第1章 概述

## 1.1 选题背景及意义

近些年来，计算机技术以及信息处理技术的飞速发展，各行各业建立了很多信息系统，积累了大量的数据，因此我们进入了大数据的时代[1]。为了使信息系统中的数据更加准确、一致，能够支持正确决策，就要求所管理的数据可靠无误，能够准确地反映现实世界的状况。数据是信息的载体，好的数据质量是各种数据分析如OLAP(Online Analytical Processing)分析、数据挖掘等能够得到有意义结果的基本条件。因此，企业数据质量的管理正在获得越来越多的关注。人们常常抱怨所谓的“数据丰富，信息贫乏”，其中一个原因是缺乏有效的数据分析技术，而另一个重要原因则是数据质量不高，这往往导致不能有效地对数据加以利用。数据质量管理如同产品质量管理一样贯穿于数据生命周期的各个阶段，但目前尚缺乏一个标准化的解决思路。数据质量的研究由来已久，涉及到统计学、人工智能、数据仓库等各个领域。现在，我们习惯于“用数据说话”，我们比任何时候都更关心数据的分析以及应用。但是由于各种原因，例如，数据录入错误，数据拼写错误，数据来源不同等，我们获取到的数据通常是以各种形式出现的脏数据，例如数据拼写错误，数据重复，数据空缺等，导致了数据的质量低下，对于大数据分析和研究的结果受到影响[2]，进而可以导致决策的失误等，因此数据质量问题的解决是十分重要的。人们对数据质量的研究由来已久，并且提出了各种的方法希望能够解决数据质量问题[3]。但是直到现在，对于数据质量问题的解决还没有一个成体系的方法。

由于人工智能和知识工程的发展，本体在计算机领域受到了空前的重视，因为人工智能和知识工程需要开发一个领域共享的，公共的概念来实现知识的共享和重用，而本体是“共享的概念模型的明确的规范化的说明”满足这一特性[4]。计算机领域的专家们开始使用本体来解决计算机领域的各种问题，例如概念建模等等。由于本体所具有的可以共享和重用的特性，考虑到大数据时代的数据量大，数据应用范围广等特点，我们提出了使用本体的方法来解决数据质量的问题。

数据质量本体是一种领域本体，基于本体来构建的数据质量评估，不依赖与特定的应用和特定的环境。使得数据质量本体具有在数据评估领域具有很强的通用性，解决了以前所使用的关系模型表示的数据评估的不通用的问题。使用关系模型所解决的数据评估问题，都是针对特定的应用和特定的应用环境，而且对于数据质量的评估过程中的复杂约束问题和简单约束问题必须分开处理，导致了数据评估过程的复杂性，同时在对于不同来源的数据或是不同应用的数据，无法进行共享评估。由于本体是“共享概念模型的明确的形式化规范说明”，所以数据质量本体在数据的评估领域是通用的。

数据质量本体是对数据评估领域知识的建模，是对数据评估领域知识的规范化模式。主要是抽取了数据质量的评估规则，并且对这些规则进行抽象，提取来构建数据质量本体。通过对数据评估领域知识的规范化，形式数据评估领域的评估标准，使得在数据评估领域有一套可以通用，共享的数据评估标准。

数据质量本体构建及应用研究是针对数据质量出现的问题，利用本体的思想和方法来构建数据质量本体。通过数据的评估规则约束，例如非空约束，值域约束，逻辑依赖约束，存在性依赖约束，等值约束，等值一致性约束，逻辑一致性约束等，抽取出数据质量本体中的属性。利用数据质量本体，对数据库中的数据进行质量评估，选出其中的不合格数据。所以数据质量的评估对于数据的分析以及研究具有很大的现实意义，能够选出不合格的数据，分析数据质量的高低。

## 1.2 国内外研究的现状

在本体研究的初期，构建本体的过程都由人手工完成的，即传统手工构建方 法。这种方法首先需要明确本体的应用领域和范围，再人工地提取领域中的概念， 并确定这些概念间的关系和层次，然后再利用现有的本体编辑工具进行本体构建， 最后通过评估对所构建的本体进行进一步的修改和完善。本体编辑工具仅仅是为 用户提供了一个本体的编辑环境，在本体概念的获取，概念层次关系的确立，这 都还需要领域专家的参与。随着本体研究的全面发展，产生了一些利用现有资源 和技术来半自动、自动化构建本体的方法[8]。 目前国内外对基于关系数据库的半自动、自动化本体构建进行了大量的理论 研究和应用实验。关系数据库资源的知识表示的形式是统一的，无论是概念还是 关系都是用一张二维表表示的，这个特点有利于本体的抽取与构建。现阶段的研 究主要集中在通过分析关系模式的语义信息来抽取本体构建时所需要的概念与关 系。现在获取的大量的数据都是存储于数据库中，鉴于目前还没有一个完全系统的方法来进行数据质量的评估，所以使用本体的方法进行数据质量的评估。

国外关于数据质量的研究起步较早，在数据质量框架和数据质量技术手段与 方式等方面都做了大量的创新性研究。随着全球经济的迅速崛起，在信息化迅速 发展的今天，国际上一些企业巨头非常重视企业的信息化建设，把数据作为企业 逐步壮大的重要资产，而数据质量更是企业发展、决策的核心。全球很多国际化 的企业已经充分认识到了数据质量的重要性，每年都投入大量的人力、物力来提 高企业的数据质量，以提升企业的决策管理水平。目前一些企业的数据建设已经 实现了企业信息集成和智能决策阶段。本文将基于jess对数据质量进行评价研究。

第2章 **本体及相关技术**

## 2.1 OWL 本体语言简介

本体最早是由哲学领域产生的，是人们对于存在的探究。在人工智能界，最 早给出本体定义的是 Neches 等人，他们将 Ontology 定义为“给出构成相关领域词 汇的基本术语和关系，以及利用这些术语和关系构成的规定这些词汇外延的规则 的定义”。Neches 认为：“本体定义了组成主题领域的词汇表的基本术语及其关系， 以及结合这些术语和关系来定义词汇表外延的规则。”（“An ontology defines the basic terms and relations comprising the vocabulary of a topic area, as well as the rules for combining terms and relations to define extensions to the vocabulary.”）。后来在信 息系统、知识系统等领域，越来越多的人研究 Ontology，并给出了许多不同的定义。其中最著名并被引用得最为广泛的定义是由 Gruber 提出的，“本体是概念化的 明确的规范说明”。所以，本体就是由若干概念及其在某种逻辑理论（如一阶谓词 演算）支持下的定义所构成的一种分类法。有人已经为本体提出的定义就是，本 体是一种对于某种概念体系（概念表达、概念化、概念化体系或者说概念化过程） 的明确而又详细的说明。对于特定一个领域而言，本体表达的是其那套术语、实 体、对象、类、属性及其之间的关系，提供的是形式化的定义和公理，用来约束 对于这些术语的解释。本体允许使用一系列丰富的结构关系和非结构关系，如泛 化、继承、聚合和实例化，并且可以为软件应用程序提供精确的领域模型。例如， 本体可以为传统软件提供面向对象型系统的对象模式（object schema），以及类的 定义。

## 2.2 Protégé本体建模工具

Protégé[13]是斯坦福大学为知识获取而开发的一个工具，主要应用于知识的获取 以及现存本体合并和排列，可以免费下载并公开源代码，再加上其支持中文， Protégé已经成为目前国内使用最为广泛的本体编辑工具和基于知识的框架。 Protégé主要具有以下特征[14]： （1）可扩展的知识模型能够使用户重新定义原始知识集合； （2）友好的本体导入导出功能，可以从 RDFS、带 DTD 的 XML 文件、 XMLSchema 等文件中导入本体，也可以将本体转化为多种形式化语言描述，如 RDF(S)、OWL 等； （3）具有友好的开发界面； （4）具有强大的功能插件体系和开放的模块化风格，基于开放式组件的体系 结构使系统开发者可以通过生成恰当的插件以增加新的功能； （5）提供一个半自动化工具 PROMPT 用于自动地执行本体的合并和排列； （6）Protégé平台支持两种类型的本体建模。 Protégé- Frames 编辑器用于构建基于框架的本体。在这种模型中，本体是由具 有层次结构的类集合组成，类的槽(slots)集合表示概念的属性和关系，类的实例集 合则表示概念的具有特定属性值的个体。 Protégé- OWL 编辑器则用于构建应用于 语义网的本体，专门使用 W3C 的 OWL 语言描述,一个 OWL 本体包含类的描述、 属性以及实例。 Protégé因其简单易用性、不断升级的品质、免费获取的特性以及强大的功能 可扩展性使之成为国内最受欢迎的本体构建工具之一。

## 2.3 Jena技术

Jena 的 ModelFactory类是创建不同类型模型的首选方式。在这种情况下，您想要空的、内存模型，所以要调用的方法是ModelFactory.createDefaultModel()。这种方法返回 Model实例，您将使用它创建表示家庭中每个成员的 Resource。创建了资源后，可以编写关于这些资源的语句并添加到模型中。

在 Jena 中，语句的主题永远是 Resource，谓词由 Property表示，对象是另一个 Resource或常量值。常量在 Jena 中通过 Literal类型表示。所有这些类型共享公共接口 RDFNode。将需要四个不同的 Property实例表示家庭树中的关系。这些实例使用 Model.createProperty()创建。

将语句添加到模型中的最简单方法是通过调用Resource.addProperty()。此方法以 Resource作为主题在模型中创建语句。该方法使用两个参数，表示语句谓词的 Property和语句的对象。addProperty()方法被过载：一个过载使用 RDFNode作为对象，所以可以使用 Resource或 Literal。还有有益过载，它们使用由 Java 原语或String表示的常量。在示例中，语句的对象是表示其他家庭成员的Resource。

Jena 的 com.hp.hpl.jena.rdql包包含在 Java 代码中使用 RDQL 所需的所有类和接口。要创建 RDQL 查询，将 RDQL 放入 String中，并将其传送给 Query的构造函数。通常直接设置模型用作查询的源，除非在 RDQL 中使用 FROM子句指定了其他的源。一旦创建了 Query，可以从它创建 QueryEngine，然后执行查询。

给定了本体和模型后，Jena 的推理引擎可以派生模型未明确表达的其他语句。Jena 提供了多个 Reasoner类型来使用不同类型的本体。因为要将 OWL 本体与 WordNet 模型一起使用，所以需要 OWLReasoner。

下例显示了如何将 OWL WordNet 本体应用到 WordNet 模型自身以创建推理模型。这里我实际将使用 WordNet 模型的子集，仅包含下位词层次结构中“plant life”之下的那些名词。仅使用子集的原因是推理模型需要保存在内存中，WordNet 模型对于内存模型过大而不能实现。我用来从整个 WordNet 模型中提取 plants 模型的代码包含在文章来源中，名为 ExtractPlants.java

首先从ReasonerRegistry中获得OWLReasoner。ReasonerRegistry. getOWLReasoner()在它的标准配置中返回 OWL reasoner，这对于此简单情况已经足够。下一步是将 reasoner 与 WordNet 本体绑定。此操作返回可以应用本体规则的 reasoner。然后，将使用绑定的 reasoner 从 WordNet 模型创建 InfModel。

从原始数据和 OWL 本体创建了推理模型后，它就可以像任何其他 Model实例一样进行处理。因此，如清单 16 所示，通过 FindHypernym.java 与正常 Jena 模型一起使用的 Java 代码和 RDQL 查询可以重新应用到推理模型，而不进行任何更改。

## 2.4 jess介绍

Jess是[Java](http://baike.baidu.com/item/Java/85979" \t "http://baike.baidu.com/_blank)平台上的[规则引擎](http://baike.baidu.com/item/%E8%A7%84%E5%88%99%E5%BC%95%E6%93%8E" \t "http://baike.baidu.com/_blank)，它是CLIPS[程序设计语言](http://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1%E8%AF%AD%E8%A8%80" \t "http://baike.baidu.com/_blank)的超集，由Sandia国家实验室的Ernest Friedman-Hill开发。它的第一个版本写于1995年晚期。

Jess提供适合自动化专家系统的逻辑编程，它常被称作“专家系统外壳”。近年来，智能代理系统也在相似的能力上发展起来。

与一个程序中有一个只运行一次的循环的命令式编程语言不同，Jess使用的声明式编程通过一个名为“模式匹配”的过程连续的对一个事实的集合运用一系列规则。规则可以修改事实集合，或者运行任何Java代码。

Jess可以被用来构建使用规则定义形式的知识来推倒结论和推论的Java Servlet、EJB、Applet和应用程序。因为不同的规则匹配不同的输入，所以有了一些有效的通用匹配算法。Jess[规则引擎](http://baike.baidu.com/item/%E8%A7%84%E5%88%99%E5%BC%95%E6%93%8E" \t "http://baike.baidu.com/_blank)使用Rete算法。

Jess非常小巧、灵活，并且是已知[规则引擎](http://baike.baidu.com/item/%E8%A7%84%E5%88%99%E5%BC%95%E6%93%8E" \t "http://baike.baidu.com/_blank)中最快的。核心Jess语言与CLIPS依然兼容，许多Jess脚本也是有效的CLIPS脚本。与CLIPS一样，Jess使用Rete[运算法则](http://baike.baidu.com/item/%E8%BF%90%E7%AE%97%E6%B3%95%E5%88%99" \t "http://baike.baidu.com/_blank)来处理规则。Rete是一个非常高效的算法，用于解决复杂、艰深的多对多匹配问题。在CLIPS的基础上，Jess添加了许多特性：后向[链表](http://baike.baidu.com/item/%E9%93%BE%E8%A1%A8" \t "http://baike.baidu.com/_blank)、[运行内存](http://baike.baidu.com/item/%E8%BF%90%E8%A1%8C%E5%86%85%E5%AD%98" \t "http://baike.baidu.com/_blank)查询以及操作和直接推理Java对象的能力。Jess同时也是一个强有力的Java脚本环境。通过它你能够创建Java对象、调用Java方法而无需编译任何Java代码。

## 2.5 ontology与数据质量

本体在计算机领域的定义经历了一个演化过程。在人工智能界，最早给出本体定义的是Neches等人，他们将本体定义为“给出构成相关领域词汇的基本术语和关系，以及利用这些术语和关系构成的规定这些词汇外延的规则的定义[19]”。1993年，Gruber给出了本体的一个最为流行的定义，即“本体是概念模型的明确的规范说明[20]”。后来，Borst在此基础上，给出了本体的另外一种定义：“本体是共享概念模型的形式化规范说明”。

Studer等人在对本体做了深入研究后，扩展了Gruber的定义，认为“本体是共享概念模型的明确的形式化规范说明（An ontology is a formal, explicit specification of a shared conceptualisation.）”。该定义包含四层含义：

1）概念模型（Conceptualization），即本体是通过抽象客观世界的概念而得到的模型，它表示的含义独立于具体的环境状态；

2）明确性（Explicit），即本体所使用的概念及在这些概念之上的约束都有明确的定义，没有二义性；

3）形式化（Formal），即本体是计算机可处理的，而不是自然语言；

4）共享（Shared），即本体体现的是共同认可的知识，反映的是相关领域中公认的概念集合，它所针对的是团体而不是个体。

本体的目标是捕获相关的领域的知识，提供对该领域知识的共同理解，确定该领域内共同认可的词汇，并从不同层次的形式化模式上给出这些词汇和词汇之间相互关系的明确定义

## 2.6 熵权法

按照信息论基本原理的解释，信息是系统有序程度的一个度量，熵是系统无序程度的一个度量；如果指标的信息熵越小，该指标提供的信息量越大，在综合评价中所起作用理当越大，权重就应该越高。运用层次分析法确定各层次评价指标的权重对专家经验水平要求很高，评价结果受人为主观因素影响较大，而信息熵可以有效地弥补这一不足。信息熵是用来度量随机变量不确定程度，可以用来解决信息量的度量问题，即对不确定性的了解所需的信息量，可以被用来消除不确定性的多少来表示。如果某评价指标的熵越小，说明该指标提供的信息量就越大，在综合评价中所起的作用就越大，权重就越高[1]。应用熵权法[7?13]可以尽可能消除人为因素对应用层次分析时计算各指标权重的影响，使评价结果更为实际。

## 2.7 Java语言与Java EE

Java是一种广泛使用的计算机编程语言，拥有跨平台、面向对象、泛型编程的特性，广泛应用于企业级Web应用开发和移动应用开发。

任职于太阳微系统的詹姆斯·高斯林等人于1990年代初开发Java语言的雏形，最初被命名为Oak，目标设置在家用电器等小型系统的程序语言，应用在电视机、电话、闹钟、烤面包机等家用电器的控制和通信。由于这些智能化家电的市场需求没有预期的高，Sun公司放弃了该项计划。随着1990年代互联网的发展，Sun公司看见Oak在互联网上应用的前景，于是改造了Oak，于1995年5月以Java的名称正式发布。Java伴随着互联网的迅猛发展而发展，逐渐成为重要的网络编程语言。

Java编程语言的风格十分接近C++语言。继承了C++语言面向对象技术的核心，Java舍弃了C++语言中容易引起错误的指针，改以引用取代，同时移除原C++与原来运算符重载，也移除多重继承特性，改用接口取代，增加垃圾回收器功能。在Java SE 1.5版本中引入了泛型编程、类型安全的枚举、不定长参数和自动装/拆箱特性。

Java不同于一般的编译语言或直译语言。它首先将源代码编译成字节码，然后依赖各种不同平台上的虚拟机来解释执行字节码，从而实现了“一次编写，到处运行”的跨平台特性。

与传统类型不同，Sun公司在推出Java时就将其作为开放的技术。全球数以万计的Java开发公司被要求所设计的Java软件必须相互兼容。后来Sun公司被甲骨文公司并购，Java也随之成为甲骨文公司的产品。现时，移动操作系统 Android 大部分的代码采用Java 编程语言编程。

Java EE，Java平台企业版（Java Platform Enterprise Edition），是Sun公司为企业级应用推出的标准平台。Sun公司在1998年发表JDK1.2版本的时候，使用了新名称Java 2 Platform，即“Java2平台”，修改后的JDK称为Java 2 Platform Software Developing Kit，即J2SDK。并分为标准版（Standard Edition，J2SE），企业版（Enterprise Edition，J2EE），微型版（MicroEdition，J2ME）。J2EE便由此诞生。

2005年6月，JavaOne大会召开，SUN公司公开Java SE 6。此时，Java的各种版本已经更名以取消其中的数字“2”：J2EE更名为Java EE, J2SE更名为Java SE，J2ME更名为Java ME。

## 2.8 B/S模式

浏览器-服务器（Browser/Server）结构，简称B/S结构，与C/S结构不同，其客户端不需要安装专门的软件，只需要浏览器即可，浏览器通过Web服务器与数据库进行交互，可以方便的在不同平台下工作；服务器端可采用高性能计算机，并安装Oracle、Sybase、Informix等大型数据库。B/S结构简化了客户端的工作，它是随着Internet技术兴起而产生的，对C/S技术的改进，但该结构下服务器端的工作较重，对服务器的性能要求更高。

B/S是现今最为流行和使用的应用模式。它以Web为中心，采用TCP/IP、HTTP传输协议，客户端通过浏览器（Browser）访问Web服务器（Web Server）以及与Web相连的后台数据库。B/S模式由浏览器、应用服务器、数据库服务器三层结构组成，作为传统的C/S模式的继任者，B/S模式有以下特点[5]：

(1) 系统开发、维护、升级方便。

(2) B/S模式具有很强的开放性。

(3) B/S模式的结构易于扩展。

(4) 用户使用方便。

## 2.9 MySQL数据库

MySQL原本是一个开放源代码的关系数据库管理系统，原开发者为瑞典的MySQL AB公司，该公司于2008年被昇阳微系统（Sun Microsystems）收购。2009年，甲骨文公司（Oracle）收购昇阳微系统公司，MySQL成为Oracle旗下产品。

MySQL在过去由于性能高、成本低、可靠性好，已经成为最流行的开源数据库，因此被广泛地应用在Internet上的中小型网站中。随着MySQL的不断成熟，它也逐渐用于更多大规模网站和应用，比如维基百科、Google和Facebook等网站。非常流行的开源软件组合LAMP中的“M”指的就是MySQL。

MySQL具有以下一些特性：

1．支持 AIX、FreeBSD、HP-UX、Linux、Mac OS、NovellNetware、OpenBSD、OS/2 Wrap、Solaris、Windows等多种操作系统。

2．为多种编程语言提供了 API。这些编程语言包括 C、C++、Python、Java、Perl、PHP、Eiffel、Ruby,.NET和 Tcl 等。

3．支持多线程，充分利用 CPU 资源。

4．优化的 SQL查询算法，有效地提高查询速度。

5．提供 TCP/IP、ODBC 和 JDBC等多种数据库连接途径。

6．提供用于管理、检查、优化数据库操作的管理工具。

7．支持大型的数据库。可以处理拥有上千万条记录的大型数据库。

8．支持多种存储引擎。

9．.MySQL 使用标准的 SQL数据语言形式。

10．原生JSON支持

## 2.10 IntelliJ IDEA介绍

IDEA 全称 IntelliJ IDEA，是Java语言开发的集成环境，IntelliJ在业界被公认为最好的Java开发工具之一，它集成了开发过程中实用的众多功能，尤其在智能代码助手、代码自动提示、重构、J2EE支持、各类版本工具(git、svn、github等)、JUnit、CVS整合、代码分析、 创新的GUI设计等方面的功能可以说是超常的。IDEA是JetBrains公司的产品，它的旗舰版本还支持HTML，CSS，PHP，MySQL，Python等。免费版只支持Java等少数语言。

这款IDE最突出的功能是调试（Debug)，它可以对Java代码，JavaScript，JQuery，Ajax等技术进行调试。最后，在多线程调试的情况下，Log on console的功能可以帮你检查多线程执行的情况。

IDEA所提倡的是智能编码，是减少程序员的工作，IDEA的特色功能有以下25点：

智能的选取

在很多时候我们要选取某个方法，或某个循环或想一步一步从一个变量到整个类慢慢扩充着选取，IDEA就提供这种基于语法的选择，在默认设置中Ctrl+W，可以实现选取范围的不断扩充，这种方式在重构的时候尤其显得方便。

丰富的导航模式

IDEA提供了丰富的导航查看模式，例如Ctrl+E显示最近打开过的文件，Ctrl+N显示你希望显示的类名查找框（该框同样有智能补充功能，当你输入字母后IDEA将显示所有候选类名）。在最基本的project视图中，你还可以选择多种的视图方式。

历史记录功能

不用通过[版本管理](http://baike.baidu.com/item/%E7%89%88%E6%9C%AC%E7%AE%A1%E7%90%86" \t "http://baike.baidu.com/_blank)服务器，单纯的IDEA就可以查看任何工程中文件的历史记录，在版本恢复时你可以很容易的将其恢复。

[JUnit](http://baike.baidu.com/item/JUnit" \t "http://baike.baidu.com/_blank)的完美支持

对重构的优越支持

IDEA是所有IDE中最早支持重构的，其优秀的重构能力一直是其主要卖点之一。

编码辅助

Java规范中提倡的toString()、hashCode()、equals()以及所有的get/set方法，你可以不用进行任何的输入就可以实现代码的自动生成，从而把你从无聊的基本方法编码中解放出来。

灵活的排版功能

基本所有的IDE都有重[排版](http://baike.baidu.com/item/%E6%8E%92%E7%89%88" \t "http://baike.baidu.com/_blank)功能，但仅有IDEA的是人性的，因为它支持排版模式的定制，你可以根据不同的项目要求采用不同的排版方式。

[XML](http://baike.baidu.com/item/XML" \t "http://baike.baidu.com/_blank)的完美支持

xml全提示支持：所有流行框架的xml文件都支持全提示，谁用谁知道。

动态语法检测

任何不符合java规范、自己预定义的规范、累赘都将在页面中加亮显示。

代码检查

对代码进行自动分析，检测不符合规范的，存在风险的代码，并加亮显示。

对JSP的完全支持

不需要任何的[插件](http://baike.baidu.com/item/%E6%8F%92%E4%BB%B6" \t "http://baike.baidu.com/_blank)，完全支持JSP。

智能编辑

代码输入过程中，自动补充方法或类。

EJB支持

不需要任何插件完全支持[EJB](http://baike.baidu.com/item/EJB" \t "http://baike.baidu.com/_blank)(6.0 支持EJB3.0)

列编辑模式

用过UtralEdit的肯定对其的列编辑模式赞赏不已，因为它减少了很多无聊的重复工作，而IDEA完全支持该模式，从而更加提高了编码效率。

预置模板

预置模板可以让你把经常用到的方法编辑进模板，使用时你只用输入简单的几个字母就可以完成全部代码的编写。例如使用比较高的public static void main(String[] args){}你可以在模板中预设pm为该方法，输入时你只要输入pm再按代码辅助键，IDEA将完成代码的自动输入。

完美的自动代码完成

智能检查类中的方法，当发现方法名只有一个时自动完成代码输入，从而减少剩下代码的编写工作。

版本控制完美支持

集成了目前市面上常见的所有版本控制工具插件，包括git、svn、github，让开发人员在编程的工程中直接在intellij idea里就能完成代码的提交、检出、解决冲突、查看版本控制服务器内容等等。

不使用代码的检查

自动检查代码中不使用的代码，并给出提示，从而使代码更高效。

智能代码

自动检查代码，发现与预置规范有出入的代码给出提示，若程序员同意修改自动完成修改。例如代码：String str = "Hello Intellij " + "IDEA"; IDEA将给出优化提示，若程序员同意修改IDEA将自动将代码修改为：String str = "Hello Intellij IDEA";

正则表达式的查找和替换功能

查找和替换支持正则表达式，从而提高效率。

[JavaDoc](http://baike.baidu.com/item/JavaDoc" \t "http://baike.baidu.com/_blank)预览支持

支持JavaDoc的预览功能，在JavaDoc代码中Ctrl+Q显示JavaDoc的结果，从而提高doc文档的质量。

程序员意图支持

程序员编码时IDEA时时检测你的意图，或提供建议，或直接帮你完成代码。

1. **数据质量本体构建**

## 3.1 本体的概念、关系以及约束公理

3.1.1本体的概念

3.1.1本体之间的关系

经过研究和实例证明，在本体中有四种基本的关系，分别是part-of，kind-of，instance-of, instance-of,其中：

（1）part-of：部分和整体的关系，表示一个类是另一个的子类，但是该类也是一个独立的类，例如树叶是树的一部分，但是树叶也是单独的一个类；

（2）kind-of：继承关系，表示一个类是另一个类的子类，对应于OWL语言的owl：SubClassOf,拥有这个关系的子类将继承父类的所有属性及约束；

（3）instance-of：实例关系，表示一个实例是属于某个类的，例如学生类中有一个实例是张三这个学生，那么张三和学生类之间的关系就是实例关系；

（4）attribute-of：属性关系，表示的是两个类之间的关系，对应于OWL语言中的owl：Property，属性关系表现了概念之间的关系，对应的OWL语言表示有：owl：ObjectProperty和owl：DatatypeProperty；

3.1.2本体的约束公理

约束(Restriction)公理是OWL类公理的核心部分，用来详细描述概念所存在的各种约束条件。每个约束公理都是一个匿名类，是被约束类的一个父类。每个约束公理只表示一个约束条件，是对某个特定属性在特定类之上的约束，也就是说，这个约束只在包含这个属性的类定义中有效，如果没有明确声明，其他包含这个属性的类定义则不受此约束。这样定义的本体满足了最小的本体承诺(Minimal Ontological Commitment)规则，即本体的承诺应该最小，只需要定义约束最弱的公理，满足特定的需求即可。约束公理可以分为值约束(value constraint)和基数约束(cardinality constraint)两种，值约束限制属性的值域(range)，基数约束限制属性取值的个数。

1.值约束公理

（1）allValuesFrom：相当于全称量词。被约束属性的所有取值都必须是由allValuesFrom所指定的类的实例，或者是指定值域的数值，取值可以为空，应用用实例属性断言的取值中；

（2）someValuesFrom：相当于存在量词。被约束属性的所有取值中至少有一个是someValuesFrom所指定的类的实例，或者是指定值域的数值。取值可以不能为空，应用用实例属性断言的取值中；

（3）allValuesFrom和someValuesFrom称为量词约束，可以用一个三元组表示：<quantifier,property,filler>，即一个量词、一个属性以及和拥有这个属性的一个类；

（4）hasValue：被约束属性的所有取值中至少有一个是hasValue所指定的值或者与指定的值语义上相当。取值不能为空，用该公理来表示非空的约束。

2.基数约束公理

（1）maxCardinality：被约束属性的值(个体或数值)最多能取maxCardinality所指定的数目的不同值，该约束用来控制实例的属性取值个数，不能超过maxCardinality所指定的数目；

（2）)minCardinality：被约束属性的值(个体或数值)最少应取minCardinality所指定的数目的不同值，该属性用来控制实例属性取值个数，当minCardinality取值为1时，表示该属性取值至少有一个，可用该约束来控制非空约束；

（3）cardinality：指定被约束属性的取值基数，可以用一组取值一致的maxCardinality和minCardinality来替代。

OWL 约束公理的应用主要在以下3个方面：

(1)定义类的必要条件。仅有必要条件约束的类称为Primitive类，也叫Partial类。只要一个个体是某个Primitive类的实例，就可以判断这个个体一定满足这些条件；但如果一个个体都满足这些条件，也不能就此判断这个个体是这个Primitive类的实例。

(2)定义类的充要条件。有充要条件约束的类称为Defined类，也叫Complete类。只要一个个体是某个Defined类的实例，就可以判断这个个体一定满足这些条件，反过来，如果一个个体都满足这些条件，也可以判断这个个体是这个Defined类的实例。

(3)定义类的封闭公理(Closure Axioms)。封闭公理用约束的形式说明某个类上某个属性的取值范围，不同于论域和值域，前者是局部约束，后者是全局约束。封闭公理有两种:

1)用allValuesFrom封闭someValuesFrom；

2)用cardinality =n封闭allValuesFrom。

## 3.2 本体的构建准则

具体工程和领域的不同，所要考虑的重点也不同，本体构建的过程方法也各不相同。目前还没有一个统一的本体构建方法标准，1995 年 Gruber 提出的五条准则[22]在本体构建领域内得到了比较广泛的认可。

1) 清晰性(Clarity)、客观性(Objectivity)与明确性(Clarity)：本体应该能够有效的说明所属的内涵，即能够通过自然的语言对需要定义的术语进行客观的、明确的定义，即能够有效地表达出所需定义的术语意思。并且定义还应该可用形式化的逻辑公理表达出。

2) 完全性(Completenese)：尽可能完整的定义本体的概念，可以完备的清晰表达出所需描述的术语内含。

3) 一致性(Coherence)：本体应该具有前后的一致性，即通过术语得到的推论和术语原本的含义间是相容的，即在与其定义一致的推理时不会产生矛盾；使用自然语言进行说明的文档与本体公理也要保持一致。

4) 最大单调可扩展性(Extendibility)：本体应该可为后期可预见的任务提供概念基础，能够在现有的概念基础之上增添新的术语，即无需修改现有的概念定义就可向本体中再添加新的专用或通用术语。

5) 最 小 承 诺 (Minimal ontology commitment) 和 最 小 编 码 偏 好(Minimal encoding bias)：在本体构建初期时本体约定应该最小，仅仅需满足特定的共享需求，之后的使用者可以按照自己的需要对本体进行专业化和实例化的约束。本体的承诺是指如何通过相容的、一致的方式来使用领域中的术语所达成的共识。

后来的一些本体研究人员又归纳了许多面向具体操作的补充性规则，Arpirez[22]提出了以下三条比较著名的规则：

1) 概念命名标准化

2) 概念层次多样化

3) 语义距离最小化

概念命名标准化可以以增强本体的适应性，便于本体的共享。概念层次多样化可以有效支持本体的多继承机制能力；语义距离最小化是指将内含相近的概念提取出来，使用同一元语进行表示。

以上的几个规则较为抽象，仅对本体的构建起到大致指导性作用。而实际的本体构建过程，可在大致参考以上几个原则的基础之上再依据本项目的实际情况进一步深化。虽然当前还没有统一的本体构建方法与方法性能评估的标准，但是构建领域本体时需要领域专家的参与是本体领域公认的。

## 3.3 数据质量本体构建流程

数据质量与本体发展到现在，已经有很多种的构建方法了。不过，目前仍然不能实现本体构建的完全自动化，各个本体的研发机构也都采用自己的本体构建机制与方法。下面介绍几个在学术界中比较被普遍接受的本体构建方法。

1. IDEF-5方法

本文中采用的就是IDEF-5方法，它是在结构化分析方法的基础上发展起来的。它的主要思想是通过使用图表语言和细化说明语言，获取关于客观存在的概念、属性和关系，最终形式化为本体。它的创建步骤如下图所示：

1. Skeletal Methodolody骨架法

定义课题、组织队伍

收集数据

本体优化与验证

本体初步开发

分析数据

骨架法是专门用来创建企业本体的方法。它常用来充当商业企业间术语和定义的集合。它的创建步骤如下图所示：

1. TOVE企业建模法

不符合

确定本体的应用目的和范围

本体分析

本体表示

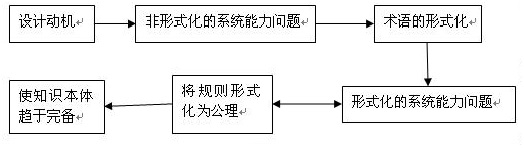
评价

本体评价

建立本体

符合

TOVE法是加拿大Toronto大学企业集成实验室开发的应用于商业过程和活动建模领域的本体开发模式。



1. Methontology方法

该方法是由GOMEZ-PEREZ法和骨架法结合而产生的一种本体构建方法，这是一种更加具有普遍性的构建方法。它将本体开发进程和本体生命周期分开为两个方面，用不同的两个技术予以支持，形成类软件工程的开发方法。它的创建步骤如下：

1. 管理阶段：用于规划任务进度，确立所需资源，以及如何保证质量等问题
2. 开发阶段：分为五个步骤，分别为规范说明、概念化、形式化、执行以及维护
3. 维护阶段：也分为五个步骤，包括知识获取、系统集成、评价、文档说明和配置管理
4. 七步法

七步法是由斯坦福大学医学院开发的本体构建方法。该方法由七个步骤组成：确定本体的专业领域和范畴；是否可以复用现有本体；列出本体中的重要术语；定义类和类的等级关系；定义类的属性；定义属性的分面；创建实例[3]。

当然了，数据质量本体的构建方法并不局限于此。不管选用哪种本体构建方法，可扩展性、术语清晰、推理一致、最小编码、本体约定最小都是领域专家在构建本体时应遵循的基本规则。具体情况下，也要根据自己所在的领域，追求的目的等各方面因素，选择或者开发最有效的本体构建方法。不过，考虑到成本的问题，在有时虽不是最有效但不存在较大问题的情况下，可以复用现有本体。只是一种极大节约成本且行之有效的方法。

## 3.4 数据质量本体类以及属性

通过protege创建数据质量本体类，部分属性如下：

表 本体类

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 对象属性 | 数据类型 | 定义域 | 值域 |
| HasMorethan | 是 | String | DATACOLUMN | DATACOLUMN |
| HasLessthan | 是 | String | DATACOLUMN | DATACOLUMN |
| HasExist | 是 | String | DATACOLUMN | DATACOLUMN |
| Equals | 是 | String | AssessedColumn | AssessedColumn |
| GreaterThan | 是 | String | AssessedColumn | AssessedColumn |
| NotEquals | 是 | String | AssessedColumn | AssessedColumn |
| LessThan | 是 | String | AssessedColumn | AssessedColumn |
| NOTNULL | 否 | Boolean | DATACOLUMN | Boolean |
| TABLE\_ID | 否 | String | DATATABLE | String |
| COLUMN\_ID | 否 | String | DATACOLUMN | String |
| COLUMN\_NAME | 否 | String | DATACOLUMN | String |
| DATA\_TYPE | 否 | String | DATACOLUMN | String |

## 3.5 数据质量规则对在本体中的描述

数据质量本体是对数据质量的评估规则构建一个本体，数据质量的具体评估方法还没有系统的评估框架。所以本文在进行数据质量本体构建时，将抽取数据质量评估规则中具体代表性的数据质量评估规则，具体的数据质量评估规则如下所示：

1. 非空约束：非空约束是完整性约束的一个部分，非空约束要求数据表中某个字段的取值不能为空。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 数据属性 | 定义域 | 值域 |
| 1 | NOTNULL | DataColumn | Boolean |

1. 值域约束：值域约束要求数据表中某个字段的取值必须在一定的范围内。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 数据属性 | 定义域 | 值域 |
| 1 | MIX\_VALUE | DataColumn | Float |
| 2 | MAX\_VALUE | DataColumn | Float |

1. 等值函数依赖约束：等值函数依赖规则表示数据表中的数据，在同一记录上，该数据项上的数据必须（或在满足某一条件时必须）由另一个或多个数据项上的数据通过函数计算得出。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 对象属性 | 定义域 | 值域 |
| 1 | Equals | AssessedColumn | AssessedColumn |

1. 逻辑依赖约束：逻辑依赖约束表示数据表中的数据，在同一记录上，该数据项上的数据必须（或在满足某一条件时必须）与另一个或多个数据项上的数据满足某种逻辑关系（如大于、小于等）。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 对象属性 | 定义域 | 值域 |
| 1 | LessThan | AssessedColumn | AssessedColumn |
| 2 | GreaterThan | AssessedColumn | AssessedColumn |
| 3 | NotEquals | AssessedColumn | AssessedColumn |

1. 等值一致性依赖约束：等值一致性依赖约束表示数据表中的数据，数据项上的数据必须（或在满足某一条件时必须）由另一个或多个数据表中的一个或多个数据项上的数据计算得出。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 对象属性 | 定义域 | 值域 |
| 1 | ReferenceEquals | AssessedColumn | ReferenceColumn |

1. 逻辑一致性依赖约束：逻辑一致性依赖约束表示数据表中的数据，该数据项上的数据必须（或在满足某一条件时必须）与另一个或多个数据表中的一个数据项上的数据满足某种逻辑关系（如大于、小于等）。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 对象属性 | 定义域 | 值域 |
| 1 | ReferenceGreaterThan | AssessedColumn | ReferenceColumn |
| 2 | ReferenceLessThan | AssessedColumn | ReferenceColumn |

1. 存在一致性依赖约束：存在一致性依赖约束表示数据表中的数据项必须（或在满足某一条件时必须）在另一个或多个数据表中的一个数据项上的数据中出现。

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 对象属性 | 定义域 | 值域 |
| 1 | HasExist | DataColumn | DataColumn |

1. **基于Jess的数据质量评价方法**

## 4.1 基于SWRL的Jess推理

4.1.1 SWRL概述

由于本体本身并不支持没有用于推理的规则，而数据质量评价需要加入支持更广泛的推理的规则，所以我们引入SWRL。它是以语义的方式呈现规则的一种语言。SWRL 规则呈现为前提和结论之间的蕴涵式，主要作用于OWL的实例及其联系，一般形式为：[7]。

4.2.2 Jess与Protege的集成

在Protege上安装JessTab插件后就完成了集成。它可以完成下面四个任务:①将SWRL的规则转换为Jess的规则或直接在Jess中基于本体概念创建规则；②将有关OWL个体的知识表示为Jess的事实；③基于规则和事实运行Jess推理引擎进行推理；④用推理后得到的新事实更新OWL知识库。工作过程如下图所示[7]。

## 4.2 熵权法在本体中的设计实现

类、属性

属

性

值

SWRL规则

个体

(实例)

本

体

知识库

规则

事实

Jess

推

理

引

擎

新事实

(1)

(2)

(3)

(4)

4.2.1熵权法用于数据质量评价的基本原理

熵权法是在已知对象集合各评定指标值的基础上，计算出在竞争意义上的激烈程度[6]。熵是对混乱程度的度量。在数据质量评价中，随着信息量的增加，不确定的因素就会越少，即熵就会越小；反之熵就会越大。 熵值为1即不存在有价值的信息，其熵权则对应为0，这就表示取消该指标。熵权的取值范围是[0,1]，熵权越大表示该指标越为重要，对最终结果的影响也越大。我们用表示第个指标的熵权，表示指标总数，则。

4.2.2熵权法的具体设计步骤

1. 首先要确立的，就是熵权法的评价对象，即为指标水平矩阵。矩阵。其中表示第个评定对象的第个指标的数值即对应位置的重要性，表示横坐标的长度即评价规则的个数，表示纵坐标的长度即指标字段（本体待评价实例）的个数。
2. 异质指标同质化处理。进行同质化是因为存在不同指标之间计量单位不同的情况。在我们进行综合指标计算之前，要先进行解决标准化处理的问题。由于正向指标与负向指标代表的含义不同，所以将正负指标分开处理。公式如下：

正向指标：

负向指标：

1. 计算第项指标的熵值。公式如下：



其中，，，

1. 计算评价指标的熵权。公式如下：



其中

4.2.3计算综合指标权重

熵权法极大地利用了原始数据中所存在的信息，客观的反映了结果。但是它的缺陷别是不能结合专家的经验与知识，以及决策者的侧重点，造成结构不够切合实际需求。而层次分析法恰好与之相反，是以专家的知识和经验、决策者的侧重点为核心，导致主观随意性过高的问题。综合来看，将两者结合起来使用就能达到理想的效果。

（）

4.2.4确定综合结果

通过加权由综合权重计算出综合结果



## 4.3 数据质量的jess推理实现

1. 完整性  
   完整性（Integrity）用来描述数值信息的完整程度。完整性包括完整性约束规则、非空约束规则、连续性约束规则三个维度的规则。
2. **系统实现**

5.1 数据质量本体

5.2 基于熵权法的数据质量评价实现

5.3 评估实现

1. **系统测试**

6.1测试基本原则

6.2软件测试步骤

6.3 测试用例

**参考文献**

[3] Uschold M,Gruninger M.Ontologies:Principles,Methods and Applications[J]. Knowledge Engineering Review,1996,11(2):93-155.

[6]李汶静．基于熵权ＡＨＰ法的企业价值组合评估应用研究［D］．成都：西南石油大学，2010．

[7]纪兆辉、李存华.基于SWRL和Jess构造语义Web规则及其对策分析.淮海工学院学报 第18卷第4期