软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(1):142−163 [doi: 10.13328/j.cnki.jos.005649]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

数据模型及其发展历程*

信俊昌[1], 王国仁[2], 李国徽[3], 高云君[4], 张志强[5]

[1](东北大学 计算机科学与工程学院,辽宁 沈阳 110819)

[2](北京理工大学 计算机学院,北京 100081)

[3](华中科技大学 计算机科学与技术学院,湖北 武汉 430074)

[4](浙江大学 计算机科学与技术学院,浙江 杭州 310058)

[5](浙江财经大学 信息管理与工程学院,浙江 杭州 310018)

通讯作者: 王国仁, E-mail: wanggrbit@126.com

摘 要: 数据库是数据管理的技术,是计算机学科的重要分支.经过近半个世纪的发展,数据库技术形成了坚实的理论基础、成熟的商业产品和广泛的应用领域.数据模型描述了数据库中数据的存储方式和操作方式.从数据组织形式,可以将数据模型分为结构化模型、半结构化模型、OLAP 分析模型和大数据模型.20 世纪 60 年代中后期到 90 年代初,结构化模型最早被提出,其主要包括层次模型、网状模型、关系模型和面向对象模型等.20 世纪 90 年代末期,随着互联网应用和科学计算等复杂应用的快速发展,开始出现半结构化模型,包括 XML 模型、JSON 模型和图模型等.21 世纪,随着电子商务、商业智能等应用的不断发展,数据分析模型成为研究热点,主要包括关系型 ROLAP 和多维型 MOLAP.2010 年以来,随着大数据工业应用的快速发展,以 NoSQL 和 NewSQL 数据库系统为代表的大数据模型成为新的研究热点.对上述数据模型进行了综述,并选取每个模型的典型数据库系统进行了性能的分析.

关键词: 数据模型;结构化模型;半结构化模型;OLAP 分析模型;大数据模型

中图法分类号: TP311

中文引用格式: 信俊昌,王国仁,李国徽,高云君,张志强.数据模型及其发展历程.软件学报,2019,30(1):142−163. http://www.jos.org.cn/1000-9825/5649.htm

英文引用格式: Xin JC, Wang GR, Li GH, Gao YJ, Zhang ZQ. State of the art data model and its research progress. Ruan Jian Xue Bao/Journal of Software, 2019,30(1):142−163 (in Chinese). http://www.jos.org.cn/1000-9825/5649.htm

State of the Art Data Model and Its Research Progress

XIN Jun-Chang[1], WANG Guo-Ren[2], LI Guo-Hui[3], GAO Yun-Jun[4], ZHANG Zhi-Qiang[5]

[1](School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China)

[2](School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China)

[3](School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

[4](College of Computer Science and Technology, Zhejiang University, Hangzhou 310058, China)

[5](School of Information Management and Engineering, Zhejiang University of Finance and Economics, Hangzhou 310018, China)

* 基金项目: 国家自然科学基金(61472069, 61732003, U1401256, 61729201, 61572215, 61522208, 61672181, 61202090, 61272184); 黑龙江省科学基金(LC2017029, F2016005); 哈尔滨市青年科技创新人才研究专项基金(2016RAXXJ036, 2015RQQXJ067)

Foundation item: National Natural Science Foundation of China (61472069, 61732003, U1401256, 61572215, 61522208, 61672181, 61202090, 61272184); Natural Science Foundation of Heilongjiang Province, China (LC2017029, F2016005); Harbin Special Fund for Young Scientists and Technicians (2016RAXXJ036, 2015RQQXJ067)

本文由“软件学科发展回顾特刊”特约编辑梅宏教授、金芝教授、郝丹副教授推荐.

收稿时间: 2018-07-02; 修改时间: 2018-08-15; 采用时间: 2018-09-27; jos 在线出版时间: 2018-11-20

CNKI 网络优先出版: 2018-11-21 09:52:33, http://kns.cnki.net/kcms/detail/11.2560.TP.20181121.0952.003.html

Abstract: Database management technology is an important branch of computer science. After the development of nearly half a century, database technology has formed a solid theoretical foundation, mature commercial products, and a wide range of applications. The data model describes the storage and operation of data in the database. According to the organizational form of data, there are four types of data models: structured models, semi-structured models, OLAP analysis models, and big data models. From the late 1960s to the early 1990s, the structured models were first proposed, which mainly includes hierarchical model, network model, relational model, and object-oriented model. In the late 1990s, with the rapid development of complex applications such as Internet applications and scientific computing, semi-structured models began to emerge, including XML models, JSON models, and graph models. In the new century, with the continuous development of applications such as e-commerce and business intelligence, the data analysis model has become a research hotspot, mainly including relational ROLAP and multi-dimensional MOLAP. Since 2010, with the rapid development of big data industry applications, the big data model represented by NoSQL and NewSQL database systems has become a new research hotspot. This article summarizes the above data models, and analyzes the performance of typical database system selected from each model.

Key words: data model; structured model; semi-structured model; OLAP analysis model; big data model

在信息化社会,数据库技术是管理信息系统、办公自动化系统、决策支持系统等各类信息系统的核心部分,是进行科学研究和决策管理的重要技术手段.数据库技术从诞生到现在,在不到半个世纪的时间里,形成了坚实的理论基础、成熟的商业产品和广泛的应用领域,吸引了越来越多的研究者加入.数据库的诞生和发展,给计算机信息管理带来了一场巨大的革命[1].几十年来,国内外已经开发建设了成千上万个数据库,它已成为企业、部门乃至个人日常工作、生产和生活的基础设施.同时,随着应用的扩展与深入,数据库的数量和规模越来越大,数据库的研究领域也已经大大地拓广和深化了.自 1966 年计算机图灵奖设立以来,数据库领域共获得了 4 次该奖项(1973 年 C.W. Bachman1,1983 年 E.F. Codd,1998 年 J. Gray 和 2014 年 M. Stonebraker),更加充分地说明了数据库是一个充满活力和创新精神的领域.

数据模型是数据库中数据的存储方式和操作方式,是数据库系统的基础.现实世界中客观存在的事物之间存在着多种联系,数据模型是将不同的联系通过筛选、归纳、总结、命名等抽象过程产生出概念模型,用以表示对现实世界的描述,然后转换成真实、容易被人们理解和便于计算机处理的数据表现形式.也可以说,数据模型用于表达现实世界中的对象,也就是将现实世界中杂乱的信息,用一种规范而形象化的方式表达出来.

根据不同模型的应用层次,可以将数据模型分为概念数据模型、逻辑数据模型和物理数据模型:概念数据模型中最常用的有E-R 模型和面向对象模型等,主要用来描述世界的概念化结构,它使数据库的设计人员在设计的初始阶段,集中精力分析数据以及数据之间的联系;逻辑数据模型反映的是系统分析设计人员对数据存储的观点,是对概念数据模型进一步的分解和细化,其中最常用的是层次模型、网状模型、关系模型和大数据模型等;物理数据模型描述了数据在储存介质上的组织结构,是在逻辑数据模型的基础上,考虑各种具体的技术实现因素,进行数据库体系结构设计,真正实现数据在数据库中的存放.根据不同模型的语义关系,可以将数据模型分为 XML、RDF 模型和超模型等:XML 模型是一种分层自描述模型;RDF 是一种基于 XML(可扩展标记语言)编写的元数据(描述数据的数据),用于描述 Web 资源的标记语言;超模型是一组超实体以及定义在它们上面的关系和约束组成,为复杂的实体模型建模提供了快捷的方法.根据不同模型的应用场景,可以将数据模型分为离线模型、在线模型和近线模型:离线模型的主要代表为 OLAP 模型;在线模型可以可靠地处理无限的数据流,像 Hadoop 批量处理大数据一样,实时处理数据,主要代表为 Storm 等;近线模型定位于在线存储和离线存储之间,是指将那些并不是经常用到或者说数据的访问量并不大的数据加以存储,但要求对这些数据寻址要迅速、传输率要高.目前,近线模型很多是基于 Hadoop 分布式文件系统构建起来的.根据数据模型的发展历程,按时间段将数据模型分为结构化模型、半结构化模型、OLAP 分析模型和大数据模型,其发展过程如图 1 所示.20 世纪 60 年代中后期,出现了结构化模型,主要包括层次模型、网状模型、关系模型和面向对象模型等.20 世纪 80 年代以前主要研究关系模型,关系模型为数据库系统和产业的发展奠定了坚实的基础.20 世纪 70 年代后期产生了 ER 模型,80 年代中期开始出现面向对象模型,到 20 世纪 90 年代初期,面向对象模型达到一个顶峰.20 世纪 90 代末期,随着互联网应用和科学计算等复杂应用的快速发展,开始出现半结构化模型,包括 XML 模型、JSON 模型、RDF 模型、图模型和超模型等.XML 模型是一种分层自描述模型;JSON 使用文本表示一个 JS 对象的信

息,支持字符串、数字、对象、数组等各种类型;图模型应用图论存储实体间关系.进入 21 世纪,随着电子商务、商业智能等应用的不断发展,数据分析模型成为研究热点,主要包括 ROLAP 模型、MOLAP 模型和 Storm 模型等.ROLAP 模型主要研究事实表和维表的组织表示及其变种,包括星型模型、雪花模型等;MOLAP 模型主要研究数据立方及其优化计算算法.2010 年以来,随着大数据工业应用的快速发展,以 NoSQL 和 NewSQL 数据库系统为代表的大数据模型成为新的研究热点,主要包括 key-value 模型、key-column 模型和 key-document 模型等.本文对上述数据模型进行了综述,并选取每个模型的典型数据库系统进行了性能的分析.

因此,根据数据模型的发展历程,本文第 1 节和第 2 节分别综述结构化模型和半结构化模型.第 3 节综述 OLAP 分析模型.第 4 节综述大数据模型.第 5 节选取每个模型的典型数据库系统进行性能的分析.第 6 节进行总结.

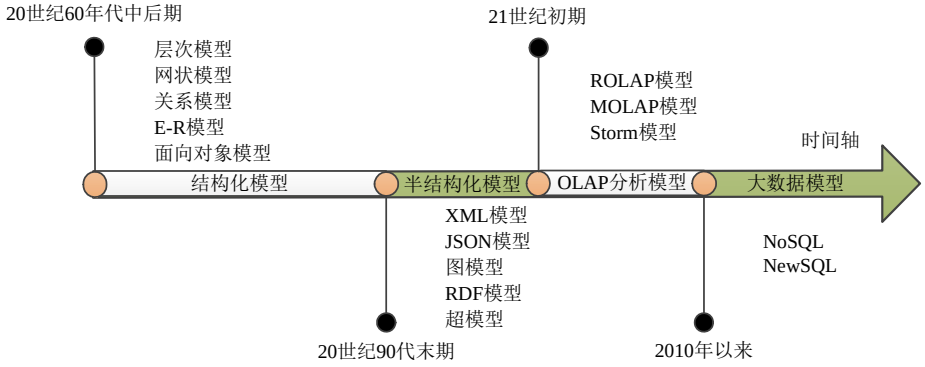


Fig.1 Timeline for the development of data models

图 1 数据模型的发展时间轴

1 结构化模型

结构化模型是最早被提出来的数据模型,是数据模型中最基础的模型之一.结构化模型主要包括层次模型、网状模型、关系模型和面向对象模型等.20 世纪 60 年代中后期,出现了以 IBM 公司的 IMS 系统[2]为代表的层次模型、DBTG[3]提出的网状模型和 Codd[4,5]提出的关系模型.关系型数据库是目前最为流行的数据库,同时也是被普遍使用的数据库.20 世纪 70 年代后期产生了 E-R 模型,通过 E-R 图,计算机专业人员与非计算机人员可以进行交流合作,以真实、合理地模拟一个单位,作为进一步设计数据库的基础.20 世纪 80 年代以前主要研究关系模型,关系模型为数据库系统和产业的发展奠定了坚实基础.20 世纪 80 年代中期开始出现面向对象模型,到 20 世纪 90 年代初期,面向对象模型达到一个顶峰.面向对象数据模型是一种可扩充的数据模型,它吸收了语义数据模型和知识表示模型的一些基本概念,同时又借鉴了面向对象程序设计和抽象数据类型的一些思想.面向对象数据模型不是一开始就有明确的定义,而是在发展中逐步形成的.直到 1991 年,美国国家标准协会(ANSI)的一个面向对象数据库工作组才提出第一个标准化的报告.

结构化模型可以分为传统数据模型和非传统数据模型,如图 2 所示.传统数据模型主要包括层次模型、网状模型和关系模型;非传统数据模型主要包括 E-R 模型和面向对象模型等.

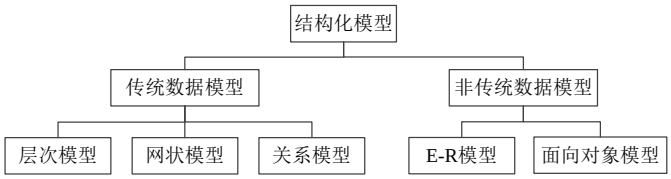


Fig.2 Classification of structured models

图 2 结构化模型类别

1.1 层次模型

层次模型采用树形结构来表示各类实体以及实体间的联系,具有数据结构比较简单清晰、数据库的查询效率高等优点.然而,现实世界中很多联系是非层次性的,而层次数据库系统只能处理一对多的实体联系;其次,由于层次模型查询子女节点必须通过双亲节点,查询效率较低.

在采用层次模型的典型数据库系统中,最著名的是IBM公司于1968年研制的IMS(information management system)[2].在IMS数据库管理系统中,数据被分层处理,层与层之间的数据彼此独立.这样使得数据保持彼此的独立完整,优化了数据的存储和获取进程.由于IBM公司具有强大的竞争力并且IMS是最早的大型数据库管理系统,因此该系统在公开之后被广泛应用.

1.2 网状模型

由于层次数据库系统只能处理一对多的实体联系,对现实世界的描述比较局限,因此出现了网状模型.网状模型的主要特点是子女节点与双亲节点的联系可以不唯一.

网状模型具有更好的性能和较高的存取效率,但网状模型结构、数据定义语言和数据操作语言比较复杂,用户不容易掌握和使用;其次,由于网状模型记录之间的联系是通过存取路径实现的,应用程序在访问数据时必须选择适当的存取路径,因此用户必须了解系统结构的细节,加重了编写应用程序的负担.在采用网状模型的典型数据库系统中,最具代表性的是20世纪70年数据系统语言研究会CODASYL(Conf. on Data Systems Language)下属的数据库任务组(Data Base Task Group)提出的DBTG系统[3],亦称CODASYL系统.DBTG在数据定义语言中提供了数据库完整性的若干概念和语句.现有的网状数据库系统大都采用DBTG方案.

1.3 关系模型

由于层次模型和网状模型缺乏充实的理论基础,于是人们开始寻求具有较充实理论基础的数据模型.IBM公司的E.F.Codd从1970年~1974年发表了一系列有关关系模型的论文[4,5],从而奠定了关系数据库的设计基础.关系数据模型使用二维表表示实体和实体之间的关系,其数学定义如下.

(1) 域:具有相同的数据类型值的集合,语义上通常是指某一对象的取值范围;

(2) 笛卡尔积:设 $D_1,D_2,\ldots,D_n$ 是 n 个域,则它们的笛卡尔积为 $D_1\times D_2\times\ldots\times D_n=\{(d_1,d_2,\ldots,d_n)|d_i\in D_i,i=1,2,\ldots,n\}$,其中每一个元素称为一个 n 元组,简称元组;元组中的每个值 d_i 称为一个分量;

(3) 关系:笛卡尔积 $D_1\times D_2\times\ldots\times D_n$ 的子集合,记作 $R(D_1,D_2,\ldots,D_n)$.其中,R 称为关系名,n 为关系的目或度.

关系型数据库管理系统的特征如下.

(1) 无论是实体、还是实体之间的联系都被映射成一张二维表;

(2) 借助关系表,表示实体之间的多对多的关系;

(3) 关系中的每个属性是不可分割的实体.

图3是学生选课系统的关系模型.从图中可以看到,学生与课程之间的联系以及教师和课程之间的多对多联系都被映射成了表格.其中,选课表中的sut_id和cour_id分别是引用学生表和课程表的cour_id的外键;教课表也是如此.

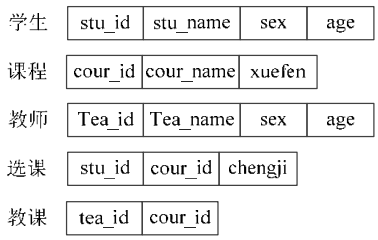


Fig.3 Relationship model instance of student

图3 学生关系模型实例

关系模型是一些表格的框架,实体的属性是表格中列的条目,实体之间的关系也是通过表格的公共属性来表示,结构简单明了;其次,存取路径对用户而言是完全隐蔽的,程序和数据具有高度的独立性;再次,关系数据模型操作方便,在模型中操作的基本对象是集合而不是某一个元组数据.但是关系数据模型查询时只需指明数据存在的表和需要的数据所在的列,不用指明具体的查找路径,因而加大了系统的负担,查询效率较低.关系型数据库是目前最流行的数据库,同时也是被普遍使用的数据库.在采用关系模型的典型数据库系统中,目前业界普遍使用的有 MySQL[6]、PostgreSQL[7]、DB2[8]、Oracle[9]以及 SQL Server[10]等系统:MySQL 数据库是由瑞典 MySQL AB 公司 1994 年开始研发的一个开源数据库系统,因其具备处理速度快、可靠性高和适应性强等特点而备受关注; PostgreSQL 是由加州大学伯克利分校计算机系开发的 POSTGRES 系统发展而来,支持处理丰富的数据类型; DB2 是由 IBM 公司开发,采用多进程、多线索体系结构,可以运行于多种操作系统之上的关系型数据库管理系统;Oracle关系数据库管理系统是由甲骨文公司研发的一种高效率、高可靠性并适应高吞吐量的数据库解决方案;SQL Server 是 Microsoft 公司推出的关系型数据库管理系统,具有使用方便、可伸缩性好与相关软件集成程度高等优点.

1.4 E-R模型

传统数据模型以记录为基础,不能很好地面向用户和应用[11].因为人们对现实世界的认识往往通过实体来实现,而现实中实体不一定与模型中的记录相对应,一个记录可能包含多个实体,一个实体可能分在多个记录中描述,有些实体也可能仅仅作为某个记录中的属性出现;其次,用户很难从传统数据模式看出实体间的全面联系,现实世界中的实体联系被淹没在关系和属性之中;再次,传统数据模型语义贫乏,支持的数据类型较少.因此从 20 世纪 70 年代后期开始,出现了第一种非传统的数据模型——E-R 模型.E-R 模型不同于传统数据模型面向数据库的实现,而是面向现实世界.E-R 模型(entity-relationship model)即实体联系模型,是非传统数据模型之一,于 1976 年由 P.Chen 提出[12].其主要有 3 个抽象概念.

(1) 实体:一般认为,客观上可以相互区分的事物就是实体,实体可以是具体的人和物,也可以是抽象的概念与联系,关键在于一个实体可与另一个实体相区别.用实体名及其属性名的集合来抽象和刻画同类实体;

(2) 属性:实体所具有的某一特性,一个实体可由若干个属性来刻画.属性不能脱离实体,属性是相对实体而言的;

(3) 联系,也称关系,信息世界中反映实体内部或实体之间的关联.实体内部的联系通常是指组成实体的各属性之间的联系;实体之间的联系通常是指不同实体集之间的联系.

E-R 模型实例如图 4 所示.由于 E-R 图直观易懂,在概念上表示了一个数据库的信息组织情况,所以若能画出 E-R 图,意味着彻底搞清了问题,此后就可以根据 E-R 图,结合具体 DBMS 的类型,把它演变为 DBMS 所能支持的数据模型.这种逐步推进的方法已经普遍用于数据库的设计中,成为数据库设计的一个重要步骤.

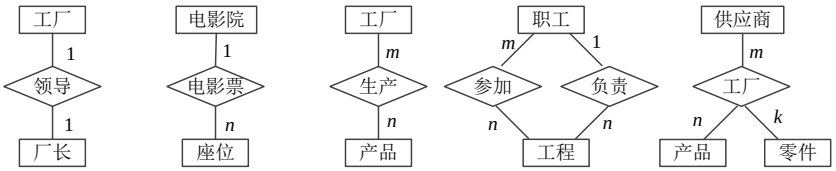


Fig.4 E-R model instance

图 4 E-R 模型实例

通过 E-R 图,计算机专业人员与非计算机人员可以进行交流合作,以真实、合理地模拟一个单位,作为进一步设计数据库的基础.在 E-R 图中:实体用矩形表示,矩形框内写明实体名,如果是弱实体的话,在矩形外面再套实线矩形;属性用椭圆形表示,并用无向边将其与相应的实体连接起来;联系用菱形表示,菱形框内写明联系名,并用无向边分别与有关实体连接起来,同时在无向边旁标上联系的类型(1:1,1:n 或 m:n).比如,老师给学生授课

存在授课关系,学生选课存在选课关系.

1.5 面向对象模型

基于面向对象模型的数据库属于非传统的数据库.与传统的数据库和 E-R 模型相比,面向对象数据库适合存储不同类型的数据,如图片、声音、视频、文本、数字等.因为,其结合了面向对象程序设计与数据库技术,提供了一个集成应用开发系统.

基于对象定义语言描述的面向对象模型允许使用以下数据类型.

(1) 原子类型:包含整型、浮点型、字符型、字符串、布尔型和枚举型,定义方式与 C 语言类同;

(2) 结构类型:用 *Struct* $N\{T_1\ F_2,T_2\ F_2,\ldots,T_N\ F_N\}$表示,其中,*Struct* 是关键字;$N$ 是结构名;$T_1,T_2,\ldots,T_N$ 是类型,一般是原子类型;$F_1,F_2,\ldots,F_N$ 是域名;第 i 个域名是 F_i,类型是 T_i;

(3) 聚集类型:一般由类型相同的原子类型或结构类型的元素聚集而成,包含集合、包、列表和数组;

(4) 接口类型:即类类型,面向对象模型定义的类型.

面向对象模型没有准确的定义,因为该名称已经应用到很多不同的产品和原型中,而这些产品和原型考虑的方面可能不一样,所以很难提供一个准确的定义来说明面向对象 DBMS 应建成什么样.

面向对象模型的数据结构是非常容易变化的.与传统的数据库(如层次、网状或关系)不同,面向对象模型没有单一固定的数据结构.编程人员可以给对象类型定义任何有用的结构,如链表、集合、数组等.此外,对象可以包含可变的复杂度,利用多重类型和多重结构.面向对象数据模型的实例如图 5 所示.

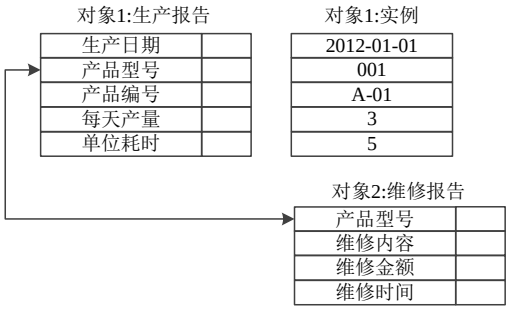


Fig.5 Object-Oriented model instance

图 5 面向对象模型实例

面向对象数据模型提供强大的特性,如继承、多态和动态绑定,允许用户不用编写特定对象的代码即可构成对象并提供解决方案,提高了数据库应用程序开发人员的开发效率;其次,面向对象数据模型明确地表示联系,支持导航式和关联式两种方式的信息访问,比基于关系值的联系更能提高数据访问性能;再次,面向对象数据模型适合于需要管理数据对象之间存在复杂关系的应用,特别适合特定的应用,如工程、电子商务、医疗等.

但是随着组织信息需求的改变,对象的定义也要求改变,并且需移植现有数据库,以完成新对象的定义,面向对象数据模型维护困难;其次,面向对象数据模型并不适合所有应用,当其被用于某些应用时,其性能将会降低.在采用面向对象模型的数据库系统中,典型的有 Gemstone[13]、Ontos[14]、O2[15]、Itasca[16]等:GemStone Systems 于 1982 年 3 月 1 日成立,名为 Servio Logic,于 1995 年 6 月更名为 GemStone Systems,Gemstone 系统是第一个作为面向对象的数据库;Ontos 系统是美国 Ontologic 公司用 C++语言开发的,采用多 C/S 体系结构,每个用户进程处理一个逻辑数据库;O2 是法国 Altair 公司研制开发的,其设计目标是集成面向对象程序技术和数据库技术,支持 CAM、CAD 等高级应用;Itasca 系统是 Itasca 公司在 ORION 系统基础上研发的商业化系统,采用基于对象服务器的多服务器多客户的分布式体系结构,支持复合对象和版本管理.

2 半结构化模型

结构化模型虽然可以很好地表达现实世界实体之间的关系,但对于诸如文本文件、超链接、HTML 文档等

半结构化数据缺乏有效的支持.不同于结构化模型,半结构化模型的特点主要有隐含的模式信息、结构不规则、缺乏严格的类型约束等[17].半结构化模型包括 XML 模型、RDF 模型、JSON 模型、图模型和超模型等.

2.1 XML模型

可扩展标记语言(extensible markup language,简称 XML)是一种标记语言被作为互联网信息交换的标准[18], XML 是一种分层自描述模型,具有良好的语义和可扩展性,可以灵活地表示和组织数据,并提供高效的查询方法,例如 XPath[19]、XQuery[20]、关键字查询[21,22]、子树匹配[23,24]等.此外,XML 模型在数学(MathML[25])、化学(CML[26])、地理(GML[27])等领域也被广泛应用.

XML 模型是由若干带有标签的节点组成的有向树,具体定义如下.

(1) 元素节点:该类型的节点为 XML 树中的标签;

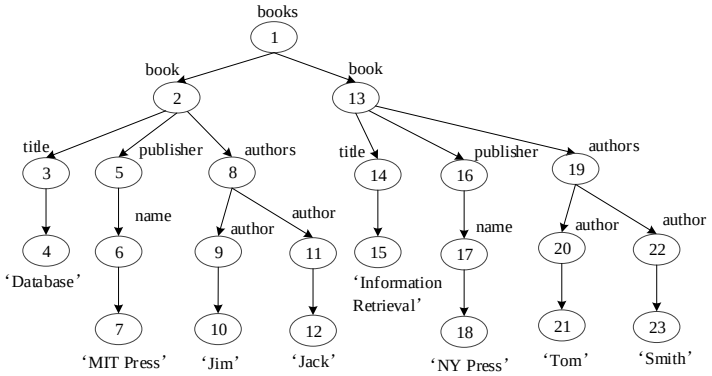
(2) 属性节点:该类型的节点为 XML 树中标签相关的属性.不同于元素节点,属性节点不是嵌套的,即,属性节点不能有任何子元素.相同的属性节点不能嵌套在同一个元素节点中,并且属性节点是无序的;

(3) 值节点(叶子节点):该类型的节点为标签的值;

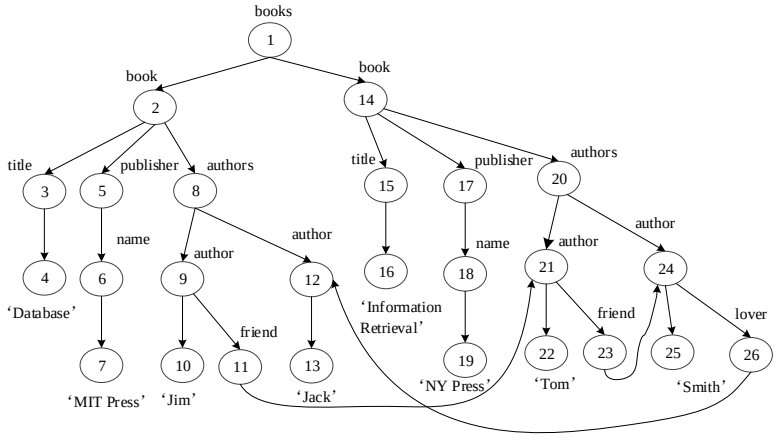
(4) 有向边描述了各类型节点之间的关系.

此外,XML 图模型允许用户引入 ID/IDREF 属性来对树模型进行扩展,其中,ID 属性唯一地标示了某个元素;IDREF 引用其他被 ID 属性标示的元素,构成一个有向无环图.

XML 模型如图 6 所示.



(a) XML 树模型



(b) XML 图模型

Fig.6 XML model instance

图 6 XML 模型实例

在图 6(a)所示的 XML 树模型中,book 节点表示一个元素,title 节点表示 book 的一个属性,而 Database 节点表示 book 的值.在图 6(b)所示的 XML 图模型中,author=“Jim”的节点通过 ID 属性指向了他的 friend 节点(author=“Tom”).

基于 XML 模型的数据库有 BaseX[28]、eXist[29]、MarkLogic Server[30]等:BaseX 是一个开源的轻量级的 XML 数据管理系统,提供 XQuery 查询,常用于 XML 数据的存储、查询和可视化等.同样,eXist 也是一个开源的 XML 源生数据库,除了支持 XML 数据,还支持 JSON、HTML 以及二进制文档的存储和管理;MarkLogic Server 是一个面向对象的 XML 数据库,支持多维模型,支持 JSON 和 RDF 数据.

2.2 RDF模型

RDF(resource description framework)模型又称为资源描述框架,本质上是一个数据模型,它提供了一个统一的标准来描述 Web 上的资源,所谓的资源可以是指类(class)、属性(property)、实例(Instance)等等.RDF 在形式上表示为一个三元组,即主语 s(subject)、谓词 p(predicate)、宾语 o(object),以描述具体的事物及关系.RDF 也可以表示为一张带有标记的有向图,图中有节点和边,节点对应实体,边对应关系或者属性,关系指的是实体之间、实体与属性之间的关系,其形式化描述如下.

RDF 三元组:给定一个 URI 集合 R、空节点集合 B、文字描述集合 L,一个 RDF 三元组 t 是形如(s,p,o)的三元组,其中,$s\in R\cup B,p\in R,o\in R\cup B\cup L$.这里的 s 通常称为主语(subject)、资源(resource)或主体,p 称为谓词(predicate)或属性(property),o 称为宾语(object)、属性值(value)或客体.例如,知识图谱中的一条知识:“人工智能之父是图灵”,其中,“人工智能”是主语,“之父是”是谓语,“图灵”是宾语.

RDF 模型以三元组的形式描述资源,简洁明了,并且令使用元数据的软件可以更容易和快速地制造.同时,索引是基于元数据而不是从正文得来,搜索者将得到更精确的搜索结果.在应用中,RDF 模型可以用来表示 Web 上的任何被标识的信息,并且使得它可以在应用程序之间交换而不丧失语义信息.因此,RDF 模型成为语义数据描述的标准,被广泛应用于元数据的描述、本体及语义网中很多机构和项目,如 Wikipedia、DBLP 等都用 RDF 表达它们的元数据.

2.3 JSON模型

虽然 XML 模型有统一的格式和标准、可提供多种复杂查询方法等优点,但数据文件较大,格式较复杂,不利于互联网上的数据传输.JSON(JavaScript object notation)是一种易于读写的轻量级数据表示格式,可以被快速、高效地解析[31,32].JSON 使用文本来表示 JavaScript 对象的信息,支持字符串、数字、对象、数组等各种类型,并且提供快速读取数据的方法,被广泛用在数据采集[33]、数据挖掘[34]中.

JSON 模型中有两种结构.

(1) 对象:一个对象包含一系列非排序的键-值对,一个对象以“{”开始,并以“}”结束.每个键-值对之间使用“:”区分;

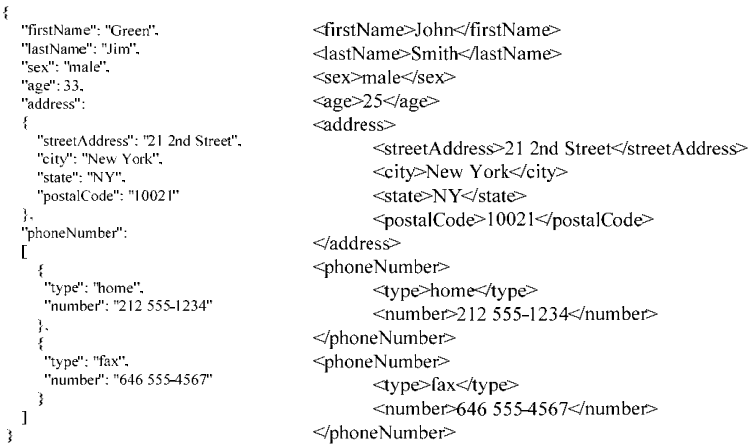
(2) 数组:一个数组是若干值的集合,一个数组以“[”开始,并以“]”结束.数组成员之间使用“,”区分.

例如,在图 7(a)所示的 JSON 模型中,“address”表示了一个数组,存储了若干键-值对,即“streetAddress”:“21 2nd Street”, “city”:“New York”等.此外,JSON 模型可以与 XML 模型进行相互转换[35],例如,图 7(a)所示的 JSON 文档可以转换成图 7(b)所示的 XML 模型.

虽然 JSON 和 XML 都可以完整地表示数据,并且可以相互转换,但它们之间存在以下不同之处.

- 首先,XML 是一种标记语言,而 JSON 是由若干键/值对组成的集合,这使得应用程序在读取 XML 时需要更多的时间开销;
- 其次,XML 的设计理念与 JSON 不同.XML 利用标记语言的特性提供了良好的延展性(如 XPath),并且提供高级检索,包括关键字查询、满足特定语义(如 SLCA)查询等操作;而 JSON 相比于 XML 更加轻量级,可以被快速解析,使其更适用于互联网中的数据传输.

JSON 模型多用于大数据的存储,支持 JSON 模型的数据库主要有 MongoDB、CouchDB 等.



(a) JSON 文档举例 (b) 转换后的 XML 文档

Fig.7 JSON and XML conversion examples

图 7 JSON 和 XML 相互转换举例

2.4 图模型

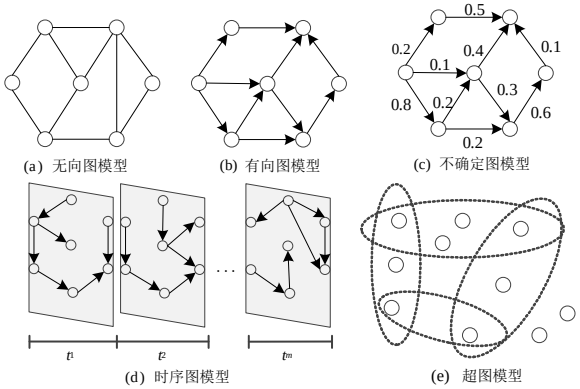
虽然 JSON 模型可以有效地解决互联网中的数据传输问题,但在表示复杂实体关系时的可读性较差.针对这一问题,研究人员基于图论提出了图模型来存储和表示实体间关系[36,37].图模型是一种良好的数据表现形式,并提供多种查询方法,例如最短路径查询[38]、子图同构[39]等.图模型的应用十分广泛,如社交网络、知识图谱、时序数据管理等.基本的图模型可以分为无向图模型和有向图模型.随着研究的进展,图模型又细分为不确定图模型[40−43]、超图模型[44−48]、时序图模型[49−52].

(1) 图 $G=(V,E,\Sigma)$,其中,V 是节点集合,E 是边集合,表示节点之间的关系,Σ表示标签集合.更进一步地,如果图中的边是没有方向的,如图 8(a)所示,则称为无向图;否则,称为有向图,如图 8(b)所示;

(2) 不确定图 $G'=(G,P)$,其中,G 表示一个有向图.对于任意边 $e\in E$,$P(e)\to(0,1]$表示 e 存在的概率.特别地,$P(e)=1$ 表示边 e 确定存在.不确定图模型如图 8(c)所示;

(3) 时序图是一种随时间而变化的图模型,一般是有向的.时序图可以被看作是一组有向图序列 $TG=\{G_{t_1},G_{t_2},...,G_{t_m}\}$,其中,$G_{t_x}$ 表示在时间点 t_x 时的图 G,时序图模型如图 8(d)所示;

(4) 超图 $H=(X,E)$,其中,X 表示一个有限集合 S,S 中的元素称为节点 e;E 是 X 的一个非空子集,称为超边或连接.超图模型如图 8(e)所示.



(a) 无向图模型 (b) 有向图模型 (c) 不确定图模型

(d) 时序图模型 (e) 超图模型

Fig.8 Graph model example

图 8 图模型举例

基于图模型的数据库有 AllegroGraph[53]、DEX[54]、HyperGraphDB[55]、Neo4j[56]等.AllegroGraph 是一个商业型的数据库,提供图模型存储数据,提供多种语言的 API 接口,并支持 SQL 语言.DEX 是一个轻量级的、可扩展的、高性能的图数据库,支持多种操作系统.HyperGraphDB 是一种通用的开源数据存储机制,提供高效的数据管理方式.Neo4j 是最常用的图数据管理系统,具有原生图存储机制,并支持 ACID 事务.

2.5 超模型

超模型与其他数据模型一样,超模型由一组超实体以及定义在它们上面的关系和约束组成,其差别是超模型必须由一个或多个模型来实现.其中,超实体被定义为实体或超实体类型的集合.超实体像实体一样,除试图捕获更高级的信息以外,它还可以有属性、协议以及参加各种关系等.超模型为复杂的实体模型建模提供了快捷的方法,其形式表示如图 9 所示.

```
Hyperentity Type<类型名> = {
  Abstraction { <实体类型表> }
  Attributes (属性表)
  Protocol(方法描述表)
  }
```

Fig.9 Super entity representation

图 9 超实体表示形式

以某集团公司为例,超实体集团公司是公司、产品、市场等实体(或超实体)类型的抽象,它还可以定义一些有意义的属性,如利润、税金等,我们给出的定义如图 10 所示.

```
Hyperentity Type Corporation = {
  Abstraction {
  Company, Officer, Product, Market, Strategy, Function
  }
  Attributes
  Revenue: Real
  Profit: Real
  …
  Protocol ?
  }
```

Fig.10 Instance of super entity

图 10 超实体实例

3 数据分析模型

3.1 OLAP模型

随着电子商务、商业智能等应用的不断发展,关系数据库之父 E. F. Codd[57]于 1993 年提出了联机分析处理(on line transaction processing,简称 OLAP)的概念.Codd 认为,联机事务处理(on-line transaction processing,简称 OLTP)[58]已不能满足终端用户对数据库查询分析的需要,SQL 对数据库进行的简单查询也不能满足用户分析的需求,故提出了多维数据库和多维分析的概念.数据分析模型主要包括关系型 ROLAP 和多维型 MOLAP[59]:ROLAP 模型主要研究事实表和维表的组织表示及其变种,包括星型模型、雪花模型等;MOLAP 模型主要研究数据立方及其优化计算算法.

为了严谨地综述 OLAP 模型[60],下面我们对一些 OLAP 模型用到的定义给出形式化描述.

(1) 度量:度量 u 是一个独立变量,它们参照每个维的某一维值,并作为 OLAP 的分析对象.度量的粒度是度量参照的维值所在的维级别,最细粒度的度量参照每一维 d 中的最低维级别的某一维值.设 u 参照维集合 $D=\{d_1,d_2,\dots,d_n\}$,$\forall d\in D$,即集合 D 可以确定度量 u,记作 $D\rightarrow u$,则满足 $D\rightarrow u\Leftrightarrow\exists!v\in md(lmd)\wedge v\rightarrow u$($d$ 有 m 个维级别),其中,$v\rightarrow u$ 是指维值 v 可以确定度量 u;

(2) 单元格:在逻辑视图中,单元格是由若干不同的度量组成的原子单元,这些度量都参照相同的维值.对于维集合 D 而言,单元格可以表示为度量的集合,记作$\{u|D\rightarrow u\}$;

(3) 数据立方:根据定义 1~定义 3,数据立方是 OLAP 中的多维数据结构,简称立方;

(4) 块:块是数据立方的逻辑划分,一个数据立方可以根据维的取值分成多个块.

3.2 ROLAP模型

ROLAP(relational OLAP)[61,62]是基于关系数据库的 OLAP 技术.ROLAP 以关系型结构进行多维数据的表示和存储.ROLAP 将多维数据库的多维结构划分为两类表:一类是事实表,用来存储数据和维关键字;另一类是维表,即对每个维度使用一个或多个表来存放层次、成员类别等维度的描述信息.ROLAP 模型主要包括星型模型、雪花模型.

星型数据模型是数据仓库和联机分析处理使用的最基本、最常用的数据模型,它能准确、简洁地描述出实体之间的逻辑关系.一个典型的星型模型包括一个大型的事实表和一组逻辑上围绕这个事实表的维度表.事实表是星型模型的核心,其中存放的大量数据,是同主题密切相关的、用户最关心的度量数据.维度是观察事实、分析主题的角度.维度表的集合是构建数据仓库数据模式的关键.维度表通过主键与事实表相连.用户依赖维表中的维度属性,从事实表中获取支持决策的数据.

图 11 所示为一个酒店经营数据仓库星型数据模型,记载了每位客人的消费形式、消费价格、促销方式、促销折扣、消费金额、成本、利润等度量数据.围绕经营主题,酒店经营数据仓库建立了客户维、消费项目维、时间维和促销维这 4 个维度.

当有一个或多个维表没有直接连接到事实表上,而是通过其他维表连接到事实表上时,其图解就像多个雪花连接在一起,故称雪花模型.雪花模型是对星型模型的扩展,它对星型模型的维表进一步层次化,原有的各维表可能被扩展为小的事实表,形成一些局部的“层次”区域,这些被分解的表都连接到主维度表而不是事实表.如图 12 所示,对消费项目维度的餐饮种类进行划分、对促销维度打折程度进行划分等.雪花型数据模型通过最大限度地减少数据存储量以及联合较小的维表来改善查询性能,除了数据冗余.

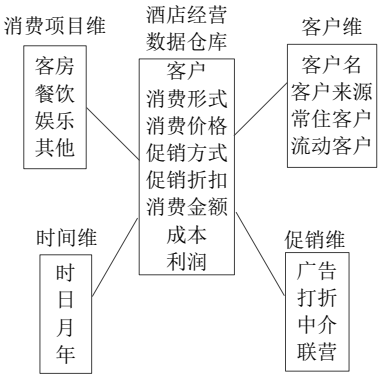


Fig.11 Star data model instance

图 11 星型数据模型实例

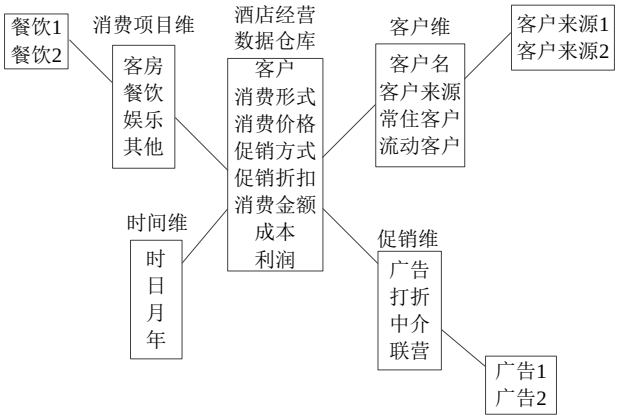


Fig.12 Snowflake data model instance

图 12 雪花型数据模型实例

微软 Access 的数据透视表(pivot table)[63]就是采用 ROLAP 数据模型的一个例子.Pivot Table 是一种交互式的表,可以根据数据在数据透视表中的排列进行求和与计数等操作.同时,数据透视表可以动态地改变它们的版面布置,以便按照不同方式分析数据,也可以重新安排行号和页字段.当每一次改变版面布置时,数据透视表会立即按照新的布置重新计算数据.

3.3 MOLAP模型

ROLAP是将用户的OLAP操作转换成SQL语句提交到数据库中执行,并且提供聚集导航功能,根据用户操作的维度和度量,将SQL查询定位到最粗粒度的事实表上.ROLAP提供了更大的灵活度,但响应速度较慢,因此提出了MOLAP(multidimensional OLAP)[64,65],这是一种基于多维数据组织的OLAP技术.

MOLAP将OLAP分析所用到的多维数据以多维数组的形式存储,形成“立方体”的结构.MOLAP事先将汇总数据计算好,存放在自己特定的多维数据库中,用户的OLAP操作可以直接映射到多维数据库的访问,不通过SQL访问,加快了响应速度.

图13所示为某一超市连锁店的销售数据立方体,其中,用户关心的是销售量,并习惯从时间、地点和商品这3个角度来分析销售数据.在这里,销售量称为度量,时间、地点和商品称为维度.如图13所示,时间维、地点维和商品维的层次分别是年、城市和商品种类.这3个维层次以及由维到度量的映射关系决定的度量数据构成了一个数据立方体实例.

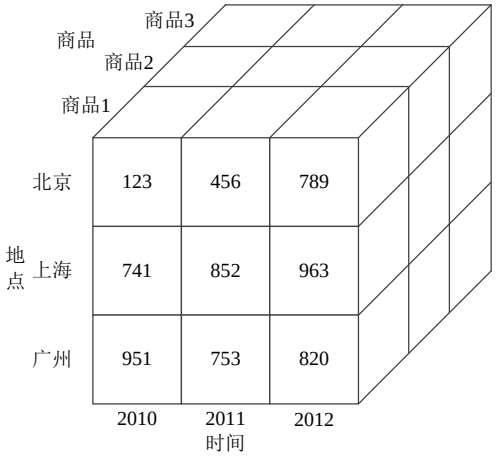


Fig.13 Data cube instance based on time, place, product

图13 基于时间、地点、商品的数据立方体实例

采用MOLAP模型的数据库系统的代表产品有Hyperion(原Arbor Software) Essbase[66]等.Hyperion Essbase平台是全球领先的企业绩效管理(business performance management,简称BPM)解决方案提供商.借助此平台,企业可以制定战略方针、构建业务模型,并有效规划企业资源.业务智能平台同时可以监控由上述因素所推动的业务改进流程,并提供详细报表.这样不仅可以分析出推动企业成长的主要动力,还可以预测核心业务的前景.

3.4 Storm模型

Storm[67,68]是一个免费并开源的分布式实时计算系统,利用Storm,可以很容易做到可靠地处理无限的数据流.像Hadoop批量处理大数据一样,Storm可以实时处理数据.Storm操作简单,可以使用任何编程语言.

Storm实现了一个数据流(data flow)的模型,在这个模型中,数据持续不断地流经一个由很多转换实体构成的网络.一个数据流的抽象叫作流(stream),流是无限的元组(tuple)序列.元组就像一个可以表示标准数据类型(例如int、float和byte数组)和用户自定义类型(需要额外序列化代码的)的数据结构.每个数据流由一个唯一的ID来标示,这个ID可以用来构建拓扑中各个组件的数据源.

Storm[69]模型对数据输入的来源和输出数据的去向没有作任何限制.像Hadoop,是需要把数据放到自己的文件系统HDFS里的.在Storm里,可以使用任意来源的数据输入和任意的数据输出,只要实现对应的代码来获取/写入这些数据即可.

Storm模型具有编程简单、低延迟、可扩展性强、容错性高和消息不易丢失等优点.Storm模型提供的编

程原语与 Hadoop 类似,也很简单,开发人员只需要关注应用逻辑;Storm 模型可以分布式处理,轻松应对数据量大、单机搞不定的场景;并且随着业务的发展,数据量和计算量越来越大,Storm 模型可以水平扩展.

4 大数据模型

随着互联网、物联网、社交网络等技术的飞速发展,传统的数据模型已经无法应对数据量的爆炸式增长.为了解决因这些海量数据造成的存储瓶颈以及管理复杂性等问题,以 NoSQL 和 NewSQL 数据库系统为代表的大数据模型逐渐成为新的研究热点.

4.1 NoSQL模型

NoSQL 模型是指非关系型、不遵循 ACID 原则的存储模型[70,71].NoSQL 模型遵循 CAP 理论[72]和 BASE 原则[73].CAP 理论指出:任何分布式系统无法同时满足一致性(consistency)、可用性(availability)和分区容错性(partition tolerance),最多只能满足其中的两个.而 BASE 指出,分布式系统在设计时需要考虑基本可用性(basically available)、软状态(soft state)和最终一致性(eventually consistent).

NoSQL 模型主要有 3 类,即 Key-Value 模型、Key-Column 模型、Key-Document 模型.

4.1.1 Key-Value 模型

Key-Value 模型的主要思想主要来自于哈希表.Key-Value 模型由一个键-值映射的字典构成.Key-Value 不仅支持字符串类型,还支持字符串列表、无序(或有序)不重复的字符串集合、键-值哈希表.Key-Value 通常将数据存储在内存中,从而提高运算速度.此外,Key-Value 模型又可以细分为临时性和永久性两种类型.临时性 Key-Value 模型中所有操作都在内存中进行,这样做的好处是读取和写入的速度非常快,但一旦数据库实例关闭后,将会丢失所有数据.临时性 Key-Value 模型的数据库通常作为高效缓存技术应用在高并发场景.而永久性 Key-Value 模型会将数据写入到硬盘上,这个过程中会造成 I/O 开销,导致性能较差,但数据不会丢失.

图 14 给出了一个 Key-Value 模型举例,其中,键 $k1$ 对应的值 $value$={11,22,33},键 $k2$ 对应的值是一个字符串数组{Name:Jim,Tel:1234}.综上可以看出,Key-Value 模型支持任意格式的值存储.

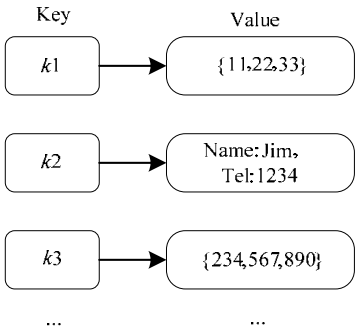


Fig.14 Key-Value model example

图 14 Key-Value 模型举例

基于 Key-Value 模型的数据库实例主要有 Memcached[74]、Redis[75]、LevelDB[76]等:Memcached 是一个通用的分布式内存缓存系统,通常用于缓存数据和对象,以减少读取外部数据源(如数据库或 API)的次数;Redis 是一款开源内存数据库项目,实现了分布式内存键-值存储和可选持久性,提供字符串、列表、位图和空间索引;LevelDB 是一个开源的 Key-Value 数据库,实现了快速读、写机制,并提供键-值之间的有序映射.

4.1.2 Key-Document 模型

虽然 Key-Document 模型可以快速地访问数据,但当数据规模较大、无固定模式时,读写的效率会明显降低.Key-Document 模型的核心思想是“数据用文档(如 JSON)来表示”,JSON 文档的灵活性使得 Key-Document 模型

数据适合存储海量数据.

Key-Document 模型如图 15 所示,Key-Document 模型是“面向集合”的,即数据被分组存储在数据集中,这个数据集称为集合(collection).每个集合都有一个唯一标识,并且存储在集合中的文档没有数量限制.Key-Document 模型中的集合类似于关系型数据库中的表结构.所不同的是,Key-Document 模型中无需定义模式(schema).

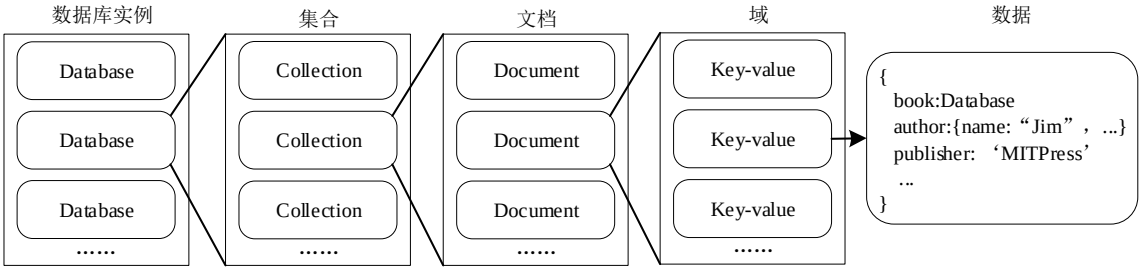


Fig.15 Key-Document model example

图 15 Key-Document 模型举例

基于 Key-Document 模型的数据库实例主要有 MongoDB[77]、CouchDB[78]等.MongoDB 是开源的、跨平台、面向文档的数据库,使用 JSON 文档和模式存储数据.CouchDB 是开源的、基于 Key-Document 模型的数据库,专注于易用性和可扩展的体系结构,它使用 JSON 来存储数据.

4.1.3 Key-Column 模型

虽然 Key-Value 模型和 Key-Document 模型在特定的场景下得到了广泛的应用,但它们对范围查询、扫描等操作的效率较低.Key-Column 模型是一个稀疏的、分布式的、持久化的多维排序图,并通过字典顺序来组织数据,支持动态扩展,以达到负载均衡.存储在 Key-Column 模型中的数据可以通过行键(row key)、列键(column 键)和时间戳(timestamp)进行检索.其中,列族(column family)是最基本的访问单位,存放在相同列族下的数据拥有相同的列属性,并使用时间戳来索引不同版本的数据,以避免数据的版本冲突问题.同时,用户可以通过指定时间戳来获得不同版本的数据.

图 16 给出了通过 Key-Column 模型来存储网页的示例.

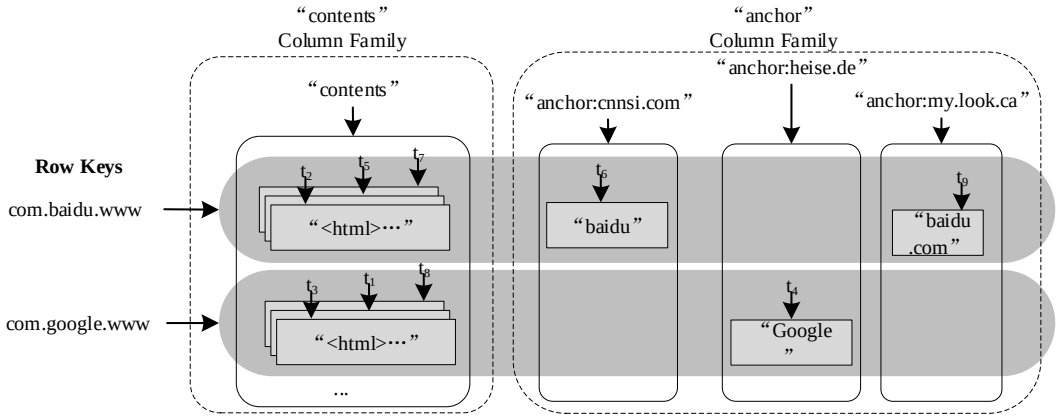


Fig.16 Key-Column model example

图 16 Key-Column 模型举例

可以看出:不同于关系型数据库中的表结构,Key-Column 模型中的表是一个“多维 Map”结构.每个行都代表了一个对象,由一个行键,如“com.baidu.www”,和一个或多个列组成,如“contents”.相关的行键标识的行对象存

储在相邻的位置.每个列由列族和列标示符组成,并用":"隔开,例如 anchor:cnnsi.com.单元格是行、列族和列标识符的组合,并且包含了一个值和一个时间戳来标示数据的版本,如 *t*2.

基于 Key-Column 模型的数据库实例主要有 BigTable[79]、HBase[80]、Cassandra[81]等:BigTable 是 Google 公司推出的一种高扩展性的分布式数据库,提供列压缩等功能,被用于存储海量数据;HBase 是 BigTable 的开源实现,提供了压缩算法、内存操作和布鲁姆过滤器等功能;Cassandra 最初是由 Facebook 开发的一款开源的 Key-Column 模型数据库,支持海量数据的读、写,拥有较高的可扩展性,并提供类 SQL 语言来操作数据.

4.2 NewSQL模型

NewSQL 被称作"下一代可扩展关系型数据库",为传统数据库系统提供了接近 NoSQL 的处理性能,使用 SQL 语言作为应用之间交互的主要机制,并且遵循 ACID 特性[82,83].值得一提的是:NewSQL 并没有提出新的数据模型,而是将传统的关系型模型和 NoSQL 模型的优点相结合.NewSQL 使用"无锁"的并发控制机制,使得应用程序在实时读取数据时不会被阻塞.此外,部署了 NewSQL 的分布式集群中的每个节点都可以处理请求,从而提高了效率,且高于传统 RDBMS 解决方案.NewSQL 使用了无共享的架构,可以方便地部署在海量节点中,并提供了良好的水平扩展性,从而使得系统不受性能瓶颈的影响[84,85].

为了更好地比较 RDBMS、NoSQL 和 NewSQL 的特点,表 1 从不同的角度分析了三者之间的优劣.通过比较结果可以看出:NewSQL 结合了传统关系型数据库和 NoSQL 数据库的优点,提供了管理海量数据的新方法,这也是未来的一个重要研究方向.

Table 1 Comparison of RDBMS, NoSQL, and NewSQL features

表 1 RDBMS、NoSQL 和 NewSQL 特点比较

	RDBMS	NoSQL	NewSQL
SQL	支持	不支持	支持
宿主机	单机	多机/分布式	多机/分布式
类型	关系型	非关系型	关系型
模式	表	key-(value,column,document)	二者都支持
物理存储	磁盘+缓存	磁盘+缓存	磁盘+缓存
特性	ACID	CAP,BASE	ACID
查询复杂度	低	高	高
一致性	高	最终一致性	高
可用性	故障转移	高	高
可扩展性	垂直扩展	水平扩展	水平扩展
复制	可配置	可配置	自动
安全性	高	低	低
大数据处理	支持,但效率低	支持	充分支持
OLTP	不完全支持	支持	充分支持

基于 NewSQL 模型的数据库主要有 H-Store[86]、S-Store[87]、Spanner[88]、VoltDB[89]等:H-Store 是一款内存并行数据库管理系统,是一个高度分布的基于行存储的关系数据库,运行在无共享的群集上;S-Store 是一个全球化的流式 OLTP 引擎,将在线事务处理和实时数据流处理相结合,并支持事务的 ACID 特性和事务的有序性;Spanner 是 Google 公司推出的全球化的基于 BigTable 模型的 NewSQL 数据库,在原有 BigTable 模型的基础上实现了事务的 ACID 特性;VoltDB 是一个横向可扩展的 NewSQL 关系数据库,支持 SQL 访问和事务的 ACID 特性,并基于连续快照和命令日志记录的组合实现数据持久性.

5 数据库实例的分析与比较

本节选取每个模型的典型实例分析特点[90],并通过实验对比了大数据模型实例的效率.

5.1 结构化模型

在结构化模型中,选取开源关系型数据库实例(MySQL 和 PostgreSQL)和商业型关系型数据库(Oracle 和 SQL Server)作为分析对象.

表 2 给出了开源的关系型数据库 MySQL 和 PostgreSQL 的分析结果:相比于 MySQL,PostgreSQL 允许对象模型的存储,具有良好的可扩展性,可扩展到多节点中进行部署;而 MySQL 不支持物化视图和对象存储.这两种数据库都支持 GUI 接口,都支持多种操作系统环境,也都提供临时表.

Table 2 Comparison between MySQL and PostgreSQL

表 2 MySQL 和 PostgreSQL 对比

	MySQL	PostgreSQL
概念	关系型数据库管理系统	对象-关系型数据库管理系统
运行环境	Windows, Mac OS X, Linux, BSD, UNIX, z/OS, Symbian, AmigaOS	Windows, Mac OS X, Linux, BSD, UNIX, z/OS, Symbian, AmigaOS
可扩展性	不支持	支持
接口	提供 GUI	提供 GUI
备份工具	Mysqldump, XtraBackup	在线备份
物化视图	提供临时表,不提供物化视图	提供临时表,且提供物化视图
数据对象	不支持	支持

表 3 给出了商业的关系型数据库 Oracle 和 SQL Server 的分析结果.

- 首先,这两种关系型数据库所使用的查询语言不同.Oracle 使用 PL/SQL,SQL Server 使用 T-SQL;
- 其次,这两种关系型数据库中的事务提交方式不同.在 Oracle 中,事务的提交依赖于管理员(DBA)发出 Commit 命令,如果没有 Commit 命令,则事务无法提交;而在 SQL Server 中,即使没有确切的 Commit 命令,事务中的命令也可以被独立地运行;
- 在 Oracle 中,所有的数据库在用户之间是共享的,Oracle 通过权限管理来限制用户的访问权限;而在 SQL Server 中,数据库是私有的,无法在多个用户之间共享;
- Oracle 支持多种操作系统;而 SQL Server 只能运行在 Windows 或 Linux 上,通用性较差.

Table 3 Comparison of Oracle and SQL Server

表 3 Oracle 和 SQL Server 对比

	Oracle	SQL Server
操作语言	PL/SQL (procedural language/SQL)	T-SQL (Transact-SQL)
事务	除非管理员(DBA)发出 COMMIT 命令,否则事务无法提交	如果事务中没有明确的操作(BEGIN 或 COMMIT),命令将被独立地执行
管理方法	数据库在所有模式和用户之间共享	用户之间不共享数据库
API 包管理	过程、函数和变量被组合成包	包不在 SQL 中
运行环境	Windows, Linux, Solaris, HP-UX, OS X, z/OS, AIX	Windows 或 Linux

5.2 半结构化模型

在半结构化模型中,选取 XML 模型实例(MarkLogic 和 BaseX)和图模型实例(Neo4j 和 HyperGraphDB)作为分析对象,结果见表 4.如前所述,由于 JSON 模型常用于大数据的存储,这里没有给出 JSON 模型实例分析结果.

在 XML 模型实例分析中,MarkLogic 支持 SQL 查询语言,支持数据划分策略并提供数据副本存储方法.在这些方面要优于 BaseX;同时,在 MarkLogic 中的事务支持 ACID 特性,而在 BaseX 中支持并发读数据,但支持单进程写数据;最后,MarkLogic 对内存计算有良好的支持,并提供范围索引.

在图模型实例分析中,Neo4j 不支持 SQL 查询语言,而 HyperGraphDB 支持 SparQL.Neo4j 没有提供数据划分策略,而 HyperGraphDB 支持数据分布到多个节点组成的联盟中.这两种图模型数据库中的事务都遵循 ACID 特性.最后,HyperGraphDB 提供良好的内存计算能力.

此外,通过比较结果可以看出,这 4 种半结构化模型实例都支持二级索引和触发器,同时支持并发操作和数据持久.

Table 4 Semi-Structured database instance comparison

表 4 半结构化数据库实例对比

数据模型	XML 模型		图模型	
数据库实例	MarkLogic	BaseX	Neo4j	HyperGraphDB
实现语言	C++	Java	Java, Scala	–
二级索引	支持	支持	支持	支持
SQL	支持	不支持	不支持	SPARQL
触发器	支持	支持	支持	支持
数据划分	分区	无	无	联盟
数据副本	多节点存储	无	遵循 Raft 协议	主从复制
外键	不支持	不支持	支持	不支持
事务特性	ACID	允许多读,单一写	ACID	ACID
操作并发	支持	支持	支持	支持
数据持久	支持	支持	支持	支持
内存计算	支持,提供范围索引	–	–	不支持

5.3 OLAP模型

在 OLAP 模型中,选取 ROLAP 模型实例(PivotTable)和 MOLAP 模型实例(Hyperion)进行分析,结果见表 5.可以看出:PivotTable 系统比 Hyperion 系统占用的存储空间要小,但 PivotTable 查询速度较慢.因此,一个新的 OLAP 结构 HOLAP 被提了出来.但是迄今为止,对 HOLAP 还没有一个正式的定义,HOLAP 不是 MOLAP 与 ROLAP 结构的简单组合,而是这两种结构技术优点的有机结合,可满足用户各种复杂的分析请求.HOLAP 的优越性就在于它能将 ROLAP 和 MOLAP 相互取长补短,充分利用 ROLAP 的灵活性和数据存储能力以及 MOLAP 的多维性和高效率.不同 OLAP 应用的优化目标也不同,有的应用优先考虑效率和相应时间,那么 MOLAP 的比重就应该加大,常用汇总数据都应该采用多维数据库来存储,有的应用对存储容量的要求较高,那么就应该充分利用关系数据库的存储能力,把大部分统计数据用 ROLAP 的模式来存储.

Table 5 Comparison between PivotTable and Hyperion

表 5 PivotTable 和 Hyperion 对比

	PivotTable	Hyperion
维度管理	增加一个维度只需增加一张维表,操作方便	增加维度需要重建数据库,操作复杂
数据存储	存储空间耗费小,维数没有限制	占用大量空间,因此难以达到 TB 级(只能 10GB~20GB)
数据加载	只需创建索引和概要表,更快捷	需要计算所有立方体中的数据,耗时长
查询性能	计算复杂,耗时长	采用索引搜索与直接寻址相结合的方法,速度快
维护管理	需要打开或者关闭索引,在数据加载和聚集时需要填充多个结构,维护困难	只需用 SQL 语句输入数据即可,管理简便

5.4 大数据模型

在大数据模型中,选取了 Key-Value 模型实例(Redis、Memcached 和 LevelDB)、Key-Document 模型实例(MongoDB、DynamoDB 和 CouchDB)、Key-Column 模型实例(HBase 和 Cassandra)进行分析.结果见表 6.

通过比较结果可以看出:Key-Value 模型实例不支持二级索引、SQL、触发器、外键等传统关系型数据库的操作;Redis 支持事务操作、数据划分和数据副本存储策略;这 3 种 Key-Value 模型实例都支持内存计算,从而获得更快的执行效率.

在 3 种 Key-Document 模型实例中,都支持二级索引技术、数据分区、操作并发和数据持久,但都不支持事务操作、SQL 查询语言和外键.DynamoDB 和 CouchDB 支持触发器.MongoDB 和 CouchDB 提供数据副本策略.MongoDB 支持内存计算.

在两种 Key-Column 模型实例中,都支持触发器、数据划分、数据副本存储、操作并发和数据持久,都不支持外键、事务操作和内存计算.Cassandra 支持二级索引,并提供类 SQL 查询语言(CQL).

Table 6 NoSQL database instance comparison

表 6 NoSQL 数据库实例对比

数据模型	Key-Value 模型			Key-Document 模型			Key-Column 模型	
数据库实例	Redis	Memcached	LevelDB	MongoDB	DynamoDB	CouchDB	HBase	Cassandra
实现语言	C	C	C++	C++	–	Erlang	Java	Java
二级索引	不支持	不支持	不支持	支持	支持	支持	不支持	受限制的
SQL	不支持	不支持	不支持	不支持	不支持	不支持	不支持	CQL
触发器	不支持	不支持	不支持	不支持	支持	支持	支持	支持
数据划分	分区	不支持	不支持	分区	分区	分区	分区	分区
数据副本	主从复制; 多主节点	不支持	不支持	主从复制	–	主\从复制; 主\主复制	可选择地 复制因子	可选择地 复制因子
外键	不支持	不支持	不支持	不支持	不支持	不支持	不支持	不支持
事务特性	支持	无	无	无	无	无	无	无
操作并发	支持	支持	支持	支持	支持	支持	支持	支持
数据持久	支持	不支持	支持	支持	支持	支持	支持	支持
内存计算	支持	支持	支持	支持	–	不支持	不支持	不支持

6 总 结

在数据库领域,数据模型用于表达现实世界中的对象,也就是将现实世界中杂乱的信息用一种规范而形象化的方式表达出来.经过近半个世纪的发展,数据模型形成了坚实的理论基础和广泛的应用领域.本文综述了结构化模型、半结构化模型、OLAP 分析模型和大数据模型这 4 种数据模型的概念、特点和国际上的主要研究进展,并选取每个模型的典型数据库系统进行了性能的分析.

结构化模型在 20 世纪 60 年代中后期被最早提出,包括了层次模型、网状模型、关系模型和面向对象模型等,主要应用有 MySQL、PostgreSQL、Oracle 和 SQL Server 等:PostgreSQL 允许对象模型的存储,比 MySQL 具有更为良好的可扩展性;Oracle 支持多种操作系统,而 SQL Server 只能运行在 Windows 或 Linux 上,通用性较差.20 世纪 90 代末期,随着互联网应用和科学计算等复杂应用的快速发展,开始出现半结构化模型,包括 XML 模型、JSON 模型和图模型等.主要的应用有 MarkLogic、BaseX、Neo4J 和 HyperGraphDB 等:MarkLogic 相较于 BaseX 支持 SQL 查询语言,支持数据划分策略并提供数据副本存储方法;而 HyperGraphDB 支持 SQL,比 Neo4J 内存计算能力更强.21 世纪,随着电子商务、商业智能等应用的不断发展,数据分析模型成为研究热点,出现了基于 ROLAP 模型的实例 PivotTable 和基于 MOLAP 模型的实例 Hyperion:PivotTable 比 Hyperion 占用存储空间更小,但 PivotTable 查询速度较慢.2010 年以来,随着大数据工业应用的快速发展,以 NoSQL 和 NewSQL 数据库系统为代表的大数据模型成为新的研究热点,主要应用实例有:基于 Key-Value 模型的 Redis、Memcached 和 LevelDB;基于 Key-Document 模型实例的 MongoDB、DynamoDB 和 CouchDB;基于 Key-Column 模型实例的 HBase 和 Cassandra 等.

目前,NewSQL 一直在尝试解决摆脱人工运维束缚,在存储层实现真正的自生长、自维护,同时实现用户可以用最自然的编程接口访问和存储数据等问题.解决这些问题以后,NewSQL 就可以摆脱存储的介质及容量的限制.在未来,NewSQL 将作为“下一代可扩展关系型数据库”,成为数据库主要的研究方向之一.

References:

[1] Meng XF, Zhou LX, Wang S. State of the art and trends in database research. Ruan Jian Xue Bao/Journal of Software, 2004,15(12): 1822−1836 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/20041208.htm

[2] Barnett AJ, Lightfoot JA. A S/360 on-line production order location and reporting system using the information management system (IMS). IN: Proc. of the IFIP Congress, Vol.2. 1968. 1192−1196.

[3] Taylor RW. Data administration and the DBTG report. In: Proc. of the 1974 ACM-SIGMOD Workshop on Data Description, Access and Control, Vol.2. Ann Arbor, 1974. 431−444.

[4] Codd EF. A relational model of data for large shared data banks. Communications of the ACM, 1970,13(6):377−387.

[5] Codd EF, Date CJ. Interactive support for non-programmers: The relational and network approaches. In: Proc. of the 1974 ACM-SIGMOD Workshop on Data Description, Access and Control, Vol.2. 1974. 11−41.

[6] Ronström M, Oreland J. Recovery principles in MySQL cluster 5.1. In: Proc. of the VLDB. 2005. 1108−1115.

[7] Stonebraker M, Rowe LA. The design of postgres. In: Proc. of the SIGMOD Conf. 1986. 340−355.

[8] Snell B, Hardman RG. DB2 performance and capacity planning case study. In: Proc. of the Int'l CMG Conf. 1986. 63−70.

[9] Gary H. The Oracle Warehouse. In: Proc. of the VLDB. 1995. 707−709.

[10] Chaudhuri S, Narasayya VR. An efficient cost-driven index selection tool for Microsoft SQL Server. In: Proc. of the VLDB. 1997. 146−155.

[11] Li JY. The Principle and Application System Development in Database. Beijing: China Water and Power Press, 2005. 20−22 (in Chinese).

[12] Chen PP. The entity-relationship model-toward a unified view of data. ACM Trans. on Database Systems, 1976,1(1):9−36.

[13] Penney DJ, Stein J. Class modification in the GemStone object-oriented DBMS. In: Proc. of the OOPSLA'87. 1987. 111−117.

[14] Soloviev V. An overview of three commercial object-oriented database management systems: ONTOS, ObjectStore, and O2. SIGMOD Record, 1992,21(1):93−104.

[15] Lécluse C, Richard P, Vélez F. O2, an object-oriented data model. In: Proc. of the DBPL'87. 1987. 257−276.

[16] Barry DK. ITASCA distributed ODBMS. In: Proc. of the SIGMOD Conf. 1992. 70.

[17] Wang J, Meng XF. Schema of semistructured data: A survey. Computer Science, 2001,28(2):6−10 (in Chinese with English abstract).

[18] Quin L. Extensible Markup Language (XML). World Wide Web Consortium (W3C), 2006. http://www.w3.org/XML/

[19] W3C Consortium. XML Path Language (XPath) 2.0. 2006. http://www.w3.org/TR/xpath20/

[20] W3C Consortium. XQuery 1.0: An XML Query Language. 2006. http://www.w3.org/TR/xquery/

[21] Le TN, Ling TW. Survey on keyword search over XML documents. ACM SIGMOD Record, 2016,45(3):17−28.

[22] Liu Z, Chen Y. Processing keyword search on XML: A survey. World Wide Web, 2011,14(5-6):671−707.

[23] Gou G, Chirkova R. Efficiently querying large XML data repositories: A survey. IEEE Trans. on Knowledge and Data Engineering, 2007,19(10).

[24] Hachicha M, Darmont J. A survey of XML tree patterns. IEEE Trans. on Knowledge and Data Engineering, 2013,25(1):29−46.

[25] Carlisle D, Ion P, Miner R. Mathematical Markup Language (MathML) Version 3.0. World Wide Web Consortium (W3C). 2010. http://www.w3.org/TR/MathML/

[26] Murray-Rust P, Rzepa H. Chemical markup language CML. 1995. http://www.xml-cml.org/

[27] Lake R, Burggraf DS, Trninic M, Rae L. Geography Mark-Up Language: Foundation for the Geo-Web. Wiley, 2004.

[28] BaseX. http://basex.org/

[29] eXist. http://exist-db.org/exist/apps/homepage/index.html

[30] MarkLogic. https://www.marklogic.com/

[31] JSON. json.org. http://www.json.org

[32] Bourhis P, Reutter JL, Suárez F, *et al*. JSON: Data model, query languages and schema specification. In: Proc. of the 36th ACM SIGMOD-SIGACT-SIGAI Symp. on Principles of Database Systems. ACM Press, 2017. 123−135.

[33] Kumar S, Morstatter F, Liu H. Twitter Data Analytics. New York: Springer-Verlag, 2014.

[34] Lin J, Ryaboy D. Scaling big data mining infrastructure: The Twitter experience. ACM SIGKDD Explorations Newsletter, 2013, 14(2):6−19.

[35] Nurseitov N, Paulson M, Reynolds R, *et al*. Comparison of JSON and XML data interchange formats: A case study. Caine, 2009,9: 157−162.

[36] Angles R, Gutierrez C. Survey of graph database models. ACM Computing Surveys (CSUR), 2008,40(1):1.

[37] Angles R. A comparison of current graph database models. In: Proc. of the 28th IEEE Int'l Conf. on Data Engineering Workshops (ICDEW). IEEE, 2012. 171−177.

[38] Wu H, Cheng J, Huang S, *et al*. Path problems in temporal graphs. Proc. of the VLDB Endowment, 2014,7(9):721−732.

[39] Bonnici V, Giugno R. On the variable ordering in subgraph isomorphism algorithms. IEEE/ACM Trans. on Computational Biology and Bioinformatics (TCBB), 2017,14(1):193−203.

[40] Zhang HY, Wang LW, Chen YX. Research progress of probabilistic graphical models: A survey. Ruan Jian Xue Bao/Journal of Software, 2013,24 (11):2476−2497 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4486.htm [doi: 10.3724/SP.J.1001.2013.04486]

[41] Larrañaga P, Moral S. Probabilistic graphical models in artificial intelligence. Applied Soft Computing, 2011,11(2):1511−1528.

[42] Getoor L. An introduction to probabilistic graphical models for relational data. IEEE Data Engineering Bulletin, 2006,29(1):32−39.

[43] Sucar LE. Probabilistic Graphical Models. London: Springer-Verlag, 2015. 978−981.

[44] Lyu B, Qin L, Lin X, *et al*. Scalable supergraph search in large graph databases. In: Proc. of the 32nd IEEE Int'l Conf. on Data Engineering (ICDE). IEEE, 2016. 157−168.

[45] Tong Y, Zhang X, Cao CC, *et al*. Efficient probabilistic supergraph search over large uncertain graphs. In: Proc. of the 23rd ACM Int'l Conf. on Information and Knowledge Management. ACM Press, 2014. 809−818.

[46] Zhang W, Lin X, Zhang Y, *et al*. Efficient probabilistic supergraph search. IEEE Trans. on Knowledge and Data Engineering, 2016, 28(4):965−978.

[47] Hellmuth M, Ostermeier L, Stadler PF. A survey on hypergraph products. Mathematics in Computer Science, 2012,6(1):1−32.

[48] Ausiello G, Laura L. Directed hypergraphs: Introduction and fundamental algorithms—A surve. Theoretical Computer Science, 2017,658:293−306.

[49] Kostakos V. Temporal graphs. Physica A: Statistical Mechanics and its Applications, 2009,388(6):1007−1023.

[50] Michail O. An introduction to temporal graphs: An algorithmic perspective. Internet Mathematics, 2016,12(4):239−280.

[51] Han W, Miao Y, Li K, *et al*. Chronos: A graph engine for temporal graph analysis. In: Proc. of the 9th European Conf. on Computer Systems. ACM Press, 2014. 1.

[52] Yan S, Xiong Y, Lin D. Spatial temporal graph convolutional networks for skeleton-based action recognition. arXiv Preprint arXiv: 1801.07455, 2018.

[53] Fernandes D, Bernardino J. Graph databases comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB. In: Proc. of the DATA 2018. 2018. 373−380.

[54] Martínez-Bazan N, Muntés-Mulero V, Gómez-Villamor S, Nin J, Sánchez-Martínez MA, Larriba-Pey JL. DEX: High-Performance exploration on large graphs for information retrieval. In: Proc. of the 16th Conf. on Information and Knowledge Management (CIKM). ACM Press, 2007. 573−582.

[55] Iordanov B. HyperGraphDB: A generalized graph database. In: Proc. of the Int'l Conf. on Web-Age Information Management. Berlin, Heidelberg: Springer-Verlag, 2010. 25−36.

[56] Webber J. A programmatic introduction to Neo4J. In: Proc. of the 3rd Annual Conf. on Systems, Programming, and Applications: Software for Humanity. ACM Press, 2012. 217−218.

[57] Codd EF. Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate. E.F. Codd and Associates, 1993.

[58] Vieira M, Madeira H. A dependability benchmark for OLTP application environments. In: Proc. of the VLDB. 2003. 742−753.

[59] Colliat G. OLAP, relational, and multidimensional database systems. SIGMOD Record, 1996,25(3):64−69.

[60] Varga J, Etcheverry L, Vaisman AA, Romero O, Pedersen TB, Thomsen C. QB2OLAP: Enabling OLAP on statistical linked open data. In: Proc. of the ICDE 2016. 2016. 1346−1349.

[61] Chen Y, Rau-Chaplin A, Dehne FKHA, Eavis T, Green D, Sithirasenan E. cgmOLAP: Efficient parallel generation and querying of terabyte size ROLAP data cubes. In: Proc. of the ICDE 2006. 2006. 164.

[62] Norman M, Wolfgang L, Shahul HP, Nitesh M, Carsten M, Sudipto C, Anil KG. SAP HANA—From relational OLAP database to big data infrastructure. In: Proc. of the EDBT 2015. 2015. 581−592.

[63] Golfarelli M, Graziani S, Rizzi S. Shrink: An OLAP operation for balancing precision and size of pivot tables. Data & Knowledge Engineering, 2014,93:19−41.

[64] Hasan KMA, Tsuji T, Higuchi K. An efficient implementation for MOLAP Basic data structure and its evaluation. In: Proc. of the DASFAA 2007. 2007. 288−299.

[65] Salka C. Ending the MOLAP/ROLAP debate: Usage based aggregation and flexible HOLAP (abstract). In: Proc. of the ICDE'98. 1998. 180.

[66] Antoniu G, Bougé L, Hatcher PJ, MacBeth M, McGuigan K, Namyst R. The hyperion system: Compiling multithreaded Java bytecode for distributed execution. Parallel Computing, 2001,27(10):1279−1297.

[67] Im DH, Cho CH, Jung IIG. Detecting a large number of objects in real-time using apache storm. In: Proc. of the ICTC 2014. 2014. 836−838.

[68] Yang MS, Ma RTB. Smooth task migration in apache storm. In: Proc. of the SIGMOD Conf. 2015. 2067−2068.

[69] Schaefer C, Manoj PM. Enabling privacy mechanisms in apache storm. In: Proc. of the BigData Congress. 2015. 102−109.

[70] Chen M, Mao S, Liu Y. Big data: A survey. Mobile Networks and Applications, 2014,19(2):171−209.

[71] Shen DR, Yu G, Wang XT, Nie TZ, Kou Y. Survey on NoSQL for management of big data. Ruan Jian Xue Bao/Journal of Software, 2013,24(8):1786−1803 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4416.htm [doi: 10.3724/SP.J.1001.2013.04416]

[72] Han J, Haihong E, Le G, *et al*. Survey on NoSQL database. In: Proc. of the 6th Int'l Conf. on Pervasive Computing and Applications (ICPCA). IEEE, 2011. 363−366.

[73] Gessert F, Wingerath W, Friedrich S, *et al*. NoSQL database systems: A survey and decision guidance. Computer Science-Research and Development, 2017,32(3-4):353−365.

[74] Memcached. https://github.com/memcached/memcached

[75] Redis. https://redis.io/

[76] LevelDB. https://github.com/google/leveldb

[77] MongDB. https://www.mongodb.com/

[78] CouchDB. http://couchdb.apache.org/

[79] Chang F, Dean J, Ghemawat S, *et al*. Bigtable: A distributed storage system for structured data. ACM Trans. on Computer Systems (TOCS), 2008,26(2):4.

[80] HBase. https://hbase.apache.org/

[81] Cassandra. http://cassandra.apache.org/

[82] Gurevich Y. Comparative survey of NoSQL and NewSQL DB systems. The Open University, 2015. https://www.openu.ac.il/lists/mediaserver_documents/academic/cs/ComparativeSurvey.pdf

[83] Grolinger K, Higashino WA, Tiwari A, *et al*. Data management in cloud environments: NoSQL and NewSQL data stores. Journal of Cloud Computing: Advances, Systems and Applications, 2013,2(1):22.

[84] Kaur K, Sachdeva M. Performance evaluation of NewSQL databases. In: Proc. of the 2017 Int'l Conf. on Inventive Systems and Control (ICISC). IEEE, 2017. 1−5.

[85] Moniruzzaman ABM. NewSQL: Towards next-generation scalable RDBMS for online transaction processing (OLTP) for big data management. arXiv Preprint arXiv:1411.7343, 2014.

[86] Kallman R, Kimura H, Natkins J, *et al*. H-Store: A high-performance, distributed main memory transaction processing system. Proc. of the VLDB Endowment, 2008,1(2):1496−1499.

[87] Cetintemel U, Du J, Kraska T, *et al*. S-Store: A streaming NewSQL system for big velocity applications. Proc. of the VLDB Endowment, 2014,7(13):1633−1636.

[88] Bacon DF, Bales N, Bruno N, *et al*. Spanner: Becoming a SQL system. In: Proc. of the 2017 ACM Int'l Conf. on Management of Data. ACM Press, 2017. 331−343.

[89] VoltDB. https://www.voltdb.com/

[90] DB-Engines. https://db-engines.com/en/ranking

附中文参考文献:

[1] 孟小峰,周龙骧,王珊.数据库技术发展趋势.软件学报,2004,15(12):1822−1836. http://www.jos.org.cn/1000-9825/20041208.htm

[11] 李建义.数据库原理及应用系统开发.北京:中国水利水电出版社,2005.20−22.

[17] 王静,孟小峰.半结构化数据的模式研究综述.计算机科学,2001,28(2):6−10.

[40] 张宏毅,王立威,陈瑜希.概率图模型研究进展综述.软件学报,2013,24(11):2476−2497. http://www.jos.org.cn/1000-9825/4486.htm [doi: 10.3724/SP.J.1001.2013.04486]

[71] 申德荣,于戈,王习特,聂铁铮,寇月.支持大数据管理的 NoSQL 系统研究综述.软件学报,2013,24(8):1786−1803. http://www.jos.org.cn/1000-9825/4416.htm [doi: 10.3724/SP.J.1001.2013.04416]

信俊昌(1977－),男,辽宁辽阳人,博士,教授,博士生导师,CCF 专业会员,主要研究领域为大数据管理与分析,感知数据管理,计算机辅助诊断,医学信息学.

王国仁(1966－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库,大数据管理与分析,生物信息学.

李国徽(1973－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为现代数据工程,实时系统.

高云君(1977－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库,大数据管理与分析,DB 驱动的 AI 技术.

张志强(1973－),男,博士,教授,CCF 高级会员,主要研究领域为智能信息处理,大数据分析处理,机器学习.