

SHANGHAIUNIVERSITY

毕业设计（论文）

**UNDERGRADUATEPROJECT(THESIS)**

**题目:面向食品领域的数据分析和存储**

|  |  |
| --- | --- |
| **学院** | **计算机工程与科学学院** |
| **专业** | **计算机科学与技术** |
| **学号** | **x** |
| **学生姓名** | **x** |
| **指导教师** | **x** |
| **起讫日期** | **2018.01.29 – 2018.06.08** |

目录

[摘要 III](#_Toc515291278)

[ABSTRACT IV](#_Toc515291279)

[第1章 绪论 1](#_Toc515291280)

[§1.1 数据分析和存储的背景及意义 1](#_Toc515291281)

[§1.2 数据领域现状及存在的问题 1](#_Toc515291282)

[§1.2.1 数据分析的现状和存在的问题 1](#_Toc515291283)

[§1.2.2 数据存储的现状和存在的问题 1](#_Toc515291284)

[§1.2.3 研究背景 2](#_Toc515291285)

[§1.2.4 研究内容和目标 2](#_Toc515291286)

[§1.3 本文组织结构 2](#_Toc515291287)

[第2章 数据分析 3](#_Toc515291288)

[§2.1 源数据分析概述 3](#_Toc515291289)

[§2.2 针对不同类型数据的分析方法 3](#_Toc515291290)

[§2.2.1 Excel和txt数据 3](#_Toc515291291)

[§2.2.2 XML和json格式 3](#_Toc515291292)

[§2.2.3 Excel（txt）格式转XML（json） 4](#_Toc515291293)

[§2.3 本章小结 6](#_Toc515291294)

[第3章 数据存储方案 7](#_Toc515291295)

[§3.1 多种数据库概述 7](#_Toc515291296)

[§3.2 关系型数据库 7](#_Toc515291297)

[§3.2.1 概述 7](#_Toc515291298)

[§3.2.2 关系型数据库实例之MySQL 8](#_Toc515291299)

[§3.2.3 关系型数据库实例之PostgreSQL 9](#_Toc515291300)

[§3.2.4 MySQL和PostgreSQL的性能对比测试 10](#_Toc515291301)

[§3.3 对象型数据库 12](#_Toc515291302)

[§3.3.1 概述 12](#_Toc515291303)

[§3.3.2 对象型数据库实例之Db4o 13](#_Toc515291304)

[§3.4 文档型数据库 15](#_Toc515291305)

[§3.4.1 概述 15](#_Toc515291306)

[§3.4.2 文档型数据库实例之Mongodb 16](#_Toc515291307)

[§3.5 图数据库 18](#_Toc515291308)

[§3.5.1 概述 18](#_Toc515291309)

[§3.5.2 图数据库实例之neo4j 20](#_Toc515291310)

[§3.6 不同数据库的性能对比 21](#_Toc515291311)

[§3.7 本章小结 22](#_Toc515291312)

[第4章 食物项目系统的设计与实现 23](#_Toc515291313)

[§4.1 系统整体设计 23](#_Toc515291314)

[§4.1.1 开发环境简介 23](#_Toc515291315)

[§4.1.2 系统整体架构 23](#_Toc515291316)

[§4.2 数据分析 23](#_Toc515291317)

[§4.3 数据库设计 24](#_Toc515291318)

[§4.3.1 Mysql设计 24](#_Toc515291319)

[§4.3.2 Neo4j设计 25](#_Toc515291320)

[§4.3.3 ProGRESQL设计 25](#_Toc515291321)

[§4.3.4 Db4o设计 25](#_Toc515291322)

[§4.3.5 MongoDB设计 25](#_Toc515291323)

[§4.4 功能模块设计 25](#_Toc515291324)

[§4.4.1 数据上传 25](#_Toc515291325)

[§4.4.2 数据显示和操作 26](#_Toc515291326)

[§4.5 本章小结 29](#_Toc515291327)

[第5章 总结与展望 30](#_Toc515291328)

[§5.1 本文总结 30](#_Toc515291329)

[§5.1.1 本文的主要工作 30](#_Toc515291330)

[§5.1.2 本文的主要创新点 30](#_Toc515291331)

[§5.2 尚未解决的问题 30](#_Toc515291332)

[§5.3 展望 30](#_Toc515291333)

[致谢 31](#_Toc515291334)

[参考文献 32](#_Toc515291335)

面向食品领域的数据分析和存储

# 摘要

数据领域在过去的几十年取得了巨大的进步，然而它的变化也给了我们很多挑战。如何整合异构数据，管理多种多样的数据类型，实现对复杂对象的完美建模都是我们面临的挑战和问题。

本文我们将结合食品领域的具体场景，进行异构数据的转换，将excel/txt文件转换为xml/json格式，之后我们会分析多种类型的数据库并使用每个类型经典的数据库管理系统进行建模和数据操作，过程中我们将从底层分析数据库系统的结构和特点，并结合模拟数据进行性能分析。最后我们将总结上述理论，描述包含了数据分析和数据存储流程的面向食品领域系统，列出其架构，功能并给出截图。

关键词：数据提取，异构数据转换，数据存储，数据建模，关联数据

Data Analysis and Storage for Food Domain

# ABSTRACT

With the rapid development of information field, great changes have been taken place in data filed these years. This brings many challenges. The way of integrating the heterogeneous data, managing a variety of data types, achieving a perfect model for a complex object are all the problems we are urgent to solve.

In this paper, we will do some research combining the scene of food field. Firstly, we will do the conversion of heterogeneous data, converting excel/txt to xml/JSON format. Then, we will analyze a variety of types of database and use each type’s classic database management system for modeling and data operation. In this process, we will analysis the underlying data structure and characteristics. Also, the test of performing analysis with simulation data is provided. After summarizing all the mentioned theory, I will describe the system which includes the data analysis and data storage process. System architecture, functions and screenshot are given in this part.

**Keywords:** Data extraction, heterogeneous data conversion, data storage, data modeling, Relational data

# 绪论

本章主要描述了数据分析和存储的背景，意义。分析了相关课题的研究现状，提出了本文要研究的重点内容和目标。

## 数据分析和存储的背景及意义

随着数据库技术不断更新换代，计算机信息数据不断变化，数据的重要程度与日俱增，对数据分析和存储的要求也越来越高。在当前社会中，上至企业，学校，下至个人都在进行信息化发展。而使用数据库技术可以更方便的管理数据，可以保持数据信息的一致性，完整性，低冗余性，甚至可以增加数据的智能性。在这样的背景下，解决对各种类型的源数据的整合提取，将数据用合适的结构存储记录的问题有着重要的意义。

## 数据领域现状及存在的问题

### 数据分析的现状和存在的问题

数据管理系统不断发展和变化，日常生活中人们用以存储和操作数据的工具或应用各有不同，再加上其他技术性问题，数据共享是常见议题，实际项目中异构数据的存在现象十分广泛。而无论是数据挖掘，还是人工智能建模，再或者应用系统使用，在工作流里面，第一步都是接入数据，如果在第一步接入数据中，数据结构乱七八糟，那么数据的处理将遇到很多麻烦。因此我们需要先整理这些碎片化的数据，即先进行数据分析（解析）。国外对异构数据转换研究开始的比国内早，他们目前已经在异构数据库集成，数据转换规则等上面有了较成熟的理论及技术。国内也已经开始了一些成功产品以方便数据的共享和转换，我在研究过程中简单借鉴他们的思路。

### 数据存储的现状和存在的问题

信息领域在近百年内飞速发展，近几年的数据又发生了巨大变化。这些变化体现在数据类型越来越多，数据结构越来越复杂，数据量越来越大。最开始的数据类型只有字符串和数字，现在又出现了图形，多媒体，cube，时间序列，流数据等类型。同时，相比以前的结构化数据，现在半结构或无结构数据越来越多，数据结构越来越复杂，对复杂数据的建模难度大大提高。而且随着数字化技术的发展，我们目前获得的数据以TB甚至PB量级迅速增长。数据显示，2010年全球存储的信息总量超过1.2ZB，2011年则继续增长了0.6ZB，这种增长还在加速，根据IDC2011年发布的宇宙数据研究显示，全球信息总量每过两年，就会增长一倍，按照目前的速度下去，2025年将达到35ZB。这也促使了大数据概念的兴起。然而，这样海量的数据使得原有的架构不能完全支撑，存储，处理功能都受到很严重的影响。大量的数据在不断产生，而它们的复杂程度，类型多样性同样也来也大，越来越多。如何解决这样问题，实现对数据的有效存储，高效访问，方便共享和安全控制等都是信息时代非常重要的问题。

### 研究背景

本文的研究工作基于一个食品方面的项目，选取该项目中的一个分支，即关于方便面口味研究的子项目，作为本文的基础数据和系统原型实现。

本文我们将假设一个食品领域的场景，场景中有若干具体的示例，如：有品牌A,B,C生产了口味a,b,c,d等n种方便面，每种方便面有自己表层物品属性，如编号，生产日期，规格，风味等。厂商为了更好的分析食品口味，研究市场对口味的喜好，请了m名测试员从色泽，光滑性，复水性，韧性，黏性，耐泡性等多个角度进行打分。同时，在口味方面，厂商进行了细化研究，通过多种评价标准，如强度等级，麻度等级，斯科维尔指数，描述词，根据方便面在不同时间的表现进行评级，得出了味道的时间强度评价和时间频率记录。当食物投放市场后，厂商进行了反馈调查，拿到了q位用户针对n种方便面的喜恶数据。我们需要根据如上场景为厂商提供一套或多套解决方案思路，方便它们在如上场景中方便的存储和管理数据，同时保证数据处理的高效率和安全性。此为研究背景。

### 研究内容和目标

我们最终可以为厂商提供一个系统，简称为食物领域系统，可以方便它们上传文件，存储数据。为了使用方便，这个系统应该还包含简单的数据可视化界面以进行数据操作。本文研究目标包括以下几个方面。一：如何存储厂商提供的原始数据（主要是excel,txt或图片格式），并将其转换为方便进行下一步处理的统一格式。二：如何为原始数据选择合适的数据结构和数据库管理工具进行存储和管理（如何存储复杂对象；当场景不断变化时，应该如何灵活适应。）。三：为厂商提供一些简易，实用的工具，以方便非IT人士管理数据。如数据可视化，数据查找，数据管理等。

## 本文组织结构

以下的论文我们将主要探讨前两个目标问题。第二章对应数据分析问题，提出了我的一些Excel转XML想法。第三章对应数据存储问题，通过对市场上常见的几种数据库思想及其典型数据库工具的对比分析，得出不同的建模思路，对该场景下不同的示例进行了建模分析，最后还进行了常见数据操作的性能对比。第四章是系统的架构，界面和功能展示，同时给出了针对第三个目标的答案。第五章是总结篇，提出了我的结论，同时抛出了本人力不能逮，尚未解决的问题。

# 数据分析

本章探讨数据分析的方法：给出本文中数据分析的概念定义，并引出针对不同类型数据的处理方法。

## 源数据分析概述

需要提前说明的是，我们这部分指的数据分析不是当前火爆的从大量原始数据中进行分析，得出未知信息的数据分析，而是指对不同格式不同类型的源数据进行处理，使其格式统一化，方便后期数据存储处理。也许用数据提取或数据交换形容会更贴切些。在目前大部分针对普通用户的项目场景下，录入的数据格式多种多样，这些不同格式，不同结构的数据也被称为异构数据，异构数据源间的数据集成和转换过程是非常必要的。只有对这些数据进行良好的合并整理，才能使得碎片数据发挥应用的作用。

这里我们简单考虑几种原始数据格式的处理情况，例如excel, txt, xml, json. 它们都有什么特性，分别应该如何处理？什么类型的数据在项目中出现的最多？当它们同时存在时，我们应该统一化为什么形式？

## 针对不同类型数据的分析方法

### Excel和txt数据

Microsoft Excel是微软公司发开的一款电子表格软件。它具有直观的界面、丰富的功能模块和图表工具，简单易用，适用于非IT人员对数据的准备，编辑和存储，这些使它成为最流行的PC端数据处理软件。但与专业的存储数据库相比，它在数据查询、统计上非常低效，而且因为先天限制，数据冗余问题严重。TXT是微软自带于操作系统的文本格式，因为微软的流行度，txt也成为了最常见的文件格式。txt格式体积小、占用空间少；格式较简单，安全性高，同时它是电脑和很多移动设备的通用格式，易于机器间的传输。目前项目中的txt文件多为机器生成。在我们假设的食品领域项目中，可以预想到，原始数据格式将以excel为主，txt为辅。所以我们需要首先处理这两种数据，将其转换为方便操作的其他格式。

### XML和json格式

XML一般指可扩展标记语言，主要用于标记电子文件和数据，使其具有结构性。结构化的数据可以方便搜索，分类，传输，这也是它被设计的目的。XML的示例如表示。

|  |
| --- |
| <country>  <name>中国</name>  <province>  <name>上海</name>  </province>  </country> |

它的标记均为自定义，所有标记成对出现，用/来区别一对标签的开始和结尾，用标记间的嵌套关系显示数据结构，整体是树形。它采用结构化的文本格式来存储和表示数据。层次结构简洁清晰，易于人编写阅读，易于机器解析或生成。因为它的简单明了，在出现后它迅速就成为了W3C提出的国际标准语言。总之，XML易于实现，可以用灵活清晰的方式管理数据整体结构，同时作为数据传输的标准语言受到广泛支持，是一种理想的数据交换语言。

JSON全称JavaScript Object Notation，是一种轻量级的数据交换格式。它的对象主要由键值对表示，用方括号标记数组，花括号标记对象，对象间用逗号隔开。示例如

|  |
| --- |
| {"InstantNoodle":  [{"name":"麻辣牛肉面","id":"1001"},{"name":"豚骨拉面","id":"1002"}]  } |

JSON与XML的定位非常相近，都是一种数据交换格式，都使用结构化方法来标记数据。但JSON在Javascript场景下更合适，更适用于web端；json的有效数据率高于xml ，可以提升网络传输效率。但JSON不像XML有清晰的结构，所以它在事先知道数据结构下非常方便，但如果不知道结构，则费时费力。在选取使用的格式时，应根据具体情况进行选择。

### Excel（txt）格式转XML（json）

目前转换excel数据为xml格式的方式主要有两种。一：使用软件。比如，1：我们可以先将excel文件导出为csv文件，一种用来交换电子表格的常用格式。2：可以在浏览器中通过头文件加” application/vnd.ms-excel”进行显示或下载。3：用MySQL，SQLServer等功能齐全的数据库里所包含的excel文件导入导出功能。 二：用编程语言开发api。这种方法比使用软件灵活度和性能高得多，但开发代码的过程较久，相比而言费时费力。在本文场景下，我们需要转换的excel格式不同，转换后存储的数据库也不同，使用软件限制太多，所以我们主要采取灵活性高的方案二，即使用编程语言开发代码解析excel数据，这里我使用的语言是python，它有一个方便的库xlrd，可以用来读取和格式化（format）excel数据。而XML库则可以处理XML文件。

Excel文件数据由表头/列名和对应数据组成。其中的表头/列名部分是静态的。为了提高效率，我们先通过XML配置文件进行结构解析。这里我们以方便面属性为例。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 厂商 | 厂商编号 | 方便面编号 | 方便面名称 | 规格 | 是否加宽 |
| 五谷道场 | 001 | 20070528761 | 庖丁时蔬 | 袋装75g | 1 |
| 统一 | 002 | 20071011SY | 浓香排骨 | 袋装75g | 0 |

其对应的XML配置文件应如：。

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <manuInstNoodle>  < manufacturer >  < value excelCol=”厂商” colIndex=”1” colName=”manufacturerName” excelType=”LABEL” />  < value excelCol=”厂商编号” colIndex=”2” colName=”manufacturerId” excelType=”NUMBER” />  </ manufacturer >  < instantNoodel >  < value excelCol=”方便面编号” colIndex=”3” colName=”instantNoodelId” excelType=”LABEL” />  < value excelCol=”方便面名称” colIndex=”4” colName=”instantNoodelName” excelType=”LABEL” />  < value excelCol=”规格” colIndex=”5” colName=” specification” excelType=”LABEL” />  < value excelCol=”是否加宽” colIndex=”6” colName=”ifWiden” excelType=”NUMBER” />  </ instantNoodel >  </ manuInstNoodle > |

当我们要使用python进行数据转换时，先根据XML配置文件进行映射。读出每个单元格的数据，根据以上config文件里excelCol列名名称和colIndex列名索引进行对应。分析以上的配置文件，我们可以通过定义三个对象，分别是厂商，方便面，和他们间的关系来建立映射，也可以使用Python强大的数据类型进行组织，即

|  |
| --- |
| [ [ { manufacturerName : manufacturerNameValue}, { manufacturerId : manufacturerIdValue } ] , [ { instantNoodelId : instantNoodelIdValue }, { instantNoodelName : instantNoodelNameValue }, { specification : specificationValue } ,{ ifWiden : ifWidenValue } ] ] |

然后再通过def ListmapXML函数进行映射。这里方法二相对简单，我采取的就是这种方法。

Txt文件的数据提取我们则可以使用正则提取的方法，通过匹配完成结构化。这里我们举例如下，这是机器生成的文件machineDatas.txt。

|  |
| --- |
| TestID: B1J1A001;Force-1: 222.871;Area-FT: 1166.289;  TestID: B1J1A002;Force-1: 222.27;Area-FT: 1093.455; |

可以写正则如:

|  |
| --- |
| machineData = data.findall(r”TestId(.+?)TestId”,str)  TestId = machineData.find(r“TestId:();”,str)  Area-FT = machineData.find(r“Area-FT:();”,str) |

通过python处理后即可映射为

|  |
| --- |
| <machineDatas>  < machineData>  <TestId>  B1J1A001  </TestId>  < Force-1>  222.871  </ Force-1>  < Area-FT >  1166.289  </ Area-FT >  </ machineData >  < machineData>  <TestId>  B1J1A001  </TestId>  < Force-1>  222.871  </ Force-1>  < Area-FT >  1166.289  </ Area-FT >  </ machineData >  </machineDatas> |

同时我们还应该考虑到，一个项目数据的录入时由多人做的，而每个人录入的每份数据可能格式有细微的差别。比如在文件夹“时间强度评价”下，对不同产品的麻度等级提供的时间强度评价表一般有16列，包括产品id一列，从0s以30s为间隔到420s。但有个别文件缺失了360-420s的记录，有个别文件多了420-450s的记录。即数据的属性可能有所缺失，多余，总体呈不完全统一状态。对于这种数据，我们在配置文件上应留下一些缓冲度，或在程度中体现出健壮性。例如：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| #file1   |  |  |  |  | | --- | --- | --- | --- | | 产品编号 | 0s | 30s | 60s | | 1001 | 0 | 4 | 13.8 | |
| #file 2   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 产品编号 | 0 | 30s | 60s | 90s | | 1002 | 0 | 9.5 | 18.4 | 20.2 | |

这里可以提出思路一，数据在提供时就应该为每个文件提供config文件说明结构，文件夹中应有一个兼容所有文件的配置文件，它包含了所有属性值，当我们处理数据时，首先读取总配置文件。但这个思路其实是将复杂的属性变化处理代偿到人工上，因为我们需要假设上传的数据文件夹自带人工整理的、正确的配置文件，而人工整理往往成本高，错误率高，所以这个思路并不完美。而第三章提到了如何以数据结构的思路解决这个问题。

## 本章小结

在本章，我们分析了食品领域项目中源数据excel和txt格式的数据处理方法，我们使用配置文件进行结构分析，然后通过映射来将其转换为XML或JSON这种半结构性数据，以方便后续操作处理。半数据结构为数据交换和存取提供了便捷，在开发项目中，我们应该结合具体情况分析，选用最合适的方法。在第三章的数据存储中，我们假定的源数据都是XML或JSON格式。

# 数据存储方案

本章是全文的重点章节，阐述和分析了数据存储方面的几种设计思路。首先简单介绍了关系型数据库，对象型数据库，图数据库，文档型数据库，随后分别举出其经典产品，分析特点和优劣，并结合本文的场景进行建模设计分析，从读，写，查等方面对比分析其性能，最后展示了以上数据库管理系统的结构和运行结果。

## 多种数据库概述

在数据存储技术发展的过程中，我们经历了三个时代。第一代数据库系统主要指的是网状和层次数据库。它们分别以网状和分层的数据结构为基础，初步解决了数据的集中和共享问题。但它们逻辑独立性差，可移植性低，使用也非常繁琐。第二代数据库为关系型数据库。是建立在关系模型基础上的[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)，借助于集合代数等数学概念来处理[数据](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE/33305)。第三代是新型数据库，主要指各种NOSQL数据库。NOSQL数据库主要有四种，列数据库，键值数据库，文档型数据库，图数据库。以下我们将结合第二代和部分第三代数据库，分析食品领域下的数据存储设计。

## 关系型数据库

### 概述

第一代数据库出现后，不到五年就进入了第二代。20世纪70年代即为关系型数据库理论研究和原型开发的时代。关系型数据库的一大进步就是引入了关系模型，关系模型的提出在数据库发展史上具有划时代的意义。关系模型由关系数据结构、[关系操作](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%93%8D%E4%BD%9C)集合、关系完整性约束组成。关系数据结构指我们将数据规范化处理为二维表，属性通过头部属性元组定义，关系表中的行将头部属性元组映射到单元值，并将数据间的联系以键的形式联系在二维表之间。数据操作是通过关系代数处理选择（where），投影（select），笛卡尔积（join）等。关系代数运算被简化在了SQL语句里，关系代数和SQL的关系如表示。关系完整性约束包含数据完整性，实体完整性，域完整性，参照完整性和用户自定义完整性。

|  |
| --- |
| SELECT x.id FROM InstantNoodle x Where x.name = "康师傅" |

相比第一代数据库，关系型数据库数据结构简单清晰，数据一致性、可维护性和独立性强，SQL语言方便程序员操作，这些优点都使得它风靡一时，虽然发展至今业务场景越来越复杂，需求变更速度越来越快，关系型数据库渐渐体现出了它对复杂类型数据的无能为力和处理事务时的繁琐复杂，但依然有着极强的生命力，依然是使用者最多的主流数据库类型。

传统的关系建模流程大致是：画草图->画er图->映射成表。在过程中需要把数据间的联系以逻辑的形式存储下来。同时应该使用范式，约束等技术整理模型结构，减少数据库中数据冗余，增进数据，这个过程被称为规范化。在第二章提到的场景下，测试员在给出属性数据时，可能出现不完整数据或多于数据，如测试属性缺失，测试次数增减，而且产品属性条目在过程中可能频繁改变，所以数据属性可能有细微的区别，结构不完全相同。以数据存储的思路，为了解决这个问题，在关系型数据库中我们有四种解决方案：

一，为每种相同属性值的产品建立类型表；这种思路会导致大量冗余表，并且每个类型的查询语句都不一样。

二：建立一个包含了所有不同属性的表；这样做消除了方法一查询语句的问题，可以用一个语句查询查询所有类型，但这样增大了存储空间开销，可能会产生稀疏表。而且表结构会经常变化，需要大量的alter table操作。

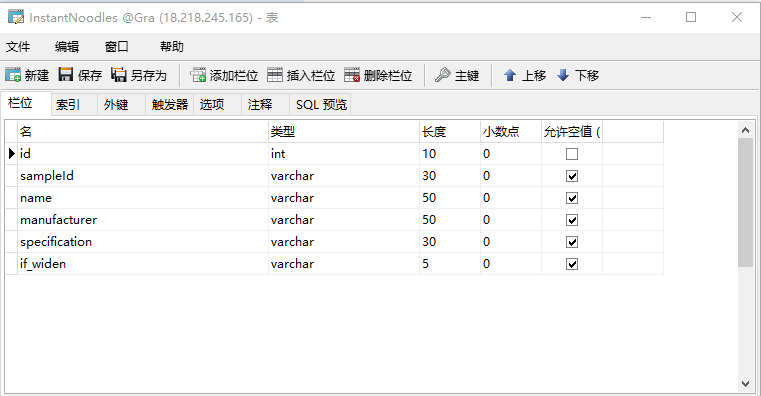
三，使用继承的思路，为产品中通用的属性建立单独的表，其他独有的属性建立具体的表，这样结合了方法一和二的优点，拥有更好的结构，但当查询时，我们需要进行join操作，每次join操作需要至少一次随机寻道，意味着低效昂贵。

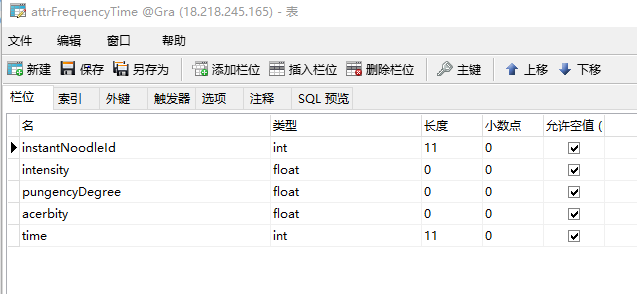
四：设置三元结构：标识id/属性名/属性值。这是最灵活但最冗余的结构，而且查询非常复杂。只能适用于写操作远远多于读操作的情况，如存储日志数据。

### 关系型数据库实例之MySQL

MySQL是最流行的关系型数据库管理系统，它开源且社区支持强大（在被Oracle收购前），深受中小型网站和个人开发者的欢迎。MySQL系统特性包含了多语言，多平台，多线程，自我修复式集群支持，并为多种常见的编程语言提供了 API。如C、C++、Python、Java、Perl、PHP、Eiffel、Ruby,.NET和 Tcl 等。MySQL既能作为一个单独的应用程序应用在服务器环境中，也可以作为库而嵌入到其他的软件中。但因为MySQL 不够成熟，性能相比竞争对手太弱，它近几年在慢慢被放弃。

这里我们把MySQL作为本文场景数据存储最基本的方法，之后我们会以MySQL为基础对比分析其他数据库的优劣。以下图一二是MySQL中数据表的示例，图一：食物属性。图二：方便面味道属性比重随时间变化情况。

 图一

图二

### 关系型数据库实例之PostgreSQL

PostgreSQL是非常优秀的开源对象关系型数据库管理系统，它是最先进的关系型数据库之一，自从MySQL被Oracle收购以后，PostgreSQL逐渐取代MySQL成为开源数据库的首选，苹果即为从MySQL 转向 PostgreSQL的代表厂家之一。它支持完整的SQL标准,具有几乎所有 SQL[构件](https://baike.baidu.com/item/%E6%9E%84%E4%BB%B6)并且提供了许多其他现代特性，如触发器、存储过程、视图、事务完整性。

在其他关系型数据库不同，PostgreSQL有自己的非标准扩展模块，这也使得它领先于同类型其他产品。一： 它有自己的语言——PL/pgSQL, 也有C，C++，[Java](https://baike.baidu.com/item/Java/85979" \t "_blank)，Perl，Tcl，Ruby和 Python的开发语言绑定。二： 他支持更多的数据类型。它拥有cube数据类型，可以将数据映射为多维向量，实现对大量数据的空间聚集，空间计算；支持文档对象，如json，hstore和jsonb，借鉴了文档型数据库的优点，其中的jsonb性能优异，甚至对文档型数据库造成了冲击。三：它有函数索引和部分数据索引方法，有更强大的查询优化器，支持复杂的查询处理，相比MySQL只支持nest join, 它还支持hash join和 sort merge join，同时还可以用正则表达式查询。四：它支持SQL/MED标准，有自己的FDW([Foreign data wrappers](https://link.zhihu.com/?target=https%3A//wiki.postgresql.org/wiki/Foreign_data_wrappers)，即外部数据包装器)，可以用统一的SQL，通过FDW去访问远程数据存储，例如关系型数据库、NoSQL数据库，不同格式的文件，在线数据集。

PostgreSQL建模、设计表格的过程与MySQL类似，这里我们举例说明PostgreSQL的查询和索引方式。在食品领域中，测试员对方便面的口味分析后给出了如表InsantNoodleTastes数据。口味打分从0到10越来越强烈，10位最强。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 方便面编号 | 麻味 | 辣味 | 鲜味 | 甜味 | 酸味 |
| 1001 | 5 | 7 | 3 | 0 | 2 |

当我们需要查询与1001编号口味类似的方便面时，我们可以建立cube数据类型的列InstantNoodleTasteVector，这是一个五维向量表，向量表中的值对应每个口味维度的打分，1001在其中的值为(5,7,3,0,2)。此时，找到与1001距离较近的向量点，也就找到了口味类似的方便面，口味的相似度与两点间的距离成正比。我们可以使用cube\_distance()方法进行SQL查找，结果按距离升序排列。

|  |
| --- |
| SELECT InsantNoodleId, cube\_distance(InstantNoodleTasteVector,’( 5,7,3,0,2)’) distc  FROM InsantNoodleTastes  ORDER BY distc; |

但这样查找我们需要进行全局遍历，依次求出他们和1001编号的位置，非常低效。我们可以使用cube\_enlarge(cube,radius,dimensions)方法限制范围，只得到以1001编号对应的点为中心，6单位立方体内，距离最近，相似度最高的3条数据。

|  |
| --- |
| SELECT \*  FROM InsantNoodleTastes s,  (SELECT InstantNoodleTasteVector,InsantNoodleId FROM InsantNoodleTastes WHERE id = '1001') x  WHERE cube\_enlarge(x.InstantNoodleTasteVector, 6, 5) @> s.InsantNoodleTaste AND x.id<>s.id  ORDER BY cube\_distance(x.InstantNoodleTasteVector, s.InstantNoodleTasteVector)  LIMIT 3; |

### MySQL和PostgreSQL的性能对比测试

上一节介绍PostgreSQL时，我们强调了它的高性能，这里我们做实验对比一下MySQL和PostgreSQL在读写上的性能。测试用表根据数据量大小分为两个，都以id为主键，结构相同，表一十万行数据，表二一千万行数据。结构如tableOne(id int primary key, value1 int, value2 varchar(20))。运行环境一样，都为ubuntu1核1G内存，服务器配置都设为默认。测试设计为四个部分，分别是：1.根据主键id随机读数据，2.单行插入。3. 单行删除。4.单行更新。测试指标包括每秒查询数和查询响应时间。每秒查询数测试了数据库系统可以支持每秒执行多少次操作，说明了它的吞吐量性能。而查询响应时间决定了数据库系统操作的速度。我将实验结果用python的matplotlib进行画图。

结果如下：

测试一：

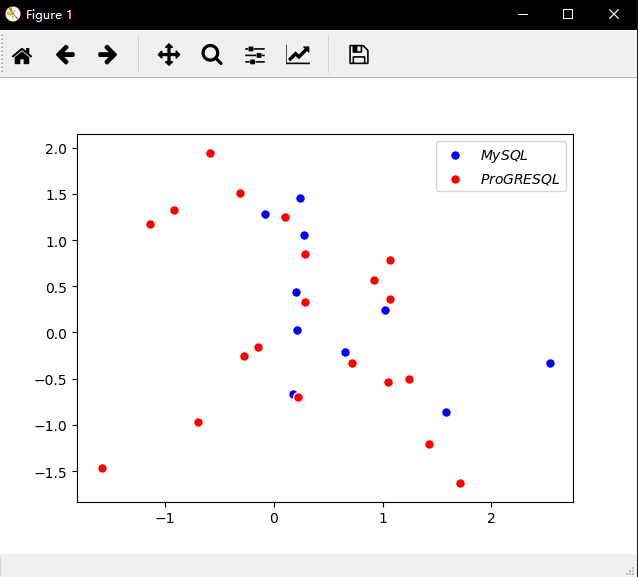


图3.2.4.1

测试二：

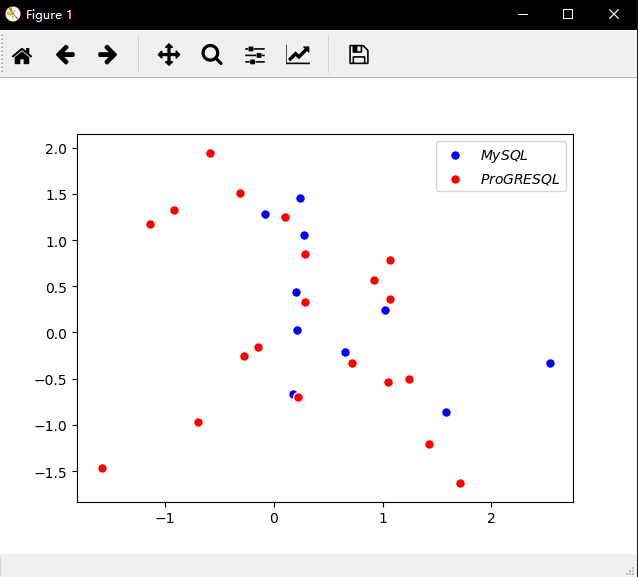


图3.2.4.2

测试三：

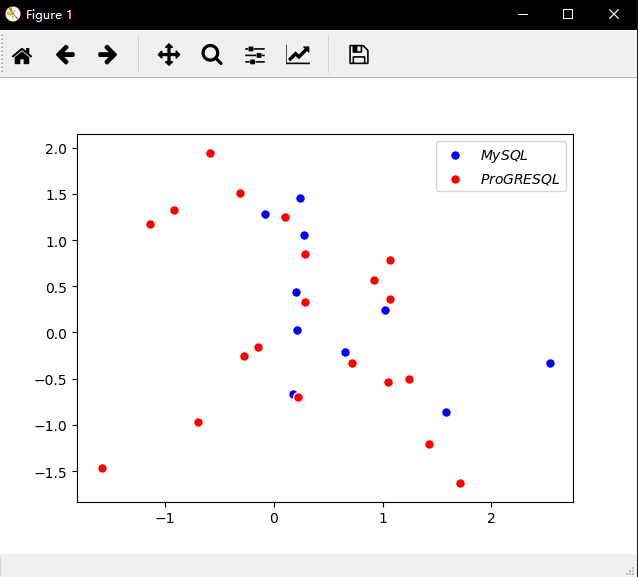


图3.2.4.3

测试四：

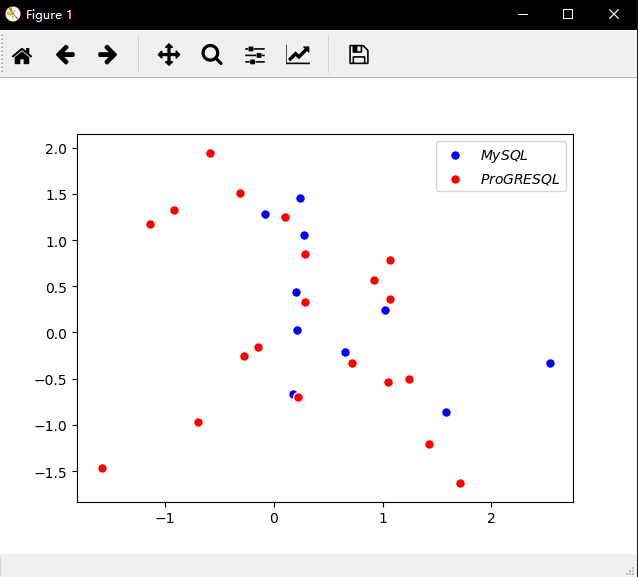


图3.2.4.4

## 对象型数据库

### 概述

虽然关系型数据库在存储管理方面性能强大，但由于以上提到的结构僵化等问题，80年代出现了面向对象数据库技术，按照 “面向对象数据库系统宣言”的思想，面向对象性数据库系统(OODBS)特性归总为必备特性（复杂对象，唯一对象标识，封装性，类型，继承性，可重载性，可扩充性，计算完备性，辅存管理，并发性，可恢复性，即时查询）和可选特性（多重继承，类型检查和推理，分布式支持，类型系统）。面向对象的数据库系统从面向记录上升为面向对象，允许结合数据抽象机制对复杂对象建立模型，从而大幅度提高效率，降低用户使用复杂性。后来对象型数据库思想不断发展分化，趋势大致如：面向对象数据库->对象关系型数据库->对象代理数据库。

面向对象数据库主要是数据库与面向对象语言的结合。由以下四部分组成：对象模型（主要包括对象的特性，对象间的联系），对象描述语言，对象查询语言，对象语言绑定。关于对象模型，有种说法是“面向对象数据库模式是类的集合”。面向对象的数据模型提供了一种类层次结构，一组类可以形成一个类层次，在一个类层次中，一个类继承其所有超类的全部属性和方法。而对象语言绑定的出现主要是为了方便的加入面向对象功能，由用户灵活的实现定义并操作对象，常见的是对象语言绑定是C++和JAVA。但语言绑定是个双刃剑，为了满足对多语言绑定的支持，对象型数据库往往需要牺牲一些良好的特性。

接下来，我们在数据建模上与关系型进行对比分析。假设现在面对的对象是个复杂对象，它结构复杂，维度多样，且对象之间有较多组合继承关系。针对这种复杂对象建模时，1.关系数据库需要为所有数据在每种尺度下建立自己的表，访问时，通过不同尺度的查询条件，查询不同的表来返回信息。往往同种数据在不同尺寸下的表会有大量数据冗余，这加大了同种数据一致性维护难度。而如果通过对象集成，可以很好的降低这一难度。2.其次，我们无法在关系型数据库中实现对数据的计算，如添加c=a\*b，c属性必须也要建立一个字段，而对象型可以在表上定义一个计算的函数或者方法来实现。3.一些结构关系复杂的数据类型无法在关系型中储存，需要通过基础的数据类型去模拟记录，这往往导致数据库表单的设计过于复杂，而且需要大量的join来实现复杂数据的表达，而对象型为了满足多尺度的要求，可以对统一对象创建多个类并根据类层次结构来进行管理。

分析其组成并结合表现，我们可以得出结论，面向对象型数据库支持复杂对象建模方法。对象的独立对象标识可以加快查询速度，对象的封装性实现了良好的透明度，使用者不需知道操作的细节，加快了效率。因为面向对象数据库对复杂对象的表现优异，它广泛应用于多媒体，工程数据，地理信息等复杂信息管理系统。但由于它没有原型数据库做基础，完全从零开始实现一个对象型数据库，虽然足够灵活，但也开发费时费力，开发者需要手动实现数据库功能，自主开发需要的扩展数据类型，同时还要考虑性能优化问题，而且与其他数据库的转接兼容也相对麻烦。所以这种思路虽然很强大，但受到的限制还是太多，缺乏数学模型的支持，复杂难用，成熟度低，最后没有撼动关系型的地位。

之后，人们在对象型数据库的基础上建立了对象关系型数据库。其对象关系数据模型结合了关系数据模型和对象思维，支持sql和关系模型，也允许用户扩展复杂数据类型或自定义函数功能。对象关系型数据库的典型有oracle数据库。

### 对象型数据库实例之Db4o

面向对象数据库的概念提出后，最早的商品化管理系统出现在 1986 年，2004年，db4o作为免费的开源纯面向对象数据库管理系统发布，之后db4o首先将Native Queries 作为面向对象的数据访问API 来实现，该API完全依赖于编程语言（Java / C＃）本身，由此它成为了100%原生数据库。目前国内相对比较流行的也是db4o，但很可惜这个项目已经在2014年被官方停止维护。

Db4o 提供两种运行模式，分别是本地模式和服务器模式。一般来说，javeEE更适用于服务器模式（这里假设api语言用的是java），嵌入式开发更适用于本地模式。Db4o有几大特点：一：轻量级。实现该数据库只需要在系统中引入400k左右的jar包或者dll文件。二：原生。开发者不需下载管理工具，不需学习新的数据库语言，可以直接用编程语言管理数据库。三：多平台支持。db4o 支持Java 平台，.NET 平台甚至 J2ME 方言环境。四：性能优秀。下图3.3.1.1为db4o 官方公布的基准测试数据，可以看到，在读写查删时，它的速度甚至可以比Hibernate/MySQL 方案的速度高出几到几十倍。

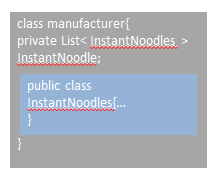


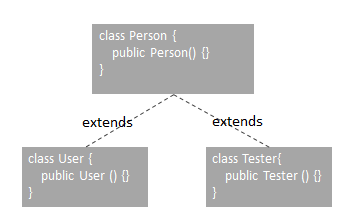
图3.3.1.1.

接下来我将使用db4o结合我们的食品场景进行数据库设计。因为db4o是原生数据库，我们拟采用java进行数据库开发。首先，在设计类时，我们可以发现面向对象的思想给方便面复杂的属性管理提供了不少便利。我们可以设计两个父类：用户，方便面。接下来使用继承的思路，把人分为测试员（测试用的机器为了方便也可以放在这个子类里，作为人的特性被设置为null即可）和用户（购买并食用方便面的客户）两个子类。方便面和人之间存在两种关系：一：用户—吃&评价是否喜欢🡪方便面，二：测试员从多个角度（时间，定性，定量，数据）--打分🡪方便面。举例如方便面类模型，可以简单描述为：

|  |
| --- |
| **public** InstantNoodles(Integer id, String sampleId, String name, String specification , String manufacturer) {  **super**();  **this**.id = id;  **this**.name = name;  **this**.sampleId = sampleId;  **this**.specification = specification;  **this**.manufacturer = manufacturer;  } |

当厂商较多时，可以单独构建厂商类，将方便面作为厂商的成员变量实现。结构如图3.3.1.2，而人类可分为两种子类，通过继承建立。结构如图3.3.1.3。

图3.3.1.2

图3.3.1.3

建立类并将数据存入后，当我们需要新增食物属性时，我们可以直接在食物的类里定义新的属性和对应的方法。当我们需要查询食物信息时，我们有三种查询方式：QBE（Query by Example）、NQ（Native Queries）、SODA（Simple Object Database Access）。QBE 标准是由IBM提供的一种比较传统成熟的查询。SODA是 db4o 最底层的查询 API，麻烦但高效。db4o 更推荐使用 NQ 进行查询，NQ 方式支持原生语言，我们可以直接使用java或者.NET来创建数据库并查询数据。但我们也需要注意，NQ的本质是转换接口，数据库实现内部会设法把NQ转换为SODA。当查询表达式的流向图难于分析时，db4o 将需要实例化某些持久对象，这个过程会影响性能。我们可以通过以下伪代码感受SODA和NQ的区别。

|  |
| --- |
| //SODA方法  **public** **static** **void** testSODAQuery(ObjectContainer db) {  Query query = db.query();  query.constrain(InstantNoodles.**class**);  Query idQuery = query.descend("id");  query.descend("name").constrain("西红柿牛腩").or(idQuery.constrain(2).greater().and(idQuery.constrain(4).smaller()));  ObjectSet<InstantNoodles> result = query.execute();  DBQuery.*listResult*(result);  } |
| //NQ方法  **public** **static** **void** testNQQuery(ObjectContainer db) {  List<InstantNoodles> result = db.query(**new** Predicate<InstantNoodles>() {  **public** **boolean** match(InstantNoodles istn) {  **return** istn.getId() > 1002 && istn.getId() < 1004 || istn.getName().equals("西红柿牛腩");  }  });  DBQuery.*listResult*(result);  } |

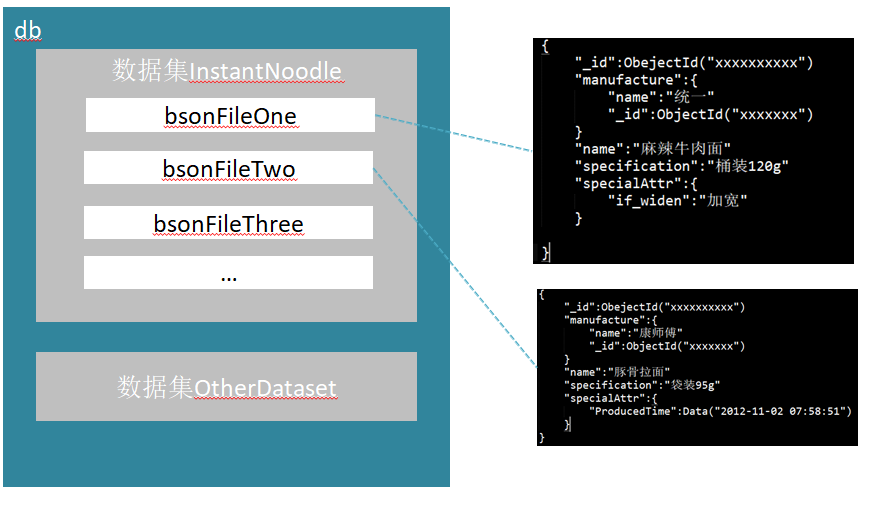
## 文档型数据库

### 概述

顾名思义，文档型数据库面对的存储对象主要是文档数据或类似的半结构化数据。每个文档对象有自己独有的标识符ID，并以某个标准化格式存储，常见的格式如JSON, YAML, XML, protobuf ，文档以键值对的格式存储数据，支持嵌套状态，字段值可以包含其他文档，数组，甚至文档数组。它是NOSQL数据库中最像关系型数据库的一种。这种相似主要体现在面向集合存储的基础上，数据被分开存储在数据集合中，每个数据集有自己的独有标识id，包含多个嵌套键值对模式的bson文档，这里数据集合和关系型数据库中的表概念相似。但不同于关系型数据库，文档型数据库没有模式，它是模式自由（Schema-free）的，数据集合内部的结构并非完全相同，一个数据集合可能包括多个格式不同的文档。当数据库中没有大量关系和标准化约束，可以分割为多个结构化文档时，使用文档型数据库是非常合适的选择。

### 文档型数据库实例之Mongodb

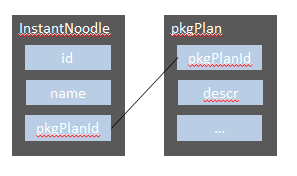
MongoDB是一个基于C++的开源分布式文档型数据库，提供适用于web的灵活的、可伸缩、高性能数据存储方案，主要针对大体量的数据集。它的数据存储格式为JSON的衍生类型——BSON。如图是一个MongoDB数据集合示例，db数据库下的InstantNoodle数据集合包含了多个bson格式文件，每个文件的结构并不完全相同。



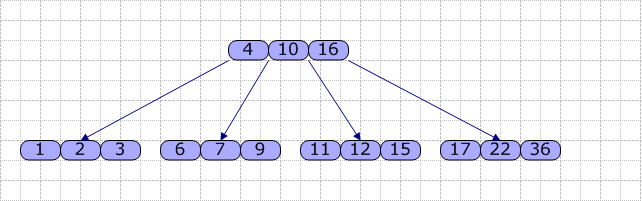
由于MongoDB的嵌套文档结构，它在查询的时候也不需要使用join操作，而是直接调取文档，使用动态查询。同时，MongoDB支持完全索引，即文档中出现的每个字段都可以用于建立索引，它内部实现索引的数据结构是B树，索引的机理是，带来的性能提升如。

MongoDB致力于解决大量数据的存储问题，因此它支持分布式，这是它区分于其他文档型数据库的地方。它可以在多台服务器上实现伸缩，通过复制和分片的方法并行执行查询。为了支持分布式， MongoDB抛弃了事务的概念。事务是关系型数据库非常重要的特点，关系型数据模型依赖于原子性的事务以保障数据库的一致性，事务间相互独立，事务成功提交后稳定持久存在，遵循ACID规则，使用两段式协议。而MongoDB作为分布式数据库，应用场景包含大量跨服务器操作，如果使用这种严格的协议和规则，数据库需要：先在每个相关服务器内准备执行事务，给数据加锁，保证没有一致性冲突；所有相关服务器准备好后，每个服务器作为事务的一部分再进行更新。这个过程增加了网络时长开销，会限制执行效率，导致极大的性能下降。因此MongoDB采取其他方式来维持一致性，模仿事务行为，比如使用嵌套结构和代偿方法。

在3.2.1中，针对属性数据可能不完整或多余的情况，关系型数据库给出了四种解决方案，此场景我们套用MongoDB模式自由，支持嵌套的思想再次思考，类似于关系型中的方法三，我们可以在通用的属性外构建键为”specialAttr”的属性，将某类型独有的属性以数组或字典的方式嵌套进去。这样我们就有了更灵活可扩展性更好的数据结构，方便了写操作。而局部嵌入后，读的性能也有所进步，考虑到数据库读数据前的磁盘寻道过程，当关系型数据库查询数据进行join操作时，系统需要使用至少一次随机寻道，而MongoDB在磁盘上连续存储文档，在查找时避免了随机寻道，提升了速度和性能。同时，这样的结构也方便了维持数据一致性，举例某方便面有多种包装,包装方案也作为数据集合存在，现在我们需要不再生产该方便面，因此需要从产品集合中删除该产品及其相关信息。如果使用关系型数据库方法三，如图3.4.2.1，我们需要执行一个包含了多个操作的事务：在产品表中删除该产品，在包装表中删除它对应的包装，处理该事务时数据库应该保证其原子性。而MongoDB中我们只需要删除这个嵌套对象即可，如图3.4.2.2。相比既保证了ACID，又优化了操作的速度。

 图3.4.2.1图3.4.2.2

MongoDB的索引结构是B+树，B+树是目前索引上表现最好的数据结构，它的原理如图3.4.2.3示，在MongoDB中的作用如图示。

图3.4.2.3

了解了MongoDB的结构和优点后，我们可以分析得出它的缺点。一：由于MongoDB是模式自由的，所以它没有固定的字段名和类型，每个文档都需要重复的存储字段名和类型，当文档值较短而属较长时，存储效率很低。二：将任意类型的任意值插入任何集合的行为，虽然自由但也很危险，没有了严格的约束和管理，在实际应用中很容易因为使用者的错误而影响效果。

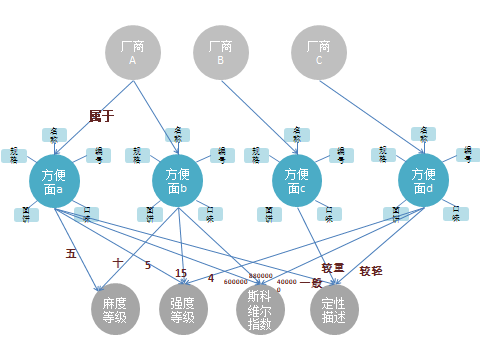
## 图数据库

### 概述

作为NoSQL的一员大将，图数据库近几年被广泛的讨论，其使用者也逐渐增多。图是节点和联系的集合，其中比较主流的图模型主要有属性图， RDF三元图和超图。拿最流行的属性图举例，它主要有两个特性。一：包含节点和联系，节点有自己的属性。二：联系有名字和方向，也可以有属性。这个模型非常通用，是一个符合人们思维习惯的建模思路，可以抽象出来用于计算机世界里的数据存储，图数据库就是建立在这样的基础上的。图数据库使用节点，边/关系(relationship)和属性。节点代表实体，关系代表实体间的联系，属性是节点的相关信息。引用维基百科对图数据库的定义“在计算机领域中，图数据库使用图形结构进行语义查询，使用节点、边和属性来表示和存储数据。该系统的关键概念在于图形直接关联数据集中的数据项。”

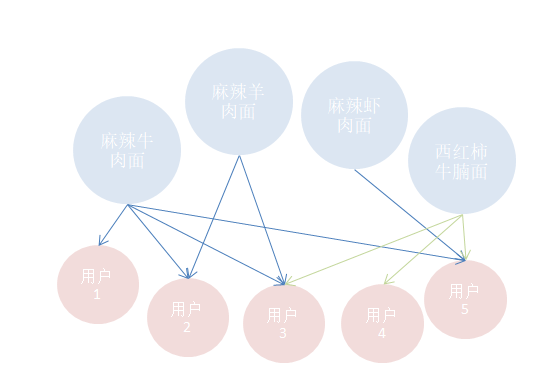
上文3.1我们讨论过传统的关系建模需要进行规范化，生成第N范式数据模型。但实际项目中，为了适应变化的需求，我们常常会使用反规范化（又称非标准化）技术。反规范化技术上是指在实现关系型解决方案的过程中，为了在适应场景，适应关系模型的同时减少join操作，提高性能，我们对范式模型强加了很多变化，人为制造重复数据和重复结构。这种反规范化是非常有必要的，但它对设计者提出了较高要求，而且增大了迁移花销和风险成本。

而图建模不需要规范化和反规范化两步。因为图建模思路和人的思维习惯相符，我们可以很轻松的将复杂的层次结构高保真的抽象出来。而在需求变化上，因为图具有天然高可扩展性，相比关系型需要不断确认逻辑模型设计，图数据库在修改实体和联系上非常方便，当我们需要新增实体并添加于已有实体间的关系时，我们不需要修改本来实体的内容，只需要新增节点然后添加一条指向已有节点的边即可。以下为食品领域场景下的方便面属性简单图形建模。



这样分析了图建模的方式后，我们可以总结出图数据库的两大优势：1. 灵活性，可扩展性高。不必要一开始就把整个系统的结构设计出来，可以在开发的过程中低成本的增删节点和联系。2. 敏捷性。和文档型数据库一样，图数据库是无模式的,它没有以模式为导向的数据管理机制。它提供了数据模型的平稳演化，数据迁移和反规范化很少会成为问题，新的事实和新的组件会成为新节点和新联系。而优化关键性能访问模式通常需要在两个原本由中介连接的节点间引入一个直接联系。总结以上的分析。图数据库适用于处理半结构化的，紧密关联且变动性大的的数据。现实中，图数据库典型使用的领域如[语义网](http://en.wikipedia.org/wiki/Semantic_Web)、[社交网络数据建模](http://blog.neo4j.org/2009/09/social-networks-in-database-using-graph.html)、[LinkedData](http://en.wikipedia.org/wiki/Linked_Data)、[基因分析](http://code.google.com/p/pygr/)、，地理空间（基于位置的服务）、网络拓扑、[推荐算法](http://wiki.github.com/tinkerpop/gremlin/linkeddata-sail#recommendation)等。

图模型在查询上性能优异。关系模型使用数据中的信息关联数据。例如，在本文的食品领域，当需要查找所有写过名字里包含“麻辣”的方便面的用户时，它通过遍历整个表，在方便面表的name字段中查找字符串“麻辣”，对应得到方便面id，再通过用户表查找与之关联的用户名。这在大型表中是一个耗时的过程，因此关系数据库提供了数据库索引的概念，以提高查询速度。而相比之下，图形数据库直接存储记录之间的关系。同样是查找用户的名字，图数据库使用户有直接指向方便面记录的指针，而不是通过方便面id查找用户表的字段。也就是说，选择一个方便面后，可以通过指针直接得到对应的用户群，而不需要搜索用户表来查找匹配的记录。这可以消除连接（join）操作。



这种优势在进行深度较大的搜索时更加明显。当系统需要查找和方便面A的口味相近（通过测试员给出的某个评价标准b判断）的其他方便面最近在喜爱它们的用户群体内一个月的购买量时，关系型需要多次搜索匹配，而图数据库只需要查找节点，顺着边继续进入下一个深度即可。从技术上讲，图数据库共用了O（log（ n））+ O（1）时间，即大致相对于数据大小的对数。而关系版本将是多个 O（log（ n））查找。由此我们可以得出结论：图数据库具有良好的查询性能。因为哪怕在数据集增大的情况下，查询也总是和图的一部分相关，因此，每个查询的执行时间只和满足查询条件的那部分遍历的图的大小而不是整个图的大小成正比。（使用免索引链接，图数据库可以将复杂的join查询转化为快速的图遍历，节点和关系只有在结果迭代器需要访问它们的时候才会被遍历并返回，对于大规模深度遍历而言，这极大地提高了性能。因此无论数据集有多大，都可以维持毫秒级的性能。）

### 图数据库实例之neo4j

在图数据库领域，现在比较流行的数据库有：[Neo4j](http://www.neo4j.org/) ，[AllegroGraph](http://www.franz.com/agraph/allegrograph/)， RDF-QuadStore，[Sones](http://www.sones.com/)，[Virtuoso](http://www.openlinksw.com/) ，[HyergraphDB](http://www.kobrix.com/hgdb.jsp)。实际开发中，我使用的数据库是Neo4j。Neo4j是一款用Java实现的可伸缩的开源高性能图数据库，数据以一种针对图形网络进行过优化的格式保存在磁盘上。它具有常见的数据库特性，使用键值和关系进行索引，完全兼容ACID，支持事务，强制要求每个对数据的更改都需要在一个事务之内完成，以保证数据的一致性。它拥有AGPL/商业许可，是开源的有一大贡献。Neo4j既可作为内嵌数据库使用，也可以作为单独的服务器使用，在这种使用场景下，它提供了广泛使用的[REST](http://wiki.neo4j.org/content/Getting_Started_REST)接口，能够方便地集成到基于[PHP](http://github.com/onewheelgood/Neo4J-REST-PHP-API-client)、.NET和JavaScript的环境里。开发者可以通过[Java-API](http://api.neo4j.org/current/)直接与图形模型交互，随着开源的发展，目前它也支持[JRuby/Ruby](http://github.com/andreasronge/neo4j)、[Scala](http://github.com/ept/neo4j-scala-template)、[Python](http://components.neo4j.org/neo4j.py/)、[Clojure](http://github.com/JulianMorrison/neo4j-clojure)等其他语言。而它的这两个版本都可以做成集群，从而可以使用主从复制实现高可用性和水平读扩展，甚至可以将两个版本进行混合，在数据库级执行。

接下来我们尝试将图数据库与场景结合起来。根据1.3.1的场景，我们简单分为三个模型。模型一：方便面的属性记录。这里主要包括表层属性，分数情况。模型二：测试员分析过程中的打分记录。这里数据模型简单，数据量大。需要注意的是如何把时间因素记录得当。模型三：用户的反馈记录。这里可以为后续的推荐功能，产量设计提供数据支持。在以上提到的食品领域，源数据的主要特点是结构简单，数据量大，变动性高。那么，个人比较遗憾的发现，图数据库最大的优点——联系在这个场景里并没有很好的得到表现。因为食物之间并不会有很多的事件交集，显然无法像社交网络那样完美的体现图数据库中联系的便利。

虽然事物间的联系不多，但从以上描述中可以清晰看到一个产品属性很多。某些属性用了多种角度分析，在味觉粒度和时间粒度上都做了细化。也就是说，一个物品的属性之间是有联系的，这样也可以让图数据库发挥特长。而且，在反馈分析这里，我认为图数据库是非常适合的。不同人对不同事物有不同的喜好。这个过程体现了3种联系：人和食物（是否喜爱），人和人（口味是否类似），食物和食物（是否是统一风格大类）。这种情况下可以使用图数据库协助做推荐系统。

同时，本情景需求变更快，源数据的变动性高，对数据模型的可扩展性要求高。如果整体模型太过僵化，那么食物每增加一种评价方式，每发生一次某属性的突然增删，测试员每提供不完全统一的数据，数据库每涉及到迁移（无论大小），无疑都是一次痛苦的重新设计，系统的维护成本升高不说，万一由于操作者（开发者）经验问题，很可能导致冗余数据/丢失数据/模型臃肿等恶性情况的发生。这大大增大了项目的风险，也有很大概率降低系统的性能。而考虑到如上的可能性，图数据库天生良好的可扩展性无疑非常适合于需求变更快的敏捷开发模式。它没有那些复杂的物理模型或概念模型转换。建立表结构非常简单，在原有表上新增属性和联系只需新增节点并画上线即可。再其次，在数据的查找方面，图数据库的查询只发生在图的局部，不需要遍历整个库，而且由于它联系分明，在深层次的数据查询方面比传统的优势巨大，传统关系型查询时间随深度增加是指数级增加的，而图数据库是线性增加，当深度大于3时，其中的差距非常明显。

然而，图数据库建模还是有表现不佳的地方，主要体现在对时间流的掌控上，图的亮点在于实体间的联系，而时间是流式的，每个时间都有新的不同的数据，图数据库表示时间一般是两种方法，时间轴数和链表。这种结构相对低效，放在本例的场景下并不合适。

以方便面属性为例，我们将方便面与厂商拆为两个对象，分别对应两个标签(label)的结点，再通过Relationship函数进行关联。设计如：

|  |
| --- |
| for InstantNoodle2 in InstantNoodles2:  node = Node("InstantNoodle2",label = "InstantNoodle2",\*\*InstantNoodle2)  tx.merge(node)  for manufacturer in manufacturers:  node = Node("manufacturer",label = "manufacturer",\*\*manufacturer)  tx.merge(node)  i=1001  while i<1031:  node1 = g.find\_one(label="InstantNoodle2",property\_key="id", property\_value=i)  node2 = g.find\_one(label="manufacturer",property\_key="name",  property\_value=str(mapInmf[i]))  rel = Relationship(node1,'belongs\_to',node2)  tx.merge(rel)  i = i + 1 |

## 不同数据库的性能对比

Xxx

graphdb-benchmarks是图数据库之间的基准工具，支持Titan，OrientDB，Neo4j和Sparksee。我们主要使用这个基准检查Neo4j在执行时间上的性能。基准测试由四个工作负载组成，即聚类，大规模插入，单插入和查询工作负载。每个工作负载的设计都是为了模拟图形数据库系统中的常见操作。测试结果如图3.6.1示。

## 本章小结

本章，我们介绍了数据库技术发展的后两代成果——关系型数据库，对象型数据库及NOSQL数据库中的图数据库和文档数据库，分别以MySQL，postgreSQL，db4o，neo4j和MongoDB为例。在每个数据库分类里，我们介绍了他们的来源，特征，优劣，并结合食品领域场景进行了简单的建模分析。目前数据存储的方式百花齐放，可以将数据库类型分为关系型，对象型，键-值型，列型，文档性，图型。虽然不同的数据库成熟度不同，使用者量级不同，底层数据结构不同，但没有哪个数据库是最好的，只有哪个数据库在某种情况下是最合适的。在选择最终使用哪个数据库的时候，我们需要考虑的不只是数据库的类型，还要考虑数据的体量，场景的复杂性，可扩展性等等。例如当我们构建社交网络的场景时，似乎第一反应就是图数据库，但我们同样需要考虑当数据量太多，我们是否需要加入“大数据”的实现，引入Rick或HBASE。所以我认为数据管理的未来会是多持久并存模型，即一个项目中使用多种数据库。这样就可以最大化利用每种数据库类型的优势。但这个想法意味着不同数据库间必须要有兼容彼此的接口，可能只能等一个足够有话语权的组织制定规范并推广开后才会实现，目前SQL标准已经添加了SQL / MED的规范，但它未得到广泛的使用，只有postgreSQL等寥寥几个数据库支持，多持久并存模型的发展依然前路漫漫。

# 食物项目系统的设计与实现

本章主要介绍食物项目系统的设计与实现，包括系统整体设计、数据库设计以及系统内各功能模块的设计与界面展示。

## 系统整体设计

### 开发环境简介

本文设计的食物项目系统搭建于Ubuntu环境上，使用的服务器是亚马逊云ASW。过程中主要使用的语言包括Python, Java , PHP, HTML, JavaScript，使用的数据库如第三章提到的，包括MySQL，postgresql，db4o,neo4j，MongoDB。

Ubuntu是最流行的开源Linux操作系统。它免费，安全，用户友好，可高级定制，有强大的社区支持。作为服务器，它相比Windows便宜，稳定，且性能更高。因此，在搭建食物项目系统时，我的Ubuntu环境版本为Ubuntu 16.04.4，Linux版本为4.4.0-1052-aws (buildd@lgw01-amd64-031)，远程连接的工具是mobaxterm。

食物项目系统的前端和后台语言主要是HTML和PHP。虽然前端开发技术日新月异，但它不是本文讨论的重点，因此本系统我们采取传统的JavaScript，结合了DevExtreme和BJUI框架。而PHP是一种语法简单的通用开源脚本语言。它广泛应用于web系统，因为它可以嵌入到HTML里，相比其他语言开发实用方便，并且，它与MySQL合作默契，适合这个个人开发者的小项目。

数据库这里我采用多种存储方式，以MySQL为切入点，既可通过更改后台接口切换数据库的选用，也可将多个数据库结合起来使用。这里db4o因为功能局限，只作为示范通过接口切换使用，ProGRESQL由于和MySQL定位相近，只做了示例和性能对比，并没有嵌入系统。而MySQL,neo4j,MongoDB则在本系统中结合使用。

### 系统整体架构

系统整体分为三部分构建。

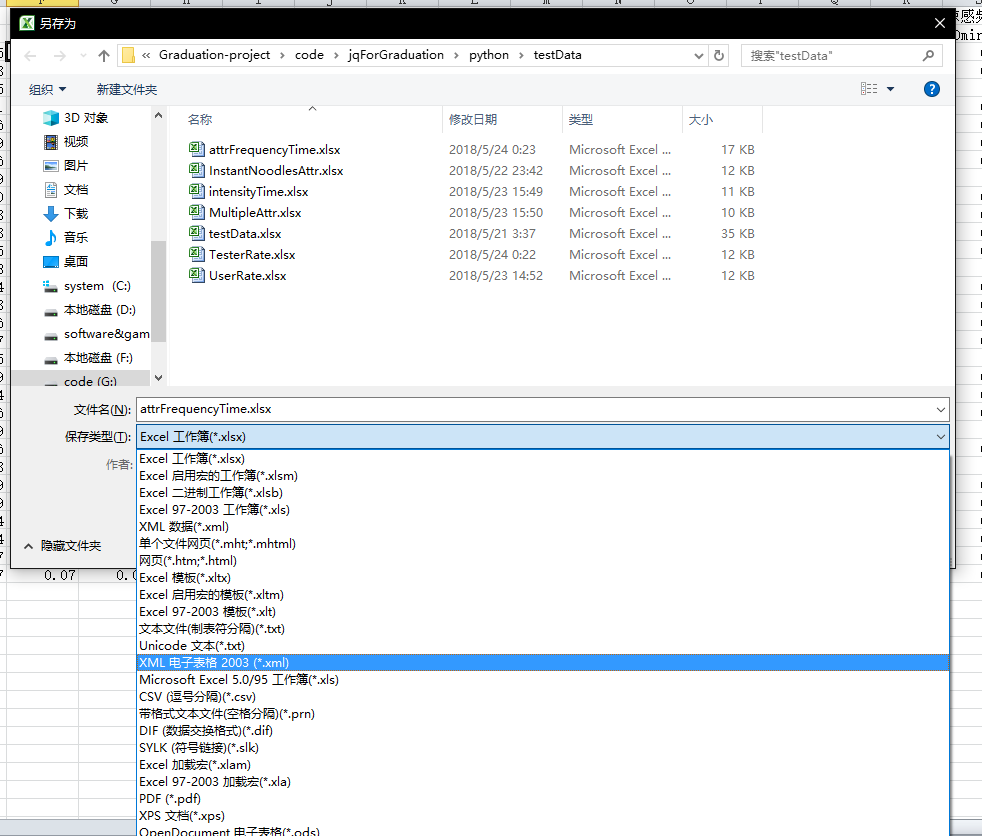
一：数据上传和分析。

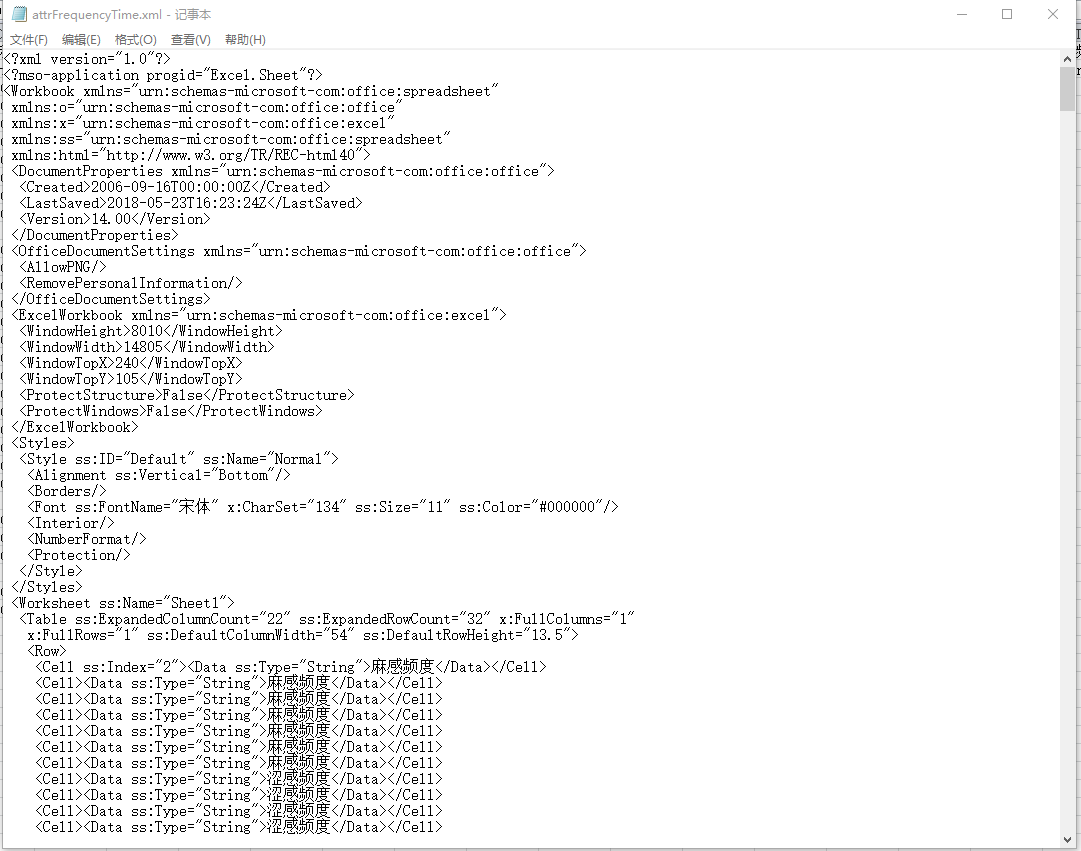
二：数据存储和操作。

## 数据分析

这里我们使用的语言是Python.，Python嵌入PHP后端时使用sys库，在PHP里使用shell\_exec命令调用。

也可以使用Excel自带的功能，在文件保存时更改文件后缀名，如图4.2.1。保存后数据如图4.2.2. 但这样结果未必准确，有一些数据丢失现象。而且解析后的xml文件结构复杂，还需要进行进一步的处理。所以这个办法灵活度不如自己写代码高，这里只做展示，并不嵌入我们的系统中。

图4.2.1

图4.2.2

## 数据库设计

### Mysql设计

建立了六个表，分别是：

|  |  |
| --- | --- |
| attrFrequencyTime | 测试员给出的随时间变化的味道频率评分 |
| file | 管理上传文件 |
| InstantNoodles | 方便面属性 |
| intensityTime | 测试员给出的随时间变化的强度评分 |
| MultipleAttr | 方便面不同指标对应 |
| userRates | 用户评分 |

### Neo4j设计

建立了多种标签的节点，包括：方便面，厂商，测试员，用户，属性等级（三种描述方式），时间点。如图4.3.2.1示是某用户给他食用过的方便面的打分情况。



图4.3.2.1

### ProGRESQL设计

图4.3.3.1

### Db4o设计

图4.3.4.1

### MongoDB设计

图4.3.5.1

## 功能模块设计

### 数据上传



这里文件上传分为了两种，一种是上传单个文件，文件格式包括常见的"doc","docx","xlxs","xls","txt", "jpeg", "jpg", "png"。第二种是文件夹上传，上传的文件夹里必须包括配置文件，以方便对文件夹内所有数据的结构解析。标签远程一栏主要为了方便图片的存储和检索，每个标签对应一个文件夹。存储数据库对应不同的数据存储接口，分流了上传文件使用的存储管理系统。

### 数据显示和操作

数据显示分为几种，第一种MySQL型是一个简单的表格显示页面，可手动配置想看到的属性值，并在前端做一些初级的筛选。图4.4.2.1示是使用DevExtreme做出来的DataGrid表格。



图4.4.2.1

这里做了链接，我们也可以通过点击数据库类型跳转到他们的可视化界面。如MySQL的数据我们也可以在phpadmin上看。示例如图4.4.2.2。



图4.4.2.2

数据显示的第二种是图数据库，或者说是neo4j的可视化界面。效果如图4.4.2.3。可以进行筛选，看到数据节点和连接，也可以在最上方输入行输入cypher语句进行查询。



图4.4.2.3

第三种是对象型数据库db4o。因为它使用Java，而目前系统使用的不是Javaee架构，所以只作为对比示例在eclipse里通过可视化工具插件OMS显示，没并入系统中，效果如图4.4.2.4。

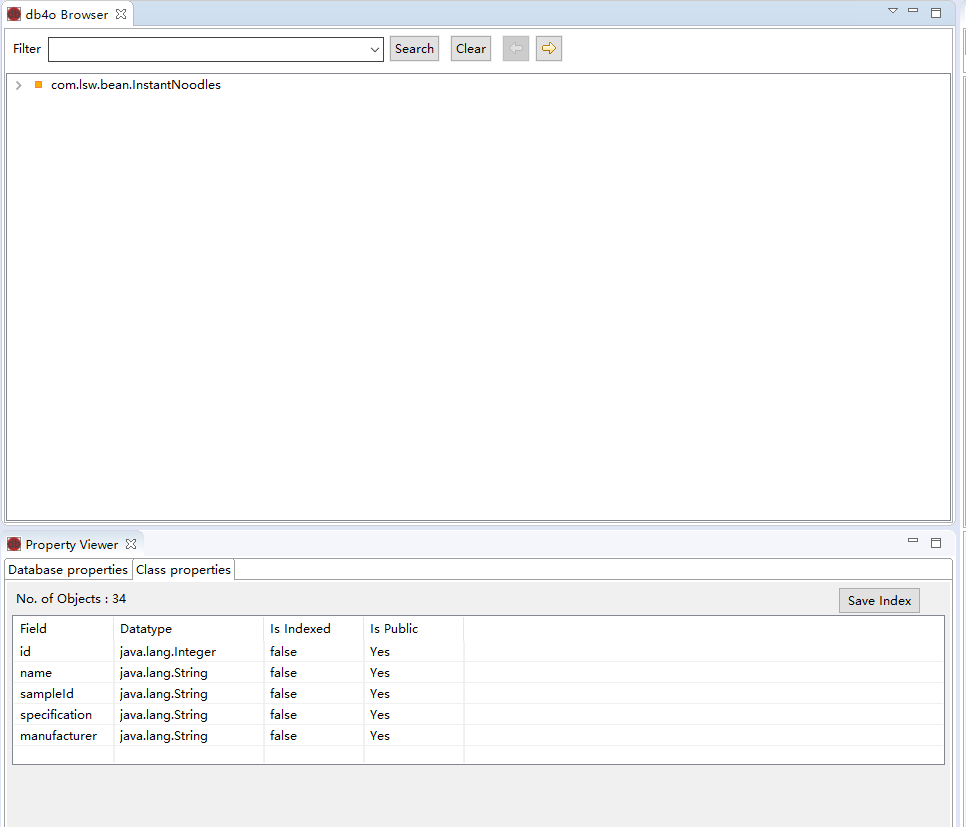


图4.4.2.4

第四种是MongoDB。MongoDB的开发没有使用可视化工具，是直接通过shell操作显示的。示例如：

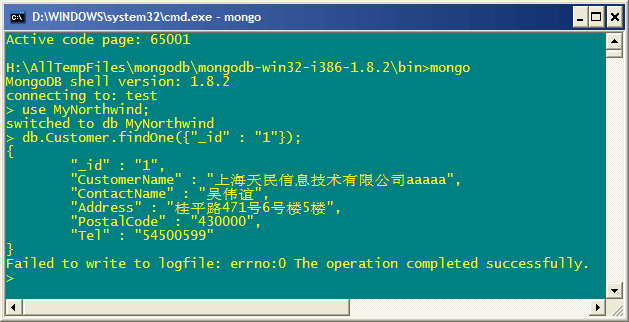


图4.4.2.5

第五种是ProGRESQL。

## 本章小结

本章介绍了系统的架构，配置，功能和设计，简单了展示了针对食品领域的代码成果。

# 总结与展望

本章对全文的主要工作和创新点作了总结，并提出需要进一步研究和优化之处，对未来进行了展望。

## 本文总结

本文通过对食品领域方便面数据的研究，探讨了不同格式不同内容的数据应该使用怎样的分析方式，转换为统一的格式，还对比分析了几种建模思路，不同的数据应该如何进行存储，存储为什么样的数据结构可以方便对数据的操作，在最后也对提出了一点从已知数据（信息）中推测出未知，也就是预测的想法。

### 本文的主要工作

在数据分析（转换）方面，我了解了几种常见的数据格式，并提出了excel/txt转xml/json的一点思路，规定了配置文件的设置和映射方法。在数据存储方面，我学习了多种类型的数据库，学习了他们的思想和使用方法，将其中的几种嵌入了系统中，另几种给出了在该场景下的建模和设计，同时使用工具在时间和空间两个维度进行了性能分析。最后我拿出了一套系统，系统的功能设计体现了数据分析和数据存储的成果。

### 本文的主要创新点

本文的创新点在于对不同类型数据库的建模分析。我找了市面上有名的各种数据库，取其中开源且典型的为例进行开发，并进行性能对比。

## 尚未解决的问题

由于本文场景数据量并不大，所以我没有考虑针对大数据的网格数据库。当子系统越来越多，网络交互越来越多时，也应该考虑深入研究分布式数据库，本文只在MongoDB部分简单提到了分片等横向扩展思路。同时，关于数据库安全性，我并没有做特殊专门的处理。限于篇幅，还有一些数据库如时态数据库，集成数据库也没有介绍分析，这些都是需要可以进一步考虑研究的问题。

## 展望

数据库技术的发展持久不衰，在未来甚至可能将有人工智能数据库，与计算机网络结合的数据库，新数据结构的数据库等，这些都有待继续研究，开发和改进，希望我辈可以在数据库技术的进化中尽一份微薄之力，也希望我的研究可以抛砖引玉，作为他人了解学习的小小参考。

# 致谢

本篇毕业设计论文是在曹旻曹老师的悉心指导下完成的，从选题到设计完成，曹老师对于我的研究和论文提出了很多宝贵的意见，有了曹老师的指导我才没有在研究中迷失方向。曹老师在学术上孜孜探索和求学的精神将会是我今后一直学习的榜样。在生活上，曹老师如邻家大姐姐一般，给予了我很多生活和学习的经验。感谢曹老师的支持和鼓励，不胜感激！希望老师可以一直身体健康，幸福快乐。

感谢关心我支持我的朋友们，感谢与我一起研究、学习的同学们，感谢所有授我以业的老师们，感谢上海大学的栽培，希望上海大学可以建设为世界一流的大学，成为有国际影响力的高水平大学。特别感谢计算机科学与工程学院三年来为我提供的良好学习环境，使我可以不受干扰的自由翱翔于知识的海洋，受益良多，谢谢!

最后，家人的关心是我最坚实的精神支柱，感谢我永远的后盾父母。

# 参考文献

1. [基于XML的通用异构数据交换模型](http://kns.cnki.net/kcms/detail/detail.aspx?filename=SJSJ201008029&dbcode=CJFQ&dbname=CJFD2010&v=" \t "kcmstarget)[J]. 胡能发,唐为萍.  计算机工程与设计. 2010(08)
2. 状态监测大数据存储及聚类划分研究[J]. 周国亮; 宋亚奇; 王桂兰; 朱永利. 电工技术学报. 2013(12)
3. 数据分析技术在中药谱效关系中的应用进展.吕邵娃; 董书羽; 郭玉岩; 孙爽; 匡海学. 中国实验方剂学杂志. 2015
4. 基于XML的Excel数据提取和集成研究[J]. 夏开建,严小泉.计算机应用.2010(04)
5. XML在异构数据交换系统中的研究与实现.陆洲.电脑编程技巧与维护.2012-01-18.期刊
6. MongoDB的形式化模型和研究[D]. 黄伟钗.中山大学 2013
7. Scalable SQL and NoSQL data stores[J] . Rick Cattell. ACM SIGMOD Record . 2011 (4)
8. Native Queries for Persistent Objects, A Design White Paper. [J].William R. Cook , Carl Rosenberger. db4object ,2005
9. NoSQL- WikiPedia. [https://en.wikipedia.org/wiki/NoSQL . 2017](https://en.wikipedia.org/wiki/NoSQL%20.%202017)
10. Modeling MongoDB with Relational Model. Gansen Zhao,Weichai Huang,Shunlin Liang,Yong Tang. Fourth International Conference on Emerging Intelligent Data and Web Technologies (EIDWT) . 2013
11. 附录：部分源程序清单