# Project 1: Relational Database System: Design, Implementation and Query Processing of Wikidata

## 项目基本流程

1. 阅读相关网页和文档，了解wikidata的数据模型结构及其存储文件json的文件格式如何进行读取操作的；
2. 设计数据库表结构；
3. 对数据进行抽取，存放在数据库中；
4. 对数据库进行优化处理，提高搜索效率；
5. 实现一个web搜索系统，能够满足如下要求：
6. 给定一个名称, 返回与该名称匹配的所有实体；
7. 给定一个实体名称，返回它所属的类别（实例和子类）；
8. 给定一个实体名称，返回与此实体共同发生的实体；
9. 给定一个实体，返回它所拥有的所有属性；
10. 设计并实现一个基本的问答系统，例如，如果我问中国的人口是多少时，它应该返回正确的答案。
11. 撰写过程报告及演示PPT。

## 项目实施工具

1. OS: Win 10；
2. IDE: MyEclipse 2016CI
3. Database : MySQL 5.6
4. Web Service : Tomcat 8.0

## 数据结构及文件存储结构

图1，wikidata数据结构图

如上图所示为wikidata的数据结构示意图。

要想将json数据文件导入到数据库中，首先需要清楚wikidata数据是以何种数据结构存储在json文件中的以及json文件格式的数据文件应该以何种读取，这关系到后继的json文件读取和数据库表结构的建立。

阅读相关维基API数据模型，其数据结构可简述为：

每一个词条对应于多个entity，每个entity都有一个唯一编号ID字段用来区分和查找各个entity；每个entity又可分为两种类型：item和property，item编号一般以Q打头，property编号一般以P打头，item一般是概念实体型词条，property一般是修饰描述性词条；labels为该实体的标签，也就是名称；aliases为该实体除了labels的其他的标签，也就是别称；descriptions是对该实体的描述，通常是以几句简短概述性的语言对该实体进行描述；claims中包含了该实体的各种属性的描述，比如对一个名人有他的出生日期的描述，国籍属性的描述等等，实体的主要信息和结构就是以上这些部分，还有一些关于实体及其属性的描述的结构过于细化在这里就不再叙述了。另外由于wikipedia是面向全球的，这就是说它的每项键值对是以多种语言存储在json文件中的。在本项目中，考虑项目的简洁性和代表性，只对多个键值对中的英文和中文键值对进行抽取处理，并且对于关于实体某些枝节没有做抽取处理，如sitelinks（该实体的网页链接），modified（该实体的修改世间）以及claims中的qualifiers（对该属性做出限定）和references（该属性值的参考来源）。

关于json文件格式，引用其官网上的描述：

JSON（JavaScript Object Notation）是一种轻量级的数据交换格式。它使人类阅读和写作都变得很容易。同时，机器解析和生成json文件也很容易。它是基于JavaScript（标准ECMA-262第3版 - 1999年12月）编程语言的一个子集 。JSON是完全独立于语言的文本格式，但是使用C语言家族的程序员熟悉的约定，包括C，C ++，C＃，Java，JavaScript，Perl，Python等等。这些属性使JSON成为理想的数据交换语言。

JSON建立在两个结构上：

名称/值对的集合。在各种语言中，被实现为对象，记录，结构，字典，散列表，键控列表或关联数组。

有序的值列表。在大多数语言中，这是作为数组，矢量，列表或序列来实现的[1]。

通俗来说就是json文件是以键值对来存储数据的，包含在{}内，用数组来存放某个具有多个值的属性列表,包含在[]中。对wikidata数据结构json文件存储格式有了较好的了解之后，后面的工作才能顺利地进行下去。

Json文件格式如下图所示：

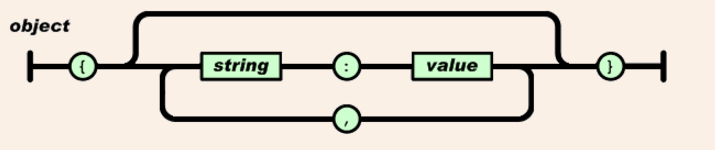


图2，json键值对结构示意图

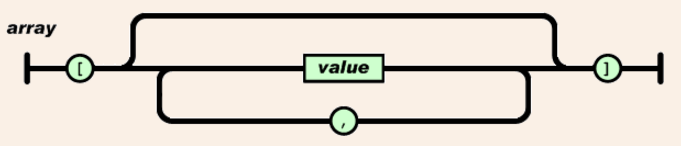


图3，json数组结构示意图

## 数据库表设计

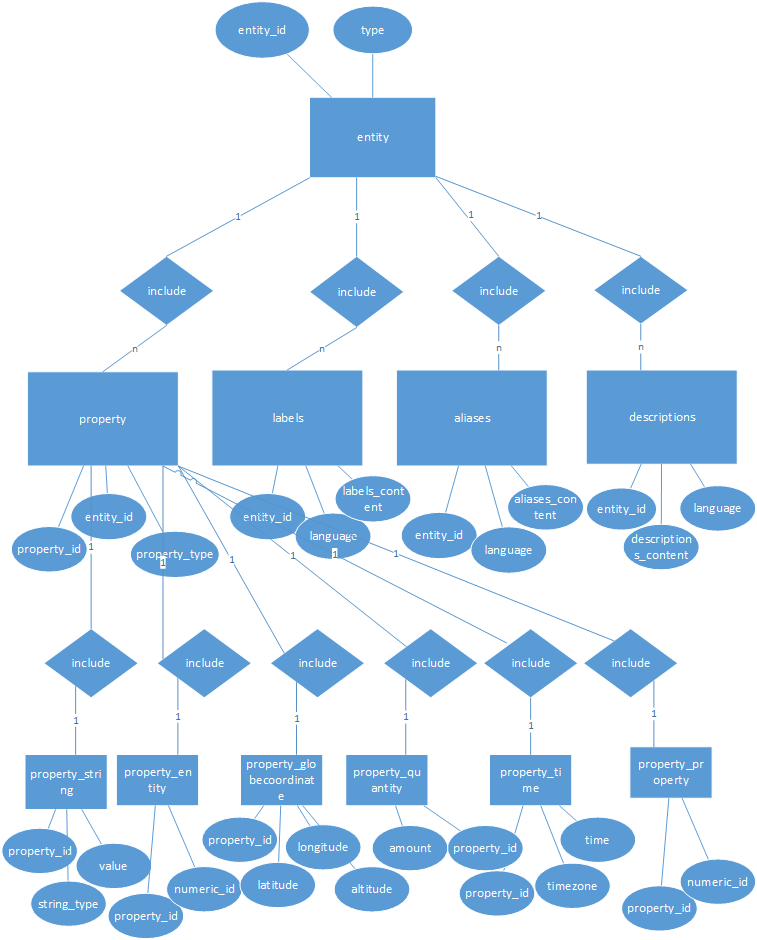
数据表的设计依据关系数据库表的设计规范，表结构如下图所示：

图4，数据库表设计图

## 数据抽取

数据抽取是把json文件中的数据读取转存到数据表中，读取程序设计图如下所示：

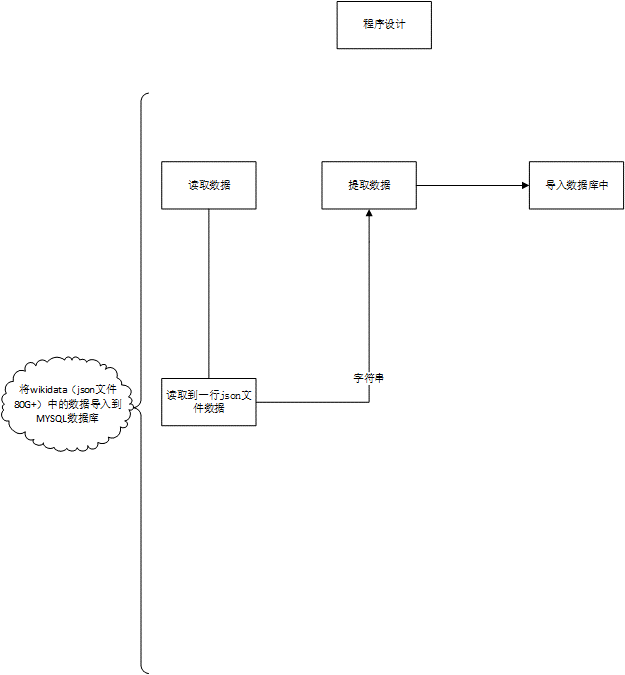
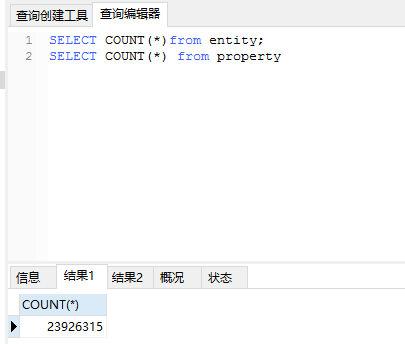
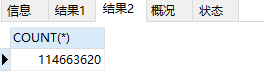


图5，数据抽取示意图

抽取过程花费6.5h，共有23926315条词条数据导入到数据库中。过程中需要注意的是在读取json文件的属性字段时候，需要判断该字段是否存在，判断存在再提取，如果直接提取，而该字段在该实体中未存在，程序就会报错；另外需要注意的是，为了提高插入数据库的效率，不要每提取一条就访问数据库插入一条数据，而是首先读取多条数据存储到容器中，再统一插入到数据库中，这样可以大大提高插入的效率。

最后插入数据库中的词条数如下图所示：

图6，数据库数据数量图

## 数据库表结构优化

本项目中，对表结构的化优的主要目的是简化表的结构和提高表的查询效率。为此我们把property属性下的各个表做了合并，合并成了一个表property。如图所示：

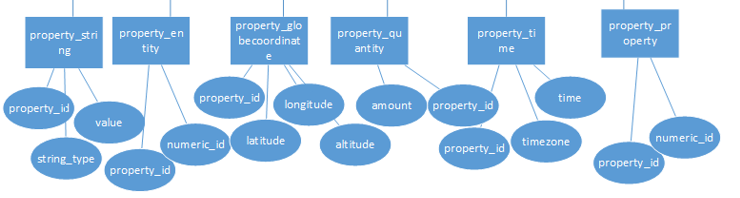


图7 表的合并

Property表的字段由3个字段property\_id，entity\_id，property\_type扩展为property\_id，entity\_id，valueType，value，把它下分的各个表property\_string，property\_entity，property\_globecoordinate，property\_quantity，property\_time和property\_property简化一起合并存储在表property中，如在property\_string中，property\_id 继续存放在property\_id中，string\_type存放在valueType中，value存入value中，在property\_entity表中，numeric\_id存入到value中，其它相应的值存放在新表的相应的字段中，然后在property\_globecoordinate表中，字段latitude，longitude，altitude一起以一个值存放在新表property的value字段中，对于property\_time表中的timezone字段我们进行了省略，property\_property依照相应字段存储在新表中。这样避免了在进行查询操作时需要事先判断property的具体属性在那个表中，简化了表的结构。

为提高查询效率，我们在各个表中的需要进行查询的字段上建立了相关的B+树索引。设计实验比较建立索引之后的查询效率的提高:

新建一个数据库wikidata2，将原数据库中的一个表导入到该新数据库中，两个表的数据都相同，两者之间的差别就是原数据库中的表有索引，新建的数据库中表没有索引。执行同一条SQL查询语句，对比建立索引与未建立索引在查询效率上的差距。

在建立索引的表中执行一条SQL查询语句，如图所示：

图8，在建立索引的表中进行查询的结果

查询时间只用了0.030s。

再看再未建立索引的表中执行相同的SQL查询语句时的情况，如图所示：

图9，在未建立索引的表中进行查询的结果

查询时间为17.370s，是建立索引所用时间的579倍。更换查询语句，在两个表中分别进行10次查询，记录查询时间如下表所示：

|  |  |  |
| --- | --- | --- |
| 序号 | 已建索引 | 未建索引 |
| 1 | 0.029s | 12.526s |
| 2 | 0.031s | 17.213s |
| 3 | 0.069s | 15.231s |
| 4 | 0.035s | 17.875s |
| 5 | 0.046s | 14.235s |
| 6 | 0.087s | 13.432s |
| 7 | 0.067s | 20.214s |
| 8 | 0.030s | 19.543s |
| 9 | 0.022s | 17.740s |
| 10 | 0.034s | 16.841s |

表1，10次对照实验查询时间表

可以看到在建立索引的表中进行查询的操作的效率远远大于在未建立索引的表中进行查询的效率，10次实验下来，在未建立索引的表中查询的时间是在建立索引的表中进行查询的时间的443.7倍。

## Web搜索系统的实现

仿照Google搜索界面制作一个wikidata数据库的搜索系统，程序结构设计示意图：

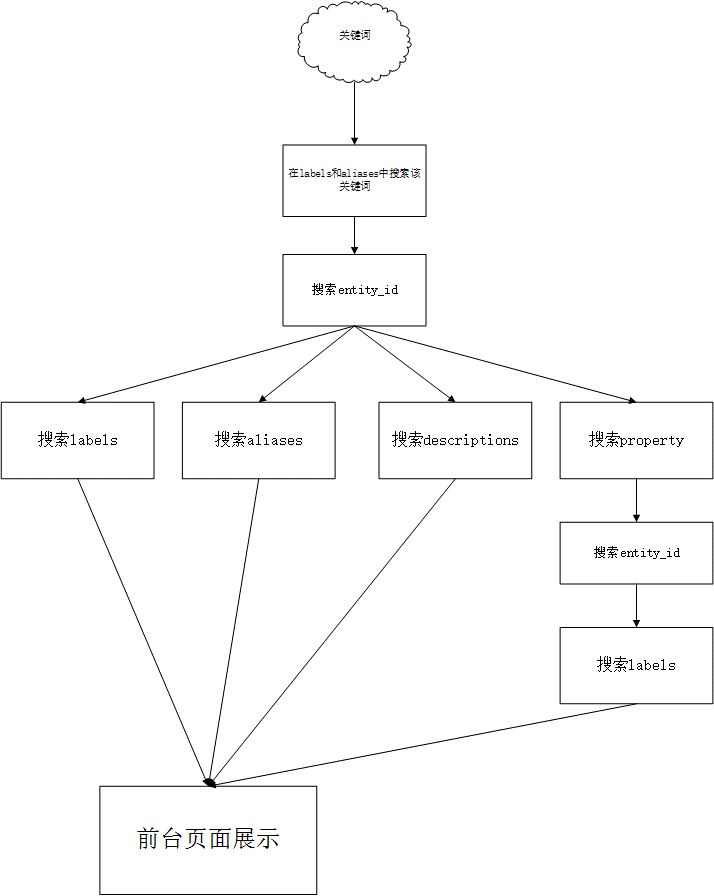


图10，搜索界面程序设计示意图

首先通过关键词在label表中和aliases表中对关键词进行匹配搜索，得到一个匹配此关键词的实体号，通过这些实体号的集合再在labels，aliases，descriptions和property表中进行搜索，最后将结果集展现在网页中。

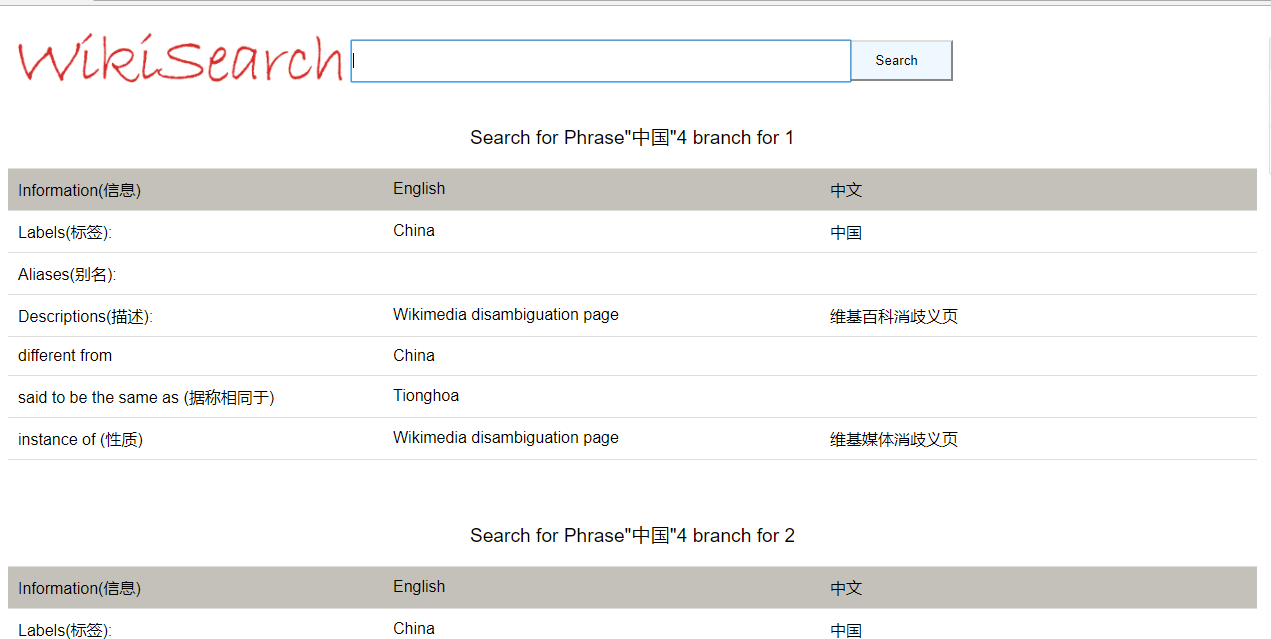




图11，搜索结果展示

另外，如果想要搜索某个词条的某个属性时，比如，想要知道“中国的人口是多少时”，只需在搜索框中键入关键词“中国 人口”即可，如图所示：

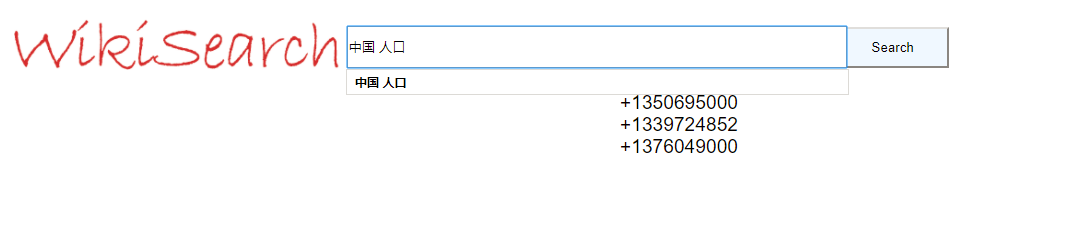


图12，词条+属性搜索结果展示

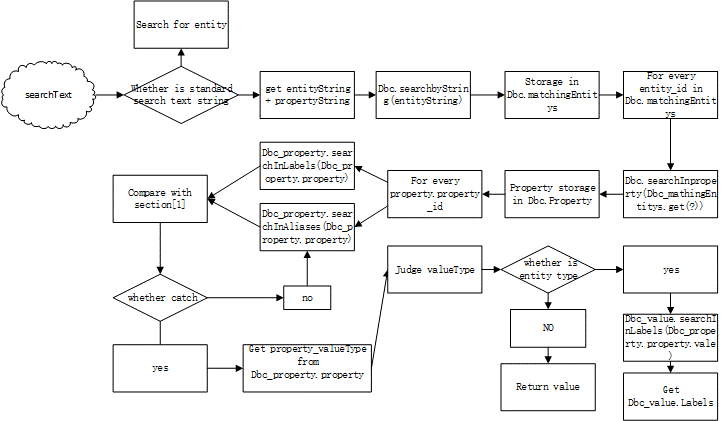
该搜索程序的设计图如下图所示：

图13，搜索词条+属性程序设计示意图

## 总结

总体上完成了项目开发的要求，各项要求都基本得到实现，但在细节上还是有不足的地方，这是由于自身技术实力的限制和时间仓促等方面的因素造成的，如搜索到多个实体时，未引入排名（Rank）属性，导致同名的实体并未按照被搜索的频率先后出现在用户搜索的界面上，另外由于未引入属性的限定（Qualifiers）属性，导致搜索某个实体的某个属性时，出现多条结果而没有相关限定词的修饰，一定程度上对用户不友好，如搜索中国的人口时，出现了三条记录是关于中国的人口的，这三条记录是按照时间的先后进行统计的，比如说第一条可能是近几年的统计结果，后面几条可能是前些年的中国的人口统计，但在向用户进行展示时并未做足够的声明，这是因为在本项目的开发系统时在由于省略了一些字段而造成的。

# 参考及引用

## 引用：

[1] : <http://www.json.org/> ．

### 编程参考

[1] : <http://blog.csdn.net/dong945221578/article/details/45198687>．

[2] : <http://blog.csdn.net/eengel/article/details/47261125>．