**哈尔滨工业大学**

**国家示范性软件学院**

**本科毕业设计(论文)中期报告**

**题 目：基于知识库的海量异构数据**

**集成系统的设计与实现**

**专 业 物联网工程**

**学 生 姓 名 李天宝**

**学 号 1133730206**

**联 系 方 式 15704600640**

**年 级 2013级**

**实 习 基 地 上海骇咕赛信息科技有限公司**

**基地指导教师 丁盛豪**

**联 系 方 式 13761793694**

**校内指导教师 王宏志**

**联 系 方 式 13069887146**

**中 检 日 期 2017.03.18**

**哈尔滨工业大学软件学院**

目 录

[1. 毕业设计（论文）内容概述 1](#_Toc317497173)

[1.1 项目来源及开发目的和意义 1](#_Toc317497174)

[1.2 主要开发任务（或总体设计内容及方案） 1](#_Toc317497175)

[1.3 本人所承担任务（模块）说明 1](#_Toc317497176)

[1.4 开发环境和开发工具 1](#_Toc317497177)

[1.5 项目原定进度安排 1](#_Toc317497178)

[2. 中期完成情况说明 2](#_Toc317497179)

[2.1 预定计划的执行情况 2](#_Toc317497180)

[2.2 中期工作说明及成果汇报 2](#_Toc317497181)

[2.2.1 ＸＸＸＸＸＸＸＸＸＸＸＸＸ 2](#_Toc317497182)

[2.2.2 ＸＸＸＸＸＸＸＸＸＸＸＸＸＸＸ 2](#_Toc317497183)

[2.3 存在的困难与问题 2](#_Toc317497184)

[2.4 如期完成预定任务的可能性分析 2](#_Toc317497185)

[2.5 后期工作安排（或进度和计划调整） 2](#_Toc317497186)

[附件：本科毕业设计(论文)中期检查意见表 3](#_Toc317497187)

# 毕业设计（论文）内容概述

## 项目来源及开发目的和意义

本项目来源于上海骇咕赛信息科技有限公司大数据平台数据集成子系统的最新需求。公司长期专注于企业级高并发高可用性技术解决方案和数据分析、数据挖掘领域，主要产品有高并发高可用性分布式缓存集群系统，该项目设计为大型计算集群、为分布式计算服务提供热缓存数据交换。在此系统之上，针对于海量数据整合、处理、分析、学习的需求，公司力主架构于该分布式系统，实现简便、高效的海量数据加工的功能，为该系统设计海量数据快速处理的API。作为数据处理平台，数据集成便是其中不可缺少的一项，本人的项目即基于此内容进行一些创新性的研发。

数据集成涉及将不同来源中的数据整合，并统一地向用户展示这些。数据集成在很多领域上非常重要，包括商业（当两个相似的公司需要合并它们的数据库）、科学（组合来自不同生物信息学数据库的研究结果）。 随着数据量和数据分享需求爆炸式的增长，数据集成出现的频率越来越高。它已成为大范围理论工作的焦点，然而许多问题仍然没有解决。

上文中，维基百科的介绍已经暗示了数据集成的内容与重要性。简言之，数据集成是一个将具有不同概念、上下文、逻辑关系的数据文本进行合并，形成一个具有统一模式的数据集。作为数据分析和运用的基础，计算机领域内数据集成具有重要的意义，包括数据清洗、模式识别、生物信息等等。然而，随着大数据时代的到来，数据每天都在被生成、分析并且大量的应用着，数据驱动的决策也成为了社会中不可或缺的一部分。现如今，互联网上分散存在着海量的数据，为了充分利用这些数据中的信息以及蕴含的价值，将这些数据有效的集成在一起成为了一个显著的需求。

大数据时代下的数据集成，与传统数据集成的考核点（数量、速度、多样性、真实性）有着一定程度上的不同。首先，单一数据源无法为单一目标储存如此大量的数据。其次，很多数据源是动态的，总会有很多数据源在生成、消失，或者某些数据源改变了储存方式。第三，在诸多数据源之间，我们很难保证他们是完全统一的，这就造成了异构数据源的产生，并且相同条目下的数据也可能重复或者存在精度、修改时间等方面的差异。

## 主要开发任务（或总体设计内容及方案）

为了将海量的异构、异源数据进行集成，首先需要做的就是将不同数据库的模式进行集成，生成一个全局的数据库模式，进而方便数据库记录的填充和数据库的融合。对于数据集成问题，传统的方法往往是预先指定一个全局的数据库模式，然而针对于海量数据的背景，人们难以在大量的数据中捕捉全局的信息来设计预定的全局模式，并且建立全局模式和每一个数据库模式之间的匹配关系也是耗时耗力的。

因此，在上述背景下，通过设计一些合理匹配关系和高效的算法，省时、准确的生成一个全局数据库模式成为了迫切的需求。

## 本人所承担任务（模块）说明

本人在整个项目中的具体任务是属于海量数据快速处理平台的数据集成系统，具体的工作任务是设计并实现模式集成子系统。这项任务包括设计完成模式集成的高效、精准的算法，使用现有公司数据进行测试，系统实现并整合到公司系统中等。

模式集成的主要工作是针对异源、异构数据表的模式，在形式和语义的两个维度将它们进行集成，从而得到一个统一的模式，既能将多个数据源中的所有属性全部包含，又能保证产生的数据模式中属性彼此不重复，满足数据库的范式要求。

## 开发环境和开发工具

项目涉及到的开发环境如下：

操作系统：Windows/Linux/MacOS

处理器：Intel Core i7，2.40GHz主频或以上

内存：8GB 1333 MHz DDR3

程序运行环境：以Linux/MacOS为主，开发所涉及的语言和库应在其他各个平台（包括但不局限于Windows）通用，不受系统环境限制

注：以上开发环境均为前期开发要求，当集成到公司系统时，应使用更高性能的服务器，具体服务器配置在此保密

如表 1‑1 开发工具表所示，本系统实际开发郭恒开发使用到了这些开发工具。

表 1‑1 开发工具表

|  |  |  |
| --- | --- | --- |
| 工具类别 | 工具名称 | 作用 |
| 集成开发环境 | Visual Studio 2013  Xcode | 程序最主要的开发、调试平台 |
| 脚本开发工具 | Atom | 数据预处理、后处理脚本开发平台 |
| 界面设计工具 | Photoshop CC2015 | 图片素材设计 |
| 前端开发工具 | PyCharm | 前端界面的实现 |
| 版本控制软件 | Github | 版本控制 |

## 项目原定进度安排

项目原定进度安排见表 1‑2 项目原定进度工作计划表。

表 1‑2 项目原定进度工作计划表

|  |  |  |  |
| --- | --- | --- | --- |
| 起始时间 | 完成时间 | 计划工作内容 | 备注 |
| 2016.07.18 | 2016.07.31 | 配置公司要求的环境，熟悉集群 | 已完成 |
| 2016.08.01 | 2016.08.28 | 学习所需基础知识 | 已完成 |
| 2016.08.29 | 2016.09.11 | 分析现有系统，寻找创新点 | 已完成 |
| 2016.09.12 | 2016.09.18 | 选择研究方向 | 已完成 |
| 2016.09.19 | 2016.10.09 | 验证选题的可行性 | 已完成 |
| 2016.10.10 | 2016.11.06 | 研读相关工作现有论文，进行比较 | 已完成 |
| 2016.11.07 | 2016.11.20 | 撰写开题报告，准备开题答辩 | 已完成 |
| 2016.11.21 | 2016.12.04 | 寻找知识库，根据需要进行处理 | 已完成 |
| 2016.12.05 | 2017.01.01 | 完成形近整合部分算法的设计 | 已完成 |
| 2017.01.02 | 2017.02.05 | 完成语义整合部分算法的设计 | 已完成 |
| 2017.02.06 | 2017.03.05 | 使用少量数据进行试验，验证准确度 | 已完成 |
| 2017.03.06 | 2017.03.18 | 准备中期答辩 | 在进行 |
| 2017.03.19 | 2017.04.01 | 用大量数据进行试验，验证效率 | 未完成 |
| 2017.04.02 | 2017.04.22 | 根据实验结果调整系统 | 未完成 |
| 2017.04.23 | 2017.05.13 | 设计、实现前端界面 | 未完成 |
| 2017.05.14 | 2017.05.27 | 进行最后的验证，完善系统 | 未完成 |
| 2017.05.28 | 2017.06.20 | 撰写论文，参加毕业答辩 | 未完成 |

# 中期完成情况说明

## 预定计划的执行情况

开题至中期答辩前时间段的工作如表 2‑1开题至中期工作计划表。

表 2‑1开题至中期工作计划表

|  |  |  |  |
| --- | --- | --- | --- |
| 起始时间 | 完成时间 | 计划工作内容 | 备注 |
| 2016.11.21 | 2016.12.04 | 寻找知识库，根据需要进行处理 | 已完成 |
| 2016.12.05 | 2017.01.01 | 完成形近整合部分算法的设计 | 已完成 |
| 2017.01.02 | 2017.02.05 | 完成语义整合部分算法的设计 | 已完成 |
| 2017.02.06 | 2017.03.05 | 使用少量数据进行试验，验证准确度 | 已完成 |

在2016.11.21-2017.03.05的工作时间段期间，已经将开题答辩时计划的工作逐项完成，包括预处理知识库、设计型近整合算法、设计语义整合算法、筛选部分有代表性的数据进行测试。由于开题时对领域了解不足，有些情况未考虑在内，在实际的工作中添加了一些额外的工作，诸如知识库-实验数据匹配、实验数据筛选、数据预处理优化等，这些具体的工作将在中期工作说明及成果汇报中详细介绍。

## 中期工作说明及成果汇报

到中期为止，预定计划中的哪些部分已经完成，详细描述具体完成过程和做法；目前能够展示的成果有哪些，需要写出具体内容。

**该部分应该是整个中期报告的重点内容！**

一般包括：系统详细设计（根据不同的项目内容可以写算法设计、功能设计、数据库设计/数据结构设计等）、大部分的系统实现过程及结果（进度快的话也可以写系统测试），根据需要可能会用到的模型有：功能结构图、用例详细描述、设计类图、包图、组件图、时序图、程序流程图、ER图、数据字典、实现结果截图等…

本节中将对现有完成工作进行详细的说明，包括但不局限于系统设计、算法设计、对应模型、成果截图。

### 系统功能结构

本系统根据功能划分主要包含5个模块：用户界面模块、预处理模块、形式整合模块、语义整合模块和全局模式模块。具体的功能结构图如图 2‑1 系统功能结构图所示。



图 2‑1 系统功能结构图

### 系统流程

整个系统以多个数据库的模式为输入，以现有知识库为依托，经过形式整和语义整合，生成全局属性。系统通过前端界面对以上过程的结果及必要的中间输出进行体现。该系统的流程如图 2‑2 系统流程图所示。

图 2‑2 系统流程图

系统的用户提供待集成的模式k个作为系统的输入，处理过程中经历了如下的流程：

1. 形式整合：将形式上相近的属性进行mapping，将在给定阈值范围下的属性归为一组（这种情况一般发生在拼写错误的情形下）
2. 语义整合：将语义上相近的属性进行mapping，将在给定阈值范围下的属性归为一组。这一过程利用了外部的知识库（经过与处理）
3. 生成全局属性：利用前两个步骤生成的属性组，在每组相似的属性中选出一个代表属性，并将其他属性在给定的k个模式中取出并取并，从而得到全局模式

### 背景概念及定义

#### 语义相似判定方法

在进行模式集成的过程中，很重要的一个问题就是将相似的属性归为同一条属性，避免在最后生成的全局属性中产生冗余。然而一个很关键的问题就是如何判定两个属性是相似的，这个概念对于人类来说很好理解，但是交给计算机来做确是非常困难的，其中一个重要原因是人类对于判断是那么样的属性是相似的具有先验知识，这个只是实在人们的成长过程中逐步积累起来、加以自己的分析和理解的，而计算机没有这样感性的能力。为了解决这个问题，在我们的系统中引入了外部的知识库，尽可能通过这种方式使计算机的计算过程模拟人脑的思维。

目前阶段我们采用了*Freebase*[[1]](#footnote-1)知识库，这是一个表示“is-a”关系的知识库，能够表示出相邻的两个知识概念就有类别从属或者包含的关系，相似的知识库还有*WordNet[[2]](#footnote-2)*, *Probase[[3]](#footnote-3)*和*YAGO[[4]](#footnote-4)*。这样的知识库整体的结构是一个图*G*，图中的每一个节点是一个概念（代表着客观存在的物体或者抽象的类别），边集*S*中的每条边*e*连接的两个节点具有从属或者包含的关系，该结构可以表示为：

虽然同义词数据库可以直接的表达我们希望得到的关系，但是现有的同义词数据库并不完善，并且在某些专业领域内，并不能简单地通过同义词来判定两个属性是相似的。而这种“is a”关系型的知识库一定程度上可以解决我们的需求，尤其是*Freebase*等在专业术语范围内有着很好的描述能力。当然采用这样形式的知识库不是唯一的，例如*NGD* (normalized Google distance)[[[5]](#endnote-1)]等均可以提供类似的信息。因此我们的目标是设计一种统一的衡量方式和集成算法，在后期需要时可以很好地将这些知识库应用到我们的系统中。

基于上文中对语义知识库的定义，为了将这种定性的知识引入到我们的算法中，需要以一种定性的方式对理解性的概念进行表示，这里对语义距离给出定义：

可以理解为*a*与*b*之间的距离可以表示为两点之间路径的长度（包含边的个数）。尽管这种表示方式可能不是非常合理，但是目前阶段取得的实验效果可以验证这种定义的准确性，后期的优化过程中可以为不同的边赋予不等的长度。在这种定义下，实际的过程中需要用户指定语义近似阈值*γ*，当两个概念之间的距离小于γ时，可以看做这两个概念是近似的。

#### 形近相似判定方法

对于形式近似的情况，我们采用最基础的编辑距离算法，并由用户给定*εt*作为其阈值。执行编辑距离算法时，我们为给定模式中的属性和知识库中的概念按照q-gram生成倒排表（长度*q*由用户给定）。具体来说，我们将词以*q*为单位长度进行划分，当两个词有足够多的分片相同时，他们之间的差异便会足够小，可以认为他们是相似的。我们的方法采用*Count Filtering*的计算方式，具体来说*a*与*b*之间如果具有*LBab*个相同的q-gram分片，我们可以认为*a*与*b*在拼写上是相近的，其中

### 数据结构设计

#### Cluster Set

在系统运行过程中需要考虑语义相似和拼写相似，而其中利用知识库的语义相似的部分其实是相对来说更困难的。由于知识库巨大，我们不可能将知识库完全导入到内存中，在寻找相似概念的时候也不可能遍历整个知识库。因此我们的想法是在知识库上利用类似于数据库中的join算法，只在待集成模式中的属性附近去寻找语义相似的概念。同时为了减少遍历次数，尽量使基于拼写相似的集成同步完成，我们设计了如下的数据结构*Cluster Set*，这种结构适用于算法执行过程中的中间结果和系统最后的输出。

**[*Cluster Set*]**

*S* as the concepts set of the knowledge base, a cluster set is a set of pairs {*U*, *SU*}, where *U* is a set of attributes and

*Cluster Set*实际上是一个用来储存经过匹配后的集合，其中*U*集合储存着被下文中算法认定为详细的属性，而为了加速这个算法的运行，*SU*中储存的是一些属性，每一个至少和和*U*中的一个元素距离小于*ε*。这些属性是可能在接下来的集成过程中归并到集合*U*中的，相比于每次在知识图上进行匹配，提前将可能用到的属性以较小的代价储存下来，以空间换时间，能够提升算法的运行效率。

#### Neighbor Table

由于数据量巨大（模式中的属性和直属库中的概念共计在100,000,000条以上），很难将数据长期储存在内存中，因此该系统和内部的算法都将是基于外存的，所以如何将数据读取的时间控制在一定范围内将是一个核心问题。通常情况下外存的系统能够高效的运行取决于数据在磁盘上如何组织，hash和B树成为了比较常见的选择。在该数据集成系统中，为了方便高效采用hash的方法将知识库中的边*e*(*a*, *b*)储存，具体来说是分别对应起点*a*和*b*，在其hash值对应的空间内储存对应终点*b*和*a*以及其他的如边的长度等信息。

由于我们的算法实际上是在知识库中进行边的连接，形成多条路径 (*start*,*end*,*length*)，对于不同的路径，只要它们拥有共同的终点*end*，便可以与边*e*(*end*, *b*)进行连接。因此可以将拥有共同终点的路径归为一类，多条路径只需做一次相同的扩展以提升效率，系统中使用数据结构*Neighbor Table*，定义如下：

**[*Neighbor Table*]**

*t* is an attribute and *P* is the set of all paths in the knowledge base. *Hk*(*t*) is a table on the disk indexed by hash value of string *t*, s.t.

*Neighbor Table*接收知识库中的一个概念*t*，并返回所有以*t*为终点、长度为*k*的路径，这样的结构可以以*t*为关键字形成一个hash表，并将生成和访问的时间保持在一个较低的数量级。类似的，边集*E*在这里可以表示为*H1*(*t*)。

### 核心算法

本系统做要处理的核心问题是在知识库中寻找语义上相近的概念，对于模式中被判断为语义相似的归并成一条。然而知识库中所含有的概念众多，在每次搜索时将其遍历一遍是不现实的，知识库中绝大多数的信息在单次匹配中是用不上的。因此我们借鉴了数据库中的join运算符，将其思路应用到我们的系统中。

为了对语义相似的属性进行匹配，本系统首先定义了一个Semantic Join操作符⋈，主要任务是将满足如下三条之一的属性认为是语义相似的属性，将这些属性进行归并，并根据新生成的*U*集的结果维护*SU*，其定义如下：

**[*Semantic Join Operator*]**

Given two families of cluster sets *R*, *T*, and a threshold *d*, two elements (*U1*, *S1*) and (*U2*, *S2*) are from *R* and *T*, respectively are semantically joined if they satisfy one of the following constraints.

The result of Semantic join on (*U1*, *S1*) and (*U2*, *S2*) is a pair (*U*, *SU*), where

这里的*γ*是用户根据数据具体情况给定的一个语义阈值，具体来说是用来限定知识库上被认定为相似的概念的距离上限。目前阶段这个值只能通过对知识库进行分析而给出，未来希望可以通过机器学习、关联挖掘等手段进行计算。从另一个角度来说，在给定目标属性集*R*和知识库边集*E*的前提下，Semantic Join算法的功能需求是将知识库上*γ*距离内的属性进行归并，其过程可抽象为下面的公式：

将上述功能映射到知识库的图结构时，可以理解为是从目标属性集*R*起始将边逐步进行连接成长度在1-*γ*之间的路径，这些路径的起始于*R*中的属性，重点属性可以看作是与*R*中属性详细的属性。

Semantic Join算法的具体流程如图 2‑3 Semantic Join算法所示。

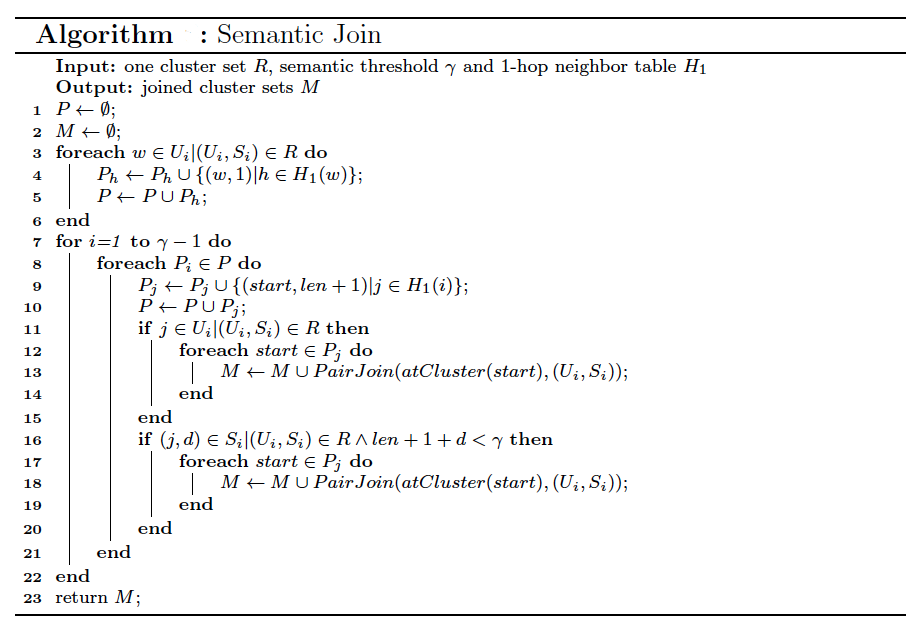


图 2‑3 Semantic Join算法

Semantic Join算法接收的输入变量有：

* *R*：待集成的属性集合（以cluster set形式储存）
* *γ*：用户指定的语义距离阈值
* *H1*：知识库的边集（以neighbor table形式储存）

该算法大体可以分为以下三个阶段：

1. 初始化（1-6行）：这一阶段扫描输入的属性集*R*，并将以*R*中的每一个属性*r*起始的路径加入到路径集合*P*中，其中*P*的定义如下：

**[*Path Set*]**

*Pa* is a path set, all paths in which share the same end node *a*, s.t.

当前过程得到的路径集合只包含了以目标属性开头的边，并按照其终点进行分组。

1. 路径扩展（7-10行）：这一阶段在知识库中执行一次join操作，即对于现有得到的属性集合再向外扩展一个单位距离。整个算法共计执行*γ*-1次join操作，使得能够包含相距至多为*γ*的概念，即满足给定阈值范围的相似属性。
2. 集合维护（11-20行）：这一阶段是将相似属性所处的集合合并，即已知*R1*中的*r1*和*R2*中的*r2*相似，需要将*R1*和*R2*合并，包括合并*U*集合和维护对应的*Su*集合。

## 存在的困难与问题

… …

## 如期完成预定任务的可能性分析

… ….

## 后期工作安排（或进度和计划调整）

写出后半期的工作进度、时间安排，以及对最初计划的调整情况

… …

# 附件：本科毕业设计(论文)中期检查意见表

|  |  |
| --- | --- |
| 基地导师意见（需写具体内容） | |
| [若导师不能亲笔签字，则将有“同意参加中检”字样的邮件整个截图（图中应能看到导师的关键信息）打印附到报告最后]  *【P2不需填写】*    签 字：  年 月 日 | |
| 校内导师意见（需写具体内容） | |
| [若导师不能亲笔签字，则将有“同意参加中检”字样的邮件整个截图（图中应能看到导师的关键信息）打印附到报告最后]  签 字：  年 月 日 | |
| 中期检查小组意见 | |
| 结论：◎ 通过  ◎ 警告  ◎ 不通过  具体意见： | 评委签字：      年 月 日 |

# 附件：导师意见邮件截图

注：1.对于亲笔签字的情况不需要此内容；

2.邮件截图中必须含有发件者信息、发件时间、意见等内容。

1. https://www.freebase.com/ [↑](#footnote-ref-1)
2. http://wordnet.princeton.edu/ [↑](#footnote-ref-2)
3. http://research.microsoft.com/en-us/projects/probase/ [↑](#footnote-ref-3)
4. http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/ [↑](#footnote-ref-4)
5. Cilibrasi, Rudi and Paul M. B. Vitányi. “Automatic Meaning Discovery Using Google.” Kolmogorov Complexity and Applications (2006). [↑](#endnote-ref-1)