据进行优化归并,系统会自动维护,不建议使用
[CMD] QUIT	表示要将系统搜索服务停止
[CMD] Version	查询搜索引擎的版本号
[CMD] Backup	索引自动备份,系统会自动维护,不建议使用
[CMD] Restore	索引自动恢复,系统会自动维护,不建议使用
[CMD] del id1 id2 ... idn	从索引中删除后续编号的文档
[cmd] WHERE- DEL 【搜索语句】	将符合【搜索语句】条件的记录删除
[cmd] WHERE- UPDATE 【搜索语句】	将符合【搜索语句】条件的记录删除,但保留最新的一条

3.3.2 命令行方式

命令行: JZSearchAgent.exe 参数 【Linux 有配套程序】

参数说明:

0. 参数不区分大小写；
1. 参数为空，系统接收用户实时输入的命令，命令行参考搜索语法手册、JZSearch 检索语法说明；
2. QUIT: 表示要将系统搜索服务停止；
3. Reload: 表示重新加载新增的索引数据；
4. Merge: 表示需要对索引数据进行优化归并；
5. VER: 查询搜索引擎的版本号
6. BACKUP: 索引自动备份
7. RESTORE: 索引自动恢复
8. delete: 从索引中删除后续参数的文档
9. INDEX: 启动索引，并自动加载进入搜索进程

需要事先配置索引的脚本，Windows 下命名为 buildIndex.bat

Linux 下命名为 buildIndex.sh

3.3.3 C 语言 API 方式

具体存放在 SearchAgentAPI/Win-C 目录下

```
JZSEARCHAGENTAPI_API int jzsearch_query_client(const char *XMLfile,const
char *query_line,int nStart,int nPageCount,const char *sResultName);
//SearchAgent.XML 的存放地址
//query_line:查询的语句 [FIELD] * [AND] 钢铁 ;*表示除标准号之外的通配符;
2.AND 后面可以放多个值;
//nStart: 起始结果的行号, 从 0 开始
//nPageCount:每页结果数目
//sResultName:result.xml 存放的文件名
//
```

3.3.4 JAVA 语言调用搜索客户端

具体存放在 SearchAgentAPI/Java 目录下

3.3.5 调用搜索服务的协议说明

查询协议

1. **Start**: 查询起始记录号, 整型数据, 四个字节;
2. **maxPageCount**: 当前页面记录总数, 整型数据, 四个字节;
3. **query.size**: 查询语句的长度, 整型数据, 四个字节;
4. **query**: 查询语句, 字符串类型, 长度: **query.size**

注意: 所有的整数类型都需要转换成网络协议支持的格式, 如C语言中, 需要调用函数htonl进行格式转换。

搜索结果协议

1. **count**: 搜索结果总数, 整型数据, 四个字节;
2. **start_ret**: 返回的起始记录号, 整型数据, 四个字节;
3. **pagecount_ret**: 返回的当前页面记录总数, 整型数据, 四个字节;
4. **col_count**: 每条记录的字段数, 也成为列数, 整型数据, 四个字节;
5. **buf_size**: 内容缓存的长度, 整型数据, 四个字节;
6. 申请二维整型数组pVecLength[pagecount_ret][col_count], 依次读取col_count*pagecount_ret个整型数据, 每个四个字节; pVecLength[i][j]表示的是第i个搜索结果第j个字段的内容长度, 内容从7中的content读取; i, j从0开始编号;
7. 申请大小为buf_size的内容串content, 读取buf_size个字符; 按照6中pVecLength[i][j]的值对缓存的内容串进行分割即可。
8. 如果有搜索分组操作, 继续下面的操作; 否则, 可直接结束
9. 分组信息读取, 详细见下面的C语言代码

```
m_pResult->group_count=ntohl(*((int*)(pBuf+m_nBufSize)));
m_nBufSize+=sizeof(tuint);
m_pResult->p_vecGroupResult=new _tGroupResult[m_pResult->group_count];
int i, j;

for (i=0; i<m_pResult->group_count; i++)
{
    m_pResult->p_vecGroupResult[i].field_id=ntohl(*((int*)(pBuf+m_nBufSize)));
    m_nBufSize+=sizeof(tuint);

    m_pResult->p_vecGroupResult[i].item_count=ntohl(*((int*)(pBuf+m_nBufSize)));
```

```
m_nBufSize+=sizeof(tuint);

m_pResult->p_vecGroupResult[i].p_count_list=newint
[m_pResult->p_vecGroupResult[i].item_count];
m_pResult->p_vecGroupResult[i].p_val_list=new FILTER_DATA_TYPE
[m_pResult->p_vecGroupResult[i].item_count];

for (j=0;j<m_pResult->p_vecGroupResult[i].item_count;j++)
{

m_pResult->p_vecGroupResult[i].p_count_list[j]=ntohl(*((int*)(pBuf+m_nBufSize)));
m_nBufSize+=sizeof(tuint);

m_pResult->p_vecGroupResult[i].p_val_list[j]=ntoh_double(*((FILTER_DATA_TYPE*)(pBuf
+m_nBufSize)));
m_nBufSize+=sizeof(FILTER_DATA_TYPE);
}
}
```

注意：所有的整数类型都需要将网络协议支持的格式转换成开发语言支持的格式，如C语言中，需要调用函数ntohl进行格式转换。

四、高级篇：API 定制开发

4.1 字段定义接口

4.1.1 JZIndexer_FieldAdd 添加一个字段

```
JZSearchAPI_API bool JZIndexer_FieldAdd(const char *sFieldName, const char
*sDBFieldName, int nFieldType, bool bIndexed, bool bRetrieved, bool bGeneral=false, bool
bAbstracted=false);
```

Routine	Required Header
JZIndexer_FieldAdd	<JZSearchAPI.h>

Return Value

Return true if init succeed. Otherwise return false.

Parameters

const char *sFieldName: 字段名称，在索引建立与搜索过程中，字段的唯一标示符

const char *sDBFieldName:: 对应于数据库的字段名称；在数据库搜索的时候，用于获取数据库的数据

int nFieldType: 数据类型，参加 3.1 节字段类型定义表

bIndexed: 该字段是否需要索引，true，需要索引，以后可以搜索该字段，否则，不提供搜索

bRetrieved: 搜索结果中是否要输出该字段内容

bGeneral: 是否纳入通配搜索的范畴

bAbstract: 是否进行搜索摘要提取并红显

Remarks

Example

```
//设置字段信息
```

```
JZIndexer_FieldAdd("title", NULL, FIELD_TYPE_TEXT, true, true, true, true);
```

```
//对标题建索引，需要搜索，需要通配搜索，需要存储，需要红显
```

```
JZIndexer_FieldAdd("content", NULL, FIELD_TYPE_TEXT, true, true, true, true);
```

```
//对内容建索引，需要搜索，需要通配搜索，需要存储，需要红显
```

```
JZIndexer_FieldAdd("id", NULL, FIELD_TYPE_INT, true, true, true, false);
```

```
JZIndexer_FieldAdd("idf", NULL, FIELD_TYPE_FLOAT, true, true, true, false);
```

```
//对内容建索引，需要搜索
```

```
JZIndexer_FieldSave("./index/FieldInfo.dat");
```

Output

4.1.2 JZIndexer_FieldSave 保存已经设置的字段信息

JZSearchAPI_API bool JZIndexer_FieldSave(const char *sFieldInfoDataFile);

保存的字段信息数据文件

Routine	Required Header
JZIndexer_FieldSave	<JZSearchAPI.h>

Return Value

Return true if init succeed. Otherwise return false.

Parameters

const char *sFieldInfoDataFile: 字段信息文件，系统特定的格式，建立索引和搜索都需要作为参数。

Remarks

Example

```
//设置字段信息
```

```
JZIndexer_FieldAdd("title", NULL, FIELD_TYPE_TEXT, true, true, true);
```

```
//对标题建索引，需要搜索
```

```
JZIndexer_FieldAdd("content", NULL, FIELD_TYPE_TEXT, true, true, true);
```

```
//对内容建索引，需要搜索
```

```
JZIndexer_FieldAdd("id", NULL, FIELD_TYPE_INT, true, true, true);
```

```
JZIndexer_FieldAdd("idf", NULL, FIELD_TYPE_FLOAT, true, true, true);
```

//对内容建索引，需要搜索

```
JZIndexer_FieldSave("./index/FieldInfo.dat");
```

Output

4.1.3 JZIndexer_FieldLoad 读取已经设置的字段信息文件

```
JZSearchAPI_API bool JZIndexer_FieldLoad(const char *sFieldInfoDataFile);
```

读取已经保存好的字段信息数据文件

Routine	Required Header
JZIndexer_FieldLoad	<JZSearchAPI.h>

Return Value

Return true if init succeed. Otherwise return false.

Parameters

const char *sFieldInfoDataFile: 字段信息文件，系统特定的格式，建立索引和搜索都需要作为参数。

Remarks

Example

```
JZIndexer_FieldLoad("./index/FieldInfo.dat");
```

Output

```
JZSearchAPI_API bool JZIndexer_FieldLoad(const char *sFieldInfoDataFile);
```

4.2 索引接口

4.2.1 JZIndexer_Init 精准索引器初始化

```
JZSearchAPI_API bool JZIndexer_Init(const char *sDictPath=0,const char *sFieldInfoFile=0, int encoding=INDEX_ENCODING_GBK_FANTI);
```

Routine	Required Header
JZIndexer_Init	<JZSearchAPI.h>

Return Value

Return true if init succeed. Otherwise return false.

Parameters

//sDictPath: 词典文件所在位置; 为空时, 采用 n-gram 索引方法

//sFieldInfoFile:域字段信息, 用于支持多域索引, 为空则只支持一个字段, 参见 3.3 节 JZIndexer_FieldSave.

//encoding:设置编码, 具体可以参见 1.5

Remarks

1. 索引系统初始化, 必须初始化后, 才能使用 CJZIndexer
2. 初始化失败的原因包括:
 - 1) 数据文件不齐全: sDictPath 需要齐备的文件包括 irreg2regularEnglish.map、dictionary.wordlist、dictionary.pdat、english.pdat
 - 2) 字段定义数据文件非法: 即不是系统自动生成的字段信息
 - 3) 授权过期
- 3.失败时, 可以查看当前目录下以日期命名的 log 日志文件, 有具体的提示;

Example

```
JZIndexer_Init("E:\\Projects\\LJSearch\\dat", 0, "LingjoinICTCLAS2010*~!@#$$%&&$$#@!~*");
```

Output

4.2.2 JZIndexer_Exit 精准索引器系统退出

```
JZSearchAPI_API bool JZIndexer_Exit();
```

//索引器系统退出

Routine	Required Header
JZIndexer_Exit	<JZSearchAPI.h>

Return Value

Return true if init succeed. Otherwise return false.

Parameters

None

Remarks

1. 执行 JZIndexer_Exit, 才能释放资源

Example

```
JZIndexer_Exit();
```

Output

4.2.3 CJZIndexer 精准索引器类

//建立索引的类

```
class JZSearchAPI_API CJZIndexer {
public:
    bool Export(const char *sFilename,const char *sWordList);
    //内存大小控制
    CJZIndexer(const char *pcIndexFile,int nMaxMemSize=512000000 );

    ~CJZIndexer(void);
    // TODO: add your methods here.
    int MemIndexing(const char *pText,const char *sFieldName=0,int nMemSize=0);
    //索引一段内存, ,
    //sFieldName: 字段名称, 为空则表示无字段信息

    int FileIndexing(const char *sTextFilename,const char *sFieldName=0);
    //索引一个文本文件,
    //sFieldName: 字段名称, 为空则表示无字段信息

    int IdIndexing(int term_id,const char *sFieldName=0);
    //词 ID 索引,
    //sFieldName: 字段名称, 为空则表示无字段信息

    int IntIndexing(int value,const char *sFieldName=0);
    //对数值进行索引,
    //sFieldName: 字段名称, 为空则表示无字段信息
    int FloatIndexing(float value,const char *sFieldName=0);
    //对数值进行索引,
```



```
//sFieldName: 字段名称, 为空则表示无字段信息
bool Save();
//索引保存的名称

//索引保存的名称
int Merge();
//索引合并

int AddDoc();//增加文档

private://以下部分为系统使用, 应用开发者不要改写, 只能读取数据
    int m_nHandle;//索引器的 Handle, 无需调用申请
};
```

Routine	Required Header
JZIndexer_Exit	<JZSearchAPI.h>

Return Value

None

Parameters

None

Remarks

1. 索引器类, 可以并行多线程处理

Example

Output

4.2.3.1 CJZIndexer::CJZIndexer 精准索引器类构造函数

```
CJZIndexer(const char *pcIndexFile,int nMaxMemSize=512000000 );
```

Routine	Required Header
CJZIndexer: : CJZIndexer	<JZSearchAPI.h>

Return Value

None

Parameters

`const char *pcIndexFile`: 指定的索引文件名，系统根据该文件名自动生成相应的数据文件，检索时使用，索引与检索的文件必须一致。

`int nMaxMemSize=512000000`: 系统给定的最大内存大小，默认为 512MB。精准搜索索引时内存由系统自行分配与管理。

Remarks:

系统异常时，可以查看当前目录下以日期命名的 log 日志文件，有具体的提示；

Example

```
CJZIndexer *pIndexer=new CJZIndexer("./index/Test0809",50000000);
```

Output

4.2.3.2 CJZIndexer::MemIndexing 精准索引器类内存索引函数

```
int MemIndexing(const char *pText,const char *sFieldName=0,int nMemSize=0);
```

索引一段内存

//sFieldName: 字段名称，为空则表示无字段信息

Routine	Required Header
CJZIndexer: : MemIndexing	<JZSearchAPI.h>

Return Value

int

Parameters

`const char *pText`: 待索引的内存文本内容起始地址。

`const char *sFieldName`: 指定的给字段名称，需要跟 3.2 定义的字段名称一致。

`Int nMemSize`: 待索引的内存文本内容长度，为 0，则按照字符串结束标准 '\0' 作为内存结束标记。

Remarks:

1. 对文本内容进行索引；系统默认只处理前 5KB 的内容，超出部分予以截断；
2. 字段可以是文本字段也可以是数值型字段；
3. 索引完成后，需要调用 AddDoc 方有效；

Example

```
CJZIndexer *pIndexer=new CJZIndexer("./index/Test0809",50000000);

//对文档列表进行索引

for (int doc_id=nDocID_Start;doc_id<vecFileList.size();doc_id++)

{
    //索引一个文件

    pIndexer->MemIndexing(vecFileList[doc_id].c_str(),"title");//对文件标题建立索引

    pIndexer->FileIndexing(vecFileList[doc_id].c_str(),"content");//对文件内容建立索引

    pIndexer->IntIndexing(doc_id,"id");//对 ID 建立索引

    pIndexer->FloatIndexing(doc_id,"idf");//对 IDF 建立索引

    pIndexer->AddDoc();
}

pIndexer->Save();//保存结果 Output
```

4.2.3.3 CJZIndexer::FileIndexing 精准索引器类文件索引函数

```
int FileIndexing(const char *sTextFilename,const char *sFieldName=0);
//索引一个文本文件，
//sFieldName: 字段名称，为空则表示无字段信息
```

Routine	Required Header
CJZIndexer: : FileIndexing	<JZSearchAPI.h>

Return Value

int

Parameters

const char *sTextFilename: 待索引的文本文件名。

const char *sFieldName: 指定的给字段名称, 需要跟 3.2 定义的字段名称一致。

Remarks:

1. 系统自动读取文件内容, 并建立索引; 系统默认只处理前 5KB 的内容, 超出部分予以截断
2. 字段可以是文本字段也可以是数值型字段;
3. 索引完成后, 需要调用 AddDoc 方有效;

Example

```
CJZIndexer *pIndexer=new CJZIndexer("./index/Test0809",50000000);

//对文档列表进行索引

for (int doc_id=nDocID_Start;doc_id<vecFileList.size();doc_id++)

{
    //索引一个文件

    pIndexer->MemIndexing(vecFileList[doc_id].c_str(),"title");//对文件标题
    建立索引

    pIndexer->FileIndexing(vecFileList[doc_id].c_str(),"content");//对文件
    内容建立索引

    pIndexer->IntIndexing(doc_id,"id");//对 ID 建立索引

    pIndexer->FloatIndexing(doc_id,"idf");//对 IDF 建立索引

    pIndexer->AddDoc();

}

pIndexer->Save();//保存结果 Output
```

4.2.3.4 CJZIndexer::BigFileIndexing 精准索引器类大文件索引函数

```
int BigFileIndexing(const char *sBigFilename,unsigned int nOffset,unsigned int nSize,const char *sFieldName=0);
```

//索引一个 Big File 中指定的偏移量和长度的文本文件，

Routine	Required Header
CJZIndexer: : BigFileIndexing	<JZSearchAPI.h>

Return Value

int

Parameters

//sBigFilename: Big File 的文件名;

//nOffset:偏移量;

//nSize: 文件长度

//sFieldName: 字段名称，为空则表示无字段信息

Remarks:

1. 系统自动读取文件内容，并建立索引；系统默认只处理前 5KB 的内容，超出部分予以截断
2. 索引完成后，需要调用 AddDoc 方有效；

Example

```
if (pFieldInfo->path.size()>0)
{
    strcpy(pFilename,pFieldInfo->path.c_str()); //get path
    strcat(pFilename,PATH_DELEMETER); //分隔符
}
strcat(pFilename,pUnit); //filename

pUnit = row[nColID++]; //get offset
nOffset=0;
if (pUnit!=0)
{
    nOffset=atoi(pUnit);
```

```
    }
    pUnit = row[nColID++]; //get length
    nLen=0;
    if (pUnit!=0)
    {
        nLen=atoi(pUnit);
    }
#ifdef LJSEARCH_DEBUG

    printf("m_pIndexer->BigFileIndexing(%s,%d,%d,%s)\n",pFilename,nOf
fset,nLen,pFieldInfo->index_name.c_str());
#endif

    m_pIndexer->BigFileIndexing(pFilename,nOffset,nLen,pFieldInfo->in
dex_name.c_str());

    //Big File Indexing
```

4.2.3.5 CJZIndexer::IdIndexing 精准索引器类 ID 索引函数

```
int IdIndexing(int term_id,const char *sFieldName=0);
//词 ID 索引,
//sFieldName: 字段名称, 为空则表示无字段信息
```

Routine	Required Header
CJZIndexer: : IdIndexing	<JZSearchAPI.h>

Return Value

Int

Parameters

int term_id: 词 ID, ID 需要大于 0,超出词典词总数, 调用无效

const char *sFieldName: 指定的给字段名称, 需要跟 3.2 定义的字段名称一致。

Remarks:

1. 字段可以是文本字段也可以是数值型字段;
2. 索引完成后, 需要调用 AddDoc 方有效;

Example

```
CJZIndexer *pIndexer=new CJZIndexer("./index/Test0809",50000000);

//对文档列表进行索引

for (int doc_id=nDocID_Start;doc_id<vecFileList.size();doc_id++)

{
    //索引一个文件

    pIndexer->MemIndexing(vecFileList[doc_id].c_str(),"title");//对文件标题
    建立索引

    pIndexer->FileIndexing(vecFileList[doc_id].c_str(),"content");//对文件
    内容建立索引

    pIndexer->IdIndexing(term_id,"term_id");//对词 ID 建立索引

    pIndexer->IntIndexing(doc_id,"id");//对 ID 建立索引

    pIndexer->FloatIndexing(doc_id,"idf");//对 IDF 建立索引

    pIndexer->AddDoc();

}

pIndexer->Save();//保存结果 Output
```

4.2.3.6 CJZIndexer::IntIndexing 精准索引器类整型索引函数

```
int IntIndexing(int value,const char *sFieldName=0);
//对整型数值进行索引,
//sFieldName: 字段名称, 为空则表示无字段信息
```

Routine	Required Header
CJZIndexer: : IntIndexing	<JZSearchAPI.h>

Return Value

int

Parameters

int value: 数值

`const char *sFieldName`: 指定的给字段名称, 需要跟 3.2 定义的字段名称一致。

Remarks:

1. 此处字段的类型必须是 `FIELD_TYPE_INT`;
2. 索引完成后, 需要调用 `AddDoc` 方有效;

Example

```
CJZIndexer *pIndexer=new CJZIndexer("./index/Test0809",50000000);

//对文档列表进行索引

for (int doc_id=nDocID_Start;doc_id<vecFileList.size();doc_id++)

{
    //索引一个文件

    pIndexer->MemIndexing(vecFileList[doc_id].c_str(),"title");//对文件标题
    建立索引

    pIndexer->FileIndexing(vecFileList[doc_id].c_str(),"content");//对文件
    内容建立索引

    pIndexer->IntIndexing(doc_id,"id");//对 ID 建立索引

    pIndexer->FloatIndexing(doc_id,"idf");//对 IDF 建立索引

    pIndexer->AddDoc();

}

pIndexer->Save();//保存结果 Output
```

4.2.3.7 CJZIndexer:: LongIndexing 精准索引器类 64 位长整型索引函数

```
#ifndef _WIN32
    int LongIndexing(long long value,const char *sFieldName=0);
    //对 64 位数值进行索引,
    //sFieldName: 字段名称, 为空则表示无字段信息
#else
    int LongIndexing(__int64 value,const char *sFieldName=0);
    //对 64 位数值进行索引,
    //sFieldName: 字段名称, 为空则表示无字段信息
```



```
#endif
```

Routine	Required Header
CJZIndexer: : LongIndexing	<JZSearchAPI.h>

Return Value

int

Parameters

value: 64 位数值

const char *sFieldName: 指定的给字段名称, 需要跟 3.2 定义的字段名称一致。

Remarks:

1. 此处字段的类型必须是 FIELD_TYPE_LONG;
2. 索引完成后, 需要调用 AddDoc 方有效;

Example

```
CJZIndexer *pIndexer=new CJZIndexer("./index/Test0809",50000000);

//对文档列表进行索引

for (int doc_id=nDocID_Start;doc_id<vecFileList.size();doc_id++)

{
    //索引一个文件

    pIndexer->MemIndexing(vecFileList[doc_id].c_str(),"title");//对文件标题
    建立索引

    pIndexer->FileIndexing(vecFileList[doc_id].c_str(),"content");//对文件
    内容建立索引

    pIndexer->LongIndexing(doc_id,"id");//对 ID 建立索引

    pIndexer->FloatIndexing(doc_id,"idf");//对 IDF 建立索引

    pIndexer->AddDoc();

}

pIndexer->Save();//保存结果 Output
```

4.2.3.8 CJZIndexer::FloatIndexing 精准索引器类浮点型索引函数

```
int FloatIndexing(float value,const char *sFieldName=0);  
//对浮点型数值进行索引,  
//sFieldName: 字段名称, 为空则表示无字段信息
```

Routine	Required Header
CJZIndexer: : FloatIndexing	<JZSearchAPI.h>

Return Value

int

Parameters

float value: 数值

const char *sFieldName: 指定的给字段名称, 需要跟 3.2 定义的字段名称一致。

Remarks:

1. 此处字段的类型必须是 FIELD_TYPE_FLOAT;
2. 索引完成后, 需要调用 AddDoc 方有效;

Example

```
CJZIndexer *pIndexer=new CJZIndexer("./index/Test0809",50000000);  
  
//对文档列表进行索引  
  
for (int doc_id=nDocID_Start;doc_id<vecFileList.size();doc_id++)  
  
{//索引一个文件  
  
    pIndexer->MemIndexing(vecFileList[doc_id].c_str(),"title");//对文件标题  
    建立索引  
  
    pIndexer->FileIndexing(vecFileList[doc_id].c_str(),"content");//对文件  
    内容建立索引  
  
    pIndexer->IntIndexing(doc_id,"id");//对 ID 建立索引
```

```
pIndexer->FloatIndexing(doc_id,"idf");//对 IDF 建立索引

pIndexer->AddDoc();

}

pIndexer->Save();//保存结果 Output
```

4.2.3.9 CJZIndexer::AddDoc 精准索引器类文档添加函数

int AddDoc();//增加文档

当前文档索引完成，提交文档。准备索引下一篇文章档;系统

Routine	Required Header
CJZIndexer: : AddDoc	<JZSearchAPI.h>

Return Value

Int

成功则返回内部的 doc_id 号；否则为 0

Parameters

None

Remarks:

1. 对当前文档索引完成后，需要执行 AddDoc 提交，方可处理下一篇文章档；
2. 内部将维护一个默认的 doc_id；AddDoc 执行后，doc_id 将自动加 1；
3. 一个索引器默认处理最大的文档数目为 2 亿，如果超过该上限，系统执行 AddDoc 将返回 0，需要重新启动一个索引器进行所有，实现多个索引器并行多机器搜索。

Example

```
nLineId=1;

while(fgets(sLine, 30690, fp) !=0)

{

    start=clock();

    pIndexer->MemIndexing(sLine,"content");//对文件内容建立索引
```

```
end=clock();

nTotalTime+=end-start;

nTotalLength+=strlen(sLine);

pIndexer->IntIndexing(doc_id,"file_index");

pIndexer->IntIndexing(nOffset,"offset");

if(!pIndexer->AddDoc())

    {

        //当前文档索引完成，提交文档。准备索引下一篇文章

        printf("Enough buffer!");

        break;

    }

if (nLineId%100==0)

    {

        printf("Indexing %s:Line %d
Completed!\n",vecFileList[doc_id].c_str(),nLineId);

    }

nLineId++;

nOffset=ftell(fp);

    }
```

4.2.3.10 CJZIndexer::Save 精准索引器类保存函数

```
bool Save();
//索引保存的名称
```

Routine	Required Header
CJZIndexer: : Save	<JZSearchAPI.h>

Return Value

Bool 成功则返回 true； 否则为 false

Parameters

None

Remarks:

1. 保持索引文件；

Example

```
for (int doc_id=nDocID_Start;doc_id<vecFileList.size();doc_id++)

    {//索引一个文件

        pIndexer->MemIndexing(vecFileList[doc_id].c_str(),"title");//对文件标题
        建立索引

        pIndexer->FileIndexing(vecFileList[doc_id].c_str(),"content");//对文件
        内容建立索引

        pIndexer->IntIndexing(doc_id,"id");//对 ID 建立索引

        pIndexer->FloatIndexing(doc_id,"idf");//对 IDF 建立索引

        pIndexer->AddDoc();

    }

    pIndexer->Save();//保存结果
```

4.2.3.11 CJZIndexer::Merge 精准索引器类索引合并函数

```
int Merge();
//索引合并
```

Routine	Required Header
CJZIndexer: : Merge	<JZSearchAPI.h>

Return Value

Int

Parameters

None

Remarks:

1. 将小的索引文件碎片，合并为完整较大的文件，以确保搜索的效率；
2. 一般在增量索引比较多次的情况下，调用

Example

```
for (int doc_id=nDocID_Start;doc_id<vecFileList.size();doc_id++)

{ //索引一个文件

    pIndexer->MemIndexing(vecFileList[doc_id].c_str(),"title");//对文件标题
    建立索引

    pIndexer->FileIndexing(vecFileList[doc_id].c_str(),"content");//对文件
    内容建立索引

    pIndexer->IntIndexing(doc_id,"id");//对 ID 建立索引

    pIndexer->FloatIndexing(doc_id,"idf");//对 IDF 建立索引

    pIndexer->AddDoc();

}

pIndexer->Save();//保存结果

pIndexer->Merge();//索引文件归并处理
```

4.2.3.12 CJZIndexer::Export 精准索引器类索引导出函数

JZSearchAPI_API bool Export(const char *sExportFile,const char *sWordList);

导出的索引数据文本，以便核查调试使用，索引比较大，一般不建议用户自行使用。

Routine	Required Header
CJZIndexer: : Export	<JZSearchAPI.h>

Return Value

Bool :成功为 true，否则为 false

Parameters

const char *sExportFile: 导出的文本文件名称

const char *sWordList: 数据词典目录下的 dictionary.wordlist 文件。

Remarks:

1. 处理失败后, 可以查看当前目录下的当天日志文件;
2. 一般不建议使用

Example

```
for (int doc_id=nDocID_Start;doc_id<vecFileList.size();doc_id++)

{ //索引一个文件

    pIndexer->MemIndexing(vecFileList[doc_id].c_str(),"title");//对文件标题
    建立索引

    pIndexer->FileIndexing(vecFileList[doc_id].c_str(),"content");//对文件
    内容建立索引

    pIndexer->IntIndexing(doc_id,"id");//对 ID 建立索引

    pIndexer->FloatIndexing(doc_id,"idf");//对 IDF 建立索引

    pIndexer->AddDoc();

}

pIndexer->Save();//保存结果

pIndexer->Export("Export0729.txt","E:\\Projects\\LJSearch\\dat\\dictionary
.wordlist");//导出索引文件的实际内容
```

4.3 检索接口

4.3.1 JZSearch_Init 精准搜索器初始化

```
JZSearchAPI_API bool JZSearch_Init(const char *pcIndexFile,const char
*sDictPath=0,const char
*sFieldInfoFile=0,int encoding=INDEX_ENCODING_GBK_FANTI));
```

检索系统初始化, 必须初始化后, 才能使用 CJZSearcher

Routine	Required Header
JZSearch_Init	<JZSearchAPI.h>

Return Value

Return true if init succeed. Otherwise return false.

Parameters

const char *pcIndexFile: 指定的索引文件名, 系统根据该文件名自动生成相应的数据文件, 检索时使用, 索引与检索的文件必须一致。

sDictPath: 词典文件所在位置; 为空时, 采用 n-gram 索引方法

sFieldInfoFile: 域字段信息, 用于支持多域索引, 为空则只支持一个字段, 参见 3.3 节 JZIndexer_FieldSave.

//encoding: 设置编码, 具体可以参见 1.5

Remarks

1. 搜索系统初始化后, 才能使用精准搜索类 CJZSearch;
2. 初始化失败的原因包括:
 - 1) 数据文件不齐全: sDictPath 需要齐备的文件包括 irreg2regularEnglish.map、dictionary.wordlist、dictionary.pdat、english.pdat
 - 2) 字段定义数据文件非法: 即不是系统自动生成的字段信息
 - 3) 授权过期
3. 失败时, 可以查看当前目录下以日期命名的 log 日志文件, 有具体的提示;

Example

```
JZSearch_Init("E:/语料库资源/Corpus/搜索语料库  
/index_ch/Test0807", "E:\\Projects\\LJSearch\\dat", "E:/语料库资源/Corpus/搜索语  
料库/index/FieldInfo.dat");
```

```
printf("TestGBKChinese JZSearch_Init OK!\n");
```

Output

4.3.2 JZSearch_Exit 精准搜索器系统退出

```
JZSearchAPI_API bool JZSearch_Exit();  
//系统退出
```


Routine	Required Header
JZSearch_Exit	<JZSearchAPI.h>

Return Value

Return true if init succeed. Otherwise return false.

Parameters

None

Remarks

2. 执行 JZSearch_Exit, 才能释放资源

Example

```
JZSearch_Exit();
```

Output

4.3.3 JZSearch_Reload 精准搜索器系统增量加载

```
JZSearchAPI_API bool JZSearch_Reload();
```

```
//系统增量更新后, 重新加载
```

Routine	Required Header
JZSearch_Reload	<JZSearchAPI.h>

Return Value

Return true if init succeed. Otherwise return false.

Parameters

None

Remarks

1. 增量索引完成后, 需要执行 JZSearch_Reload, 在不影响当前搜索服务的前提下提供新的搜索服务

Example

```
indexing("E:\\Projects\\LJSearch\\LJSearch\\TestDLL\\Corpus\\List.txt", vecFileList);
```

//增量索引，采用 CJZIndexer 建立索引的一个函数，其目的是建立指定文件列表内的索引文件

```
printf("indexing(\"\\List2.txt\" OK! Time=%f\\n", duration);
```

```
JZSearch_Reload();
```

//动态加载，进行新的搜索服务

4.3.4 JZSearch_Export 精准搜索器系统索引内容导出函数

```
JZSearchAPI_API bool JZSearch_Export(const char *sExportFile, const char *sWordList);  
//导出内部的数据，以便核查
```

```
JZSearchAPI_API bool JZSearch_Export(int nTermID, const char *sExportFile, const char *sWordList);
```

//导出内部的数据，以便核查

功能类似于 4.3.10 CJZIndexer::Export，内部调试使用，不推荐用户使用

Routine	Required Header
JZSearch_Export	<JZSearchAPI.h>

Return Value

Return true if init succeed. Otherwise return false.

Parameters

const char *sExportFile: 导出的文本文件名称

const char *sWordList: 数据词典目录下的 dictionary.wordlist 文件。

Remarks

2. 增量索引完成后，需要执行 JZSearch_Reload，在不影响当前搜索服务的前提下提供新的搜索服务

Example

```
indexing("E:\\Projects\\LJSearch\\LJSearch\\TestDLL\\Corpus\\List.txt", vecFileList);
```

//增量索引，采用 CJZIndexer 建立索引的一个函数，其目的是建立指定文件列表内的索引文件

```
printf("indexing(\"\\List2.txt\" OK! Time=%f\\n", duration);
```

```
JZSearch_Reload();
```

//动态加载，进行新的搜索服务

4.3.5 JZSearch_Merge 精准搜索器系统索引归并优化函数

```
JZSearchAPI_API bool JZSearch_Merge();
```

//系统自动优化，对索引信息进行归并处理，将多个小的索引文件合并为新的索引文件

Routine	Required Header
JZSearch_Reload	<JZSearchAPI.h>

Return Value

Return true if succeed. Otherwise return false.

Parameters

None

Remarks

1. 索引归并相对耗时，建议小的索引文件比较多时再根据一定的业务策略调用该函数；
2. 系统会自己进行优化处理，无须用户干预

Example

```
JZSearch_Rerge();
```

//系统进行归并

4.3.6 搜索结果的数据记录结构

```
typedef struct tRESULT_RECORD { //搜索结果结构，用于检索计算使用  
    int doc_id;
```

```

    int offset;//在域字段内的偏移量
    double score;//排序用的打分机制
}RESULT_RECORD;
typedef RESULT_RECORD * RESULT_RECORD_VECTOR;

```

搜索结果 XML 文件格式如下:

```

<?xml version="1.0" encoding="gb2312" standalone="yes" ?>
<LJSearch-Result>
<Result-Number>
<Total-Number>1</Total-Number>//符合条件的搜索结果总数
<Return-Number>1</Return-Number>//返回符合条件的搜索结果总数
<Start-Position>0</Start-Position>//结果集中的起始编号
</Result-Number>
<Result>
<Document>
<doc_id>0</doc_id>//字段名 doc_id 系统内部的 doc_id 编号, 由 AddDoc 函数顺序递增
<title>E:\Projects\LJSearch\LJSearch\TestDLL\Corpus1021\Corpus\Meteorological
Bulletin Automatic Generation based on Spatio-Temporal Reasoning.txt</title>//字段
名 title 由 3.2 FieldAdd 定义的需要检索出的字段
<content>Meteorological Bulletin Automatic Generation based on Spatio-Temporal
Reasoning Abstract Meteorological bulletin has more and more diversified, large scale,
highly integrated requirements and potential demands from whole society. The strong
professional efforts involved in transforming the variety of special meteorological
data to natural language text are becoming more challenging in providing sophisticated
and easily understood weather features. This paper presents a new Meteorological
bulletin automatic generation method based on spatio-temporal reasoning. To enhance
an exact and non-redundant description for complex meteorological data, and for
special future tendency dynamics in emerged interesting areas.</content>
<id>0</id>
<idf>0.000000</idf>
</Document>
</LJSearch-Result>

```

4.3.7 CJZSearcher 精准搜索器类

```

class JZSearchAPI_API CJZSearcher{
public:
    CJZSearcher(int sort_type=SORT_TYPE_DOCID);
    ~CJZSearcher(void);
    // TODO: add your methods here.

    void Search(const char *query_line,int nStart,int nPageCount,const char
*sResultName=0);

```

```
//query_line: 查询表达式
//nStart:记录起始地址
//nPageCount: 当前页返回结果数目
//nPageCount=-1:当前页需要返回所有的结果数目
//sResultName: 结果存储的 XML 地址

const RESULT_RECORD_VECTOR Search(const char *query_line,int
*p_nResultCountRet);
//query_line: 查询表达式
//p_nResultCountRet:搜索结果总数

bool DocDelete(int doc_id);
private://以下部分为系统使用，应用开发者不要改写，只能读取数据
int m_nHandle;
int m_nSortType;//排序方法编号

char *m_pResultBuf;
};
```

Routine	Required Header
JZIndexer_Exit	<JZSearchAPI.h>

Return Value

None

Parameters

None

Remarks

1. 索引器类，可以并行多线程处理

Example

Output

4.3.7.1 CJZSearcher::CJZSearcher 精准索引器类构造函数

```
CJZSearcher(int sort_type=SORT_TYPE_DOCID);
```

Routine	Required Header
CJZSearcher: : CJZSearcher	<JZSearchAPI.h>

Return Value

None

Parameters

int sort_type=SORT_TYPE_DOCID: 设定搜索的排序方式

Remarks:

- 1.系统异常时，可以查看当前目录下以日期命名的 log 日志文件，有具体的提示；
2. 必须调用 JZSearch_Init, 系统可以同时实例化多个 CJZSearcher, 采用不同的排序方式；

Example

```
JZSearch_Init("./index/Test0809", "E:\\Projects\\LJSearch\\dat", "./index/FieldInfo.dat");
```

```
printf("TestGBKChinese JZSearch_Init OK!\n");
```

```
CJZSearcher *pSearcher=new CJZSearcher(SORT_TYPE_DOCID); //按照相关度排序
```

```
printf("TestGBKChinese new CJZSearcher OK!\n");
```

Output

4.3.7.2 CJZSearcher::Search 精准索引器类搜索函数

```
void Search(const char *query_line, int nStart, int nPageCount, const char *sResultName=0);
```

根据查询表达式，生成搜索结果文件

Routine	Required Header
CJZSearcher: : Search	<JZSearchAPI.h>

Return Value

None

Parameters

query_line: 查询表达式, 需要符合搜索语法, 参见 6、JZSearch 检索语法
nStart: 记录起始地址
nPageCount: 当前页返回结果数目, -1 时, 需要返回所有的结果数目
sResultName: 结果存储的 XML 文件地址

Remarks:

1. 系统异常时, 返回结果为空, 可以查看当前目录下以日期命名的 log 日志文件, 有具体的提示;

Example

```
strcpy(sLine, "[FIELD] title [AND] 解放军");//甲型 H1N1 流感
```

```
pSearcher->Search(sLine, 0, -1, "Result.xml");//搜索 title 字段, 将包含“解放军”  
的所有记录保存到 Result.xml 文件中。
```

Output

4.3.7.3 CJZSearcher::Search 精准索引器类搜索函数

```
const RESULT_RECORD_VECTOR Search(const char *query_line, int  
*p_nResultCountRet);
```

类似于 5.8.2, 不同的是, 该函数返回的是结果记录集数组, 而不是文件本身, 开发者可以根据返回的 doc_id 进行进一步的搜索处理

Routine	Required Header
CJZSearcher: : Search	<JZSearchAPI.h>

Return Value

None

Parameters

query_line: 查询表达式
p_nResultCountRet: 搜索结果总数存储的指针

Remarks:

1. 系统异常时, 返回结果为空, 可以查看当前目录下以日期命名的 log 日志文件, 有具体的提示;

Example

```
strcpy(sLine, "[FIELD] title [AND] 解放军");//甲型 H1N1 流感
```

```
int nResultCount=0;
```

```
RESULT_RECORD_VECTOR pResult =pSearcher->Search(sLine,&nResultCount);//搜索  
title 字段，将包含“解放军”的所有记录保存到 pResult 数组中。
```

Output

4.3.7.4 CJZSearcher::DocDelete 精准索引器类索引文档删除函数

```
bool DocDelete(int doc_id);
```

删除内部编号为 doc_id 的文档

Routine	Required Header
CJZSearcher: : DocDelete	<JZSearchAPI.h>

Return Value

bool

Parameters

int doc_id; 内部编号参数

Remarks:

- 1.系统异常时，返回结果为空，可以查看当前目录下以日期命名的 log 日志文件，有具体的提示；
2. doc_id 是系统内部字段维护的，由 AddDoc 实现该字段的自动维护
- 3.调用该函数之前，一般都需要事先检索，并获取检索记录中的 doc_id;

Example

```
pSearcher->DocDelete(3908);//删除 3908 号文档
```

```
pSearcher->DocDelete(2110);//删除 2110 号文档
```

Output

4.4 利用 JZSearch 开发程序，搭建搜索引擎服务指南

搭建搜索引擎服务的步骤（建议尽量采用通用篇：五步配置搭建搜索服务）：

步骤	处理内容	参见
建立索引过程		
Step 1	定义字段索引信息	参见 3.2 JZIndexer_FieldAdd
Step 2	保存字段索引信息，	参见 3.3 JZIndexer_FieldSave
Step 3	索引器初始化	参见 4.1 JZIndexer_Init 精准索引器初始化
Step 4	建立索引器类	4.3.1 CJZIndexer::CJZIndexer
Step 5	扫描数据库或者文件中的文档，依次针对每篇文档建立索引	参见 4.3.2~4.3.6;
Step 6	处理完成文档，调用 AddDoc	4.3.7
Step 7	索引完毕，调用 Save	4.3.8
建立搜索后台服务过程		
Step 1	按照已索引的内容搭建搜索服务	参见第 7 章
建立搜索前段服务过程		
Step 1	按照客户端，选择 C 或者 Java 开发客户端程序	参见第 8 章
增量索引过程		
Step 1	完全参照第一次建立索引的过程，注意增量索引文件必须和第一次索引文件名一致	参见 建立索引过程
Step 2	通过客户端命令行方式	参见 3.3 JZIndexer_FieldSave

4.5 索引数据辅助分析

Bin 目录下的部分高级调试功能

exportIndex.bat: 导出各个索引词对应的文档数目, 可以辅助我们分析索引词的质量, 及我们索引文档的词语宏观统计特点。用户也可以利用类似的命令做各种分析。**exportIndex.bat** 的其格式如下:

JZIndexer.exe e 【索引带路径的文件名】【字段带路径的文件名】【词典所在的路径】【词典 wordlist 的路径与文件名】 【导出的文件名】i

注意: 如果在 Linux 下, 需要将 **JZIndexer.exe** 改成 **.JZIndex**

五、案例篇

JZSearch 已应用到了多个海量搜索系统中, 目前已经应用于中国邮政搜索引擎、河北省标准搜索引擎、中国对外承包工程商会的知识搜、富基融通 (纳斯达克上市公司: EFUT) 商品搜索、北京市网控办的 WBK 微博监控系统中。

5.1 中国邮政集团名址信息中心首页的邮址垂直搜索

访问地址为: www.cpdn.com.cn 其中邮编搜索、企事业单位查询的内核均由 JZSearch 实现, 目前每年的访问量超过 3000 万。



图 1 基于 JZSearch 的中国邮址搜索引擎展示效果

5.2 河北标准化研究院的标准搜索

该系统的搜索部分均由 JZSearch 对其国内外标准数据库及全文

文件库进行索引，提供全省上万家企事业单位与政府机构搜索，搜索段为 C/S 模式。



图 2 基于 JZSearch 的标准文献垂直搜索展示效果

5.3 中国对外承包工程商会的知识搜索门户

该系统主要采用 JZSearch 搜索对中国对外承包商会内部 100 台办公电脑的文档进行索引，建立内网知识搜索门户，目前已经稳定运行 2 年多。



图 3 基于 JZSearch 的中国对外承包工程商会的知识搜索

5.4 富基融通的商品比价搜索

该系统主要采用 JZSearch 搜索对庞大的商品库进行同一性判别，并提供比价搜索功能，每日新增数据 1000 万，同时支持在线搜索 1 万并发。该系统已经内嵌到其 MyStore 项目中。



图 4 基于 JZSearch 的富基融通商品比价搜索系统

5.5 WBK 微博人物搜索

WBK 为北京市网控办的微博监控系统，内置的微博搜索与人物搜索由 JZSearch 支持，搜索效果如下：

检索条件						
搜索内容:		民主			检索	更多
性别: 全部	地址:	微博名:	真实姓名:	教育:	工作:	
ID	姓名	性别	地址	操作		
<input type="checkbox"/> 1258463770	时寒冰	男	上海,浦东新区	加关注		
<input type="checkbox"/> 1891247651	民主中国-v_81s	女		加关注		
<input type="checkbox"/> 1761492784	天使城杰夫	男	海外,美国	加关注		
<input type="checkbox"/> 1233488674	王才亮	男	北京,西城区	加关注		
<input type="checkbox"/> 1879326192	全宗锦	男	北京,昌平区	加关注		
<input type="checkbox"/> 1921147133	龙光大宝贝	男	辽宁,沈阳	加关注		
<input type="checkbox"/> 1737963651	田维诗	女		加关注		
<input type="checkbox"/> 1407792312	梁香禄	男	广东,广州	加关注		
<input type="checkbox"/> 1260173284	玄泰HyunTae	男	湖南,长沙	加关注		
<input type="checkbox"/> 1160673743	乔若熙_Sabrina	女		加关注		
<input type="checkbox"/> 1243043474	像黄琪的铃	男	江苏,泰州	加关注		
<input type="checkbox"/> 2246703904	大眼跟微公益	男	甘肃,兰州	加关注		

图 5 基于 JZSearch 的微博人物搜索系统

六、问答篇：FAQ 及小技巧

6.1：环境问题

6.1.1 JZSearch 支持 Linux 吗？

支持 Windows, Linux 等各类操作系统, 目前有 Win7, WinXP 版本, 及 Linux 64 位版本, 如需其他环境, 可以直接登录 www.nlpir.org 寻求技术支持。

6.1.2 Linux 环境使用太不方便, 有什么技巧？

字段数据生成、索引生成均可以在 Windows 环境下通过可视化界面实现, 生成的数据在 Windows 和 Linux 都是兼容的。

6.2：索引问题

6.2.1 在 Windows 下, 针对 MySQL 数据库的索引老是创建不成功, 访问不了数据库, 为什么？

Windows 下为了兼容各个数据库, JZSearch 采用了 ODBC 访问数据库, MySQL 数据库的索引不成功, 往往是本地没有安装 ODBC driver, 可以访问 MySQL 官网下载, 下载地址为: <http://www.mysql.com/downloads/connector/odbc/>。

6.2.2 采用 bigfile 字段, 为什么？

6.2.3 数据库增删改如何适应, 为什么？

6.3：搜索问题

6.3.1 老是搜索不到结果或者搜索到的结果老是不变的, 怎么回事？

搜索不到结果, 主要原因包括:

- 1、搜索服务没有启动，或者客户端和服务端的 IP 及端口不匹配；
- 2、搜索语法表达式有问题，如：**field** 等关键字拼写错误，或者索引字段名称拼写错误。详细情况可以查看搜索服务目录下的日志文件，有详细的错误说明。
- 3、搜索内容确实不存在搜索结果，建议换成通用的搜索词实验。

七、作者篇



张华平 博士 副教授 硕导

北京理工大学大数据搜索与挖掘实验室 主任

地址：北京海淀区中关村南大街 5 号 100081

电话：+86-10-68918642

Email: kevinzhang@bit.edu.cn

MSN: pipy_zhang@msn.com;

网站: <http://www.nlpir.org> (自然语言处理与信息检索共享平台)

博客: <http://hi.baidu.com/drkevinzhang/>

微博: <http://www.weibo.com/drkevinzhang/>

Dr. Kevin Zhang (张华平, Zhang Hua-Ping)

Associate Professor, Graduate Supervisor

Director, Big Data Search and Mining Lab.

Beijing Institute of Technology

Add: No. 5, South St., Zhongguancun, Haidian District, Beijing, P. R. C PC: 100081

Tel: +86-10-68918642

Email: kevinzhang@bit.edu.cn

MSN: pipy_zhang@msn.com;

Website: <http://www.nlpir.org> (Natural Language Processing and Information Retrieval Sharing Platform)

Blog: <http://hi.baidu.com/drkevinzhang/>

Twitter: <http://www.weibo.com/drkevinzhang/>



自然语言处理与信息检索共享平台
Natural Language Processing & Information Retrieval Sharing Platform