4 实 验

本节通过在模拟数据上的实验来展示所提出数据修复算法的性能,主要以运行时间作为度量.除此之外,我们还将展示其他因素对算法运行的影响.

4.1 实验设置

数据描述.为了简便起见,给定关系的所有属性均是整数型类型,且各属性上的数据都遵循均匀分布.在数据生成过程中,我们引入参数 S 来表示每一个属性中每个数据的副本数.例如:假定要生成一个包含 1 000 条元组的关系,且该关系含 10 个属性,则 r=1000,c=10.若设置 S=10,对于其中的每一列中的每一个单元格,我们在(0,r/S)范围内随机生成一个整数(在这里,范围就是(0,100)),则最后在该列中每一个数值大概有 S=10 个副本.

函数依赖.函数依赖关系随机生成.本实验生成形如 $A\rightarrow B$ 的函数依赖,其中,A 和 B 分别为数据库中的两个不相同的属性.同时,为了避免函数依赖之间的矛盾,假设属性间存在排序关系,我们约定所生成的函数依赖左边的属性都要大于(小于)右边的属性.

条件约束.对于给定的随机生成的数据库实例,我们随机生成本文中的 4 种条件约束.同时保证这 4 种约束之间相互不会发生矛盾.例如,如果有等值约束 EC(a,b),就不可能有非等值约束 NE(a,b).

本文的所有实验代码均使用 Java JDK 1.6 编写,运行在 Windows 7 操作系统之下.实验机器为配置主频是 1.60GHz 的 Intel Core i5 处理器,内存为 4GB 的 Thinkpad x240.

4.2 实验结果

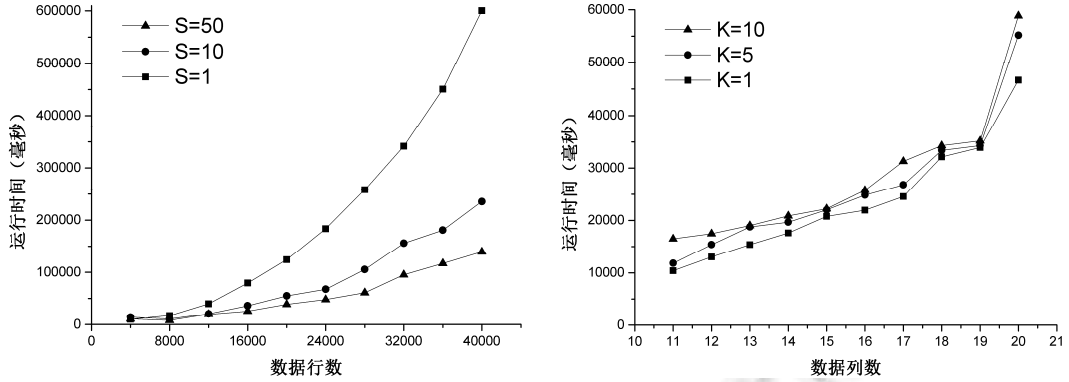
我们从 3 个方面来考察所提出算法的性能,包括总体运行时间、初始化时间以及算法修复过程中发生改变的单元格个数.最后我们考察算法的随机性.

4.2.1 算法总体运行时间

由第 3.5 节可知,本文所提算法的时间复杂度为 $O(p\times k+n\times m\times k)$,其中,$p$ 为外部知识库中的单元格个数,k 为函数依赖个数,n 为数据库中单元格的总数,m 为算法最后形成的等价类的个数.显然,算法的时间效率与多个因素有关,为了充分显示这些因素对算法总体运行时间的影响,我们通过改变数据量、函数依赖个数并控制算法最后生成的等价类来测试算法的运行效率.其中,我们通过改变数据库的行列数来变化总的数据量.而通过改变参数 S,即数据的副本数来达到控制最后生成的等价类个数的目的.显然,S 越大,最后生成的等价类个数越少.反之,最后生成的等价类个数越多.此外,我们还会变化算法中函数依赖的个数 K.由于函数依赖与数据库的列数相关,而副本数与数据库的行数相关.因此,实验中我们将行数与副本数、列数与函数依赖个数一起考量.

图 4 展示了本文所提算法的总体运行时间.其中,通过固定数据列 r=11,函数依赖个数 K=10,我们得到图 4(a).图 4(a)表示不同的数据副本数下,算法的运行时间与元组数量的关系.可以看出,随着元组数量的不断增加,总体运行时间也相应延长.这是因为,数据行数的增加使得单元格个数越来越多,执行增量生成等价类操作也越来越频繁,导致运行时间变长.此外,随着数据副本数的增加,运行时间反而会缩短.这是因为,在数据量固定的情况下,数据副本数越高,则其中包含的等价类个数就会越少.最坏情况下,如果数据副本数为 1,即一列数据中的每一个单元格的值都基本不相等,这时等价类的个数最多为数据的行数.随着等价类个数的增多,等价类的查找就会越来越耗时,最后使得运行时间也相应地变长.当 S=1 时,算法的复杂度变成了 $O(n^2)$(n 为数据量),所以其时间增长趋势呈二次曲线状.另外,图中值得注意的一点是,当数据行数较少时,运行时间会出现微小波动.这是因为,算法本身是一种随机算法,每次都是从数据库中随机取出数据.

通过固定数据行数 r=10000,副本数 S=10,我们得到图 4(b).图 4(b)显示在不同函数依赖个数的条件下,算法的运行时间随着数据列数的变化情况.同样地,随着数据列数的增加,使得单元格个数增多,导致运行时间变长.此外,随着函数依赖个数的增加,算法的运行时间也会相应变长.这是由于,函数依赖个数越多,增量构建等价类时检查的列数也会增多,导致时间变长.显然,函数依赖对运行时间的影响小于副本数对时间的影响.这是因为,当插入新单元格时需要检查同一列中所有等价类来判断它们的值是否相等.而当单元格加入到相应等价类之后,只需要检查当前单元格的等价类所对应列上的相关等价类之间的函数依赖,而不用检查列上其他的等价类.



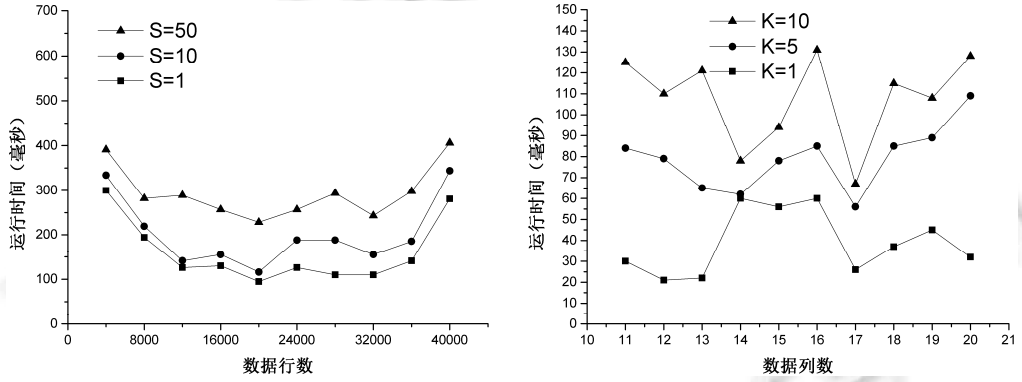
(a) 不同行数和副本数 (b) 不同列数和函数依赖个数

Fig.4 The overall running time

图 4 算法总体运行时间

4.2.2 初始化运行时间

通过固定 4 种条件约束的个数,我们同样考虑行数与副本数、列数与函数依赖个数对初始化的影响.固定数据列 r=11,函数依赖个数 K=10 得到图 5(a).图 5(a)展示在不同副本数的条件下,初始化阶段的运行时间随数据行数的变化情况.由于条件约束的个数固定,所以不同的数据量对初始化的时间影响不大,而且初始化的时间本身较短,所以时间统计的随机性相对较强.另一方面,随着副本数的增加,初始化的运行时间同样也有所延长.



(a) 不同行数和副本数 (b) 不同列数和函数依赖个数

Fig.5 The running time during the initialization phase

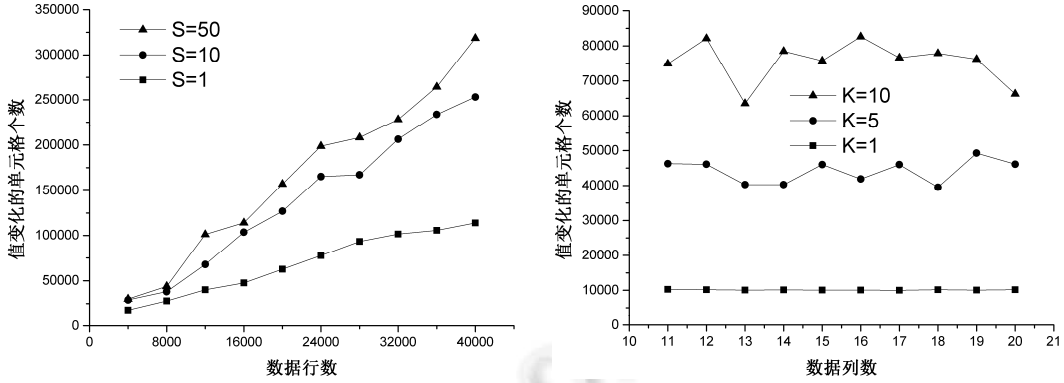
图 5 初始化运行时间

固定数据行 r=10000,副本数 S=10 得到图 5(b).图 5(b)显示在不同的函数依赖条件下,初始化阶段的运行时间随数据列数的变化情况.同样地,由于条件约束个数固定,初始化运行时间表现与列数无关.但是随着函数依赖个数的增加,初始化中函数依赖检查的消耗会增多,从而导致运行时间变长.

4.2.3 改变单元格的个数

改变单元格的个数能从一定程度上反映数据修复的质量,如果改变的单元格越少,则修复后的数据库与元数据库越接近,其质量越高.反之,如果改变的单元格越多,则修复后的数据库与原来的数据库偏差越大,所保留的原数据越少,其质量也越低.我们通过数据量、副本数和函数依赖个数的改变来观察改变单元格的个数.固定数据列 r=11,函数依赖个数 K=10 得到图 6(a).图 6(a)反映在不同的副本数的条件下,改变单元格的个数随数据行数的变化情况.可以看出,随着数据行数的增加,单元格违反函数依赖的可能性增加,需要改变的单元格个数也随之增多.此外,副本数越多,改变的单元格个数也越多.这是因为,副本数越多,值相等的单元格也越多,加入等价类后导致的函数依赖检查也越频繁,违反函数依赖的可能性越大,使得改变的单元格个数也越多.

固定数据行 r=10000,副本数 S=10 得到图 6(b).图 6(b)显示在不同的函数依赖个数的情况下,改变单元格的个数随数据列数的变化情况.可以看出,增加列数并不会明显地改变发生变化的单元格的个数,这是由于,在函数依赖的个数固定的情况下,增加列数并不会导致更多的函数依赖检查,使得改变单元格的个数没有明显的相关变化.然而,随着函数依赖个数的增长,改变单元格的个数会随之变多.这是因为,函数依赖越多,违反函数依赖的可能性越大,发生改变的单元格越多.



(a) 不同元组个数和副本数 (b) 不同列数和函数依赖个数

Fig.6 The number of cells changed

图 6 改变单元格的个数

4.2.4 算法的随机性

由于本文提出的算法是一种随机算法,所以原始数据库中的单元格进行修复过程的顺序不同,需要改变的单元格的个数不确定,产生的修复数据库也不相同,同时算法运行过程中的消耗也会有所差异.为了展示算法的随机性,我们利用同一个随机生成的数据库(行数 r=10000,列数 c=11),同样的条件约束和函数依赖(K=10),在副本数 S=10 的条件下独立地运行 10 次算法得到表 1.可以看出,每次修复的结果都有差异,因为改变单元格的个数都不一样,算法的总体运行时间也有所差异.但是算法总体稳定,时间消耗大约在 11s 左右,改变单元格的个数大约为 70 900.因此上述的性能实验比较是有意义的.另一方面,与第 4.2.2 节类似,由于初始化阶段的用时极短,虽然过程不存在随机性,但是消耗波动性很大,容易受到计算环境的影响.

Table 1 The running time in random

表 1 算法随机运行情况

更改单元格的个数	初始化时间(ms)	总时间(ms)
70 995	343	11 662
70 994	266	11 066
70 898	243	11 141
71 007	158	11 400
70 875	203	11 116
70 927	157	10 867
71 074	141	10 974
71 044	141	11 174
70 954	170	11 112
70 942	157	11 094

5 小 结

数据修复是数据管理领域的重要研究课题,其目的在于提升数据的质量.现有工作主要是采用函数依赖集合进行验证.但是,函数依赖集合仅仅能够部分表达数据质量问题,还需要考虑其他约束条件才能更准确地表达语意.本文提出了多种外部约束条件,包括数量约束、等值约束和非等值约束等.要求修复策略不仅要考虑到函数依赖集合,还应该同时符合这些约束条件.在修复策略目标上,本文采用了基数集合最小化目标,以平衡“最少修复”和“必要修复”二者之间的关系.

外在的约束条件还有更多.因此,在未来我们将沿着两条路线进行探索.首先,我们拟考虑更多的约束定义,或者是约束定义的复杂表达式.其次,我们拟从执行性能上作优化,进一步提升执行效率.

References:

[1] President's Council of Advisors on Science and Technology. Designing a digital future: Federally funded research and development in networking and information technology. 2010. http://www.whitehouse.gov/sites/default/files/microsites/ostp/pcast-nitrd-report-2010.pdf

[2] Big data: Science in the peta-byte era. Nature, 2008,455:1−136. http://www.nature.com/nature/journal/v455/n7209/edsumm/e080904-01.html [doi: 10.1038/455001a]

[3] Dealing with the data. Science, 2011,331(6018):639−806. http://www.sciencemag.org/site/special/data/

[4] Gong XQ, Jin CQ, Wang XL, Zhang R, Zhou AY. Data-Intensive science and engineering: Requirements and challenges. Chinese Journal of Computers, 2012,35(8):1−16 (in Chinese with English abstract).

[5] Ao L, Shu JW, Li MQ. Data deduplication techniques. Ruan Jian Xue Bao/Journal of Software, 2010,21(5):916−929 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3761.htm [doi: 10.3724/SP.J.1001.2010.03761]

[6] Jeffery S, Garofalakis M, Franklin M. Adaptive cleaning for RFID data streams. In: Proc. of the VLDB. Seoul: VLDB Endowment, 2006. 163−174.

[7] Redman TC. The impact of poor data quality on the typical enterprise. Communications of the ACM, 1998,41(2):79−82.

[8] Bohannon P, Flaster M, Fan WF, Rastogi R. A cost-based model and effective heuristic for repairing constraints by value modification. In: Proc. of the SIGMOD. New York: ACM Press, 2005. 143−154. [doi: 10.1145/1066157.1066175]

[9] Kolahi S, Lakshmanan LVS. On approximating optimum repairs for functional dependency violations. In: Proc. of the ICDT. New York: ACM Press, 2009. 53−62. [doi: 10.1145/1514894.1514901]

[10] Arenas M, Bertossi LE, Chomicki J. Consistent query answers in inconsistent databases. In: Proc. of the PODS. New York: ACM, 1999. 68−79. [doi: 10.1145/303976.303983]

[11] Lopatenko A, Bertossi LE. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In: Proc. of the ICDT. Barcelona: Springer-Verlag, 2007. 179−193. [doi: 10.1007/11965893_13]

[12] Bekales G, Ilyas IF, Golab L. Sampling the repairs of functional dependency violations under hard constraints. In: Proc. of the VLDB. Singapore: VLDB Endowment, 2010,3(1):197−207. [doi: 10.14778/1920841.1920870]

[13] He C, Tan ZJ, Chen Q, Sha CF, Wang ZH, Wang W. Repair diversification for functional dependency violation. In: Proc. of the DASFAA. Heidelberg: Springer-Verlag, 2014. 468−482. [doi: 10.1007/978-3-319-05813-9_31]

[14] Chen Q, Tan ZJ, He C, Sha CF, Wang W. Repairing functional dependency violations in distributed data. In: Proc. of the DASFAA. Heidelberg: Springer-Verlag, 2015. 441−457. [doi: 10.1007/978-3-319-18120-2_26]

[15] Li J, Liu X. An important aspect of big data: Data usability. Journal of Computer Research and Development, 2013, 50(6):1147−1162 (in Chinese with English abstract).

[16] Bleiholder J, Szott S, Herschel M, Kaufer F, Naumann F. Subsumption and complementation as data fusion operators. In: Proc. of the EDBT. New York: ACM Press, 2010. 513−524. [doi: 10.1145/1739041.1739103]

[17] Liu HP, Jin CQ, Zhou AY. A pattern-based entity resolution algorithm. Chinese Journal of Computers, 2015,38(9):1796−1808 (in Chinese with English abstract).

[18] Xie JY, Yang J, Chen YG, Wang HX, Yu PS. A sampling-based approach to information recovery. In: Proc. of the ICDE. Iscataway, NJ: IEEE Computer Society, 2008. 476−485. [doi: 10.1109/ICDE.2008.4497456]

[19] Chen HQ, Ku WS, Wang HX, Sun MT. Leveraging spatio-temporal redundancy for RFID data cleansing. In: Proc. of the SIGMOD. New York: ACM Press, 2010. 51−62. [doi: 10.1145/1807167.1807176]

[20] Zhuang YZ, Chen L. In-Network outlier cleaning for data collection in sensor networks. In: Proc. of the CleanDB. New York: VLDB Endowment, 2006. 41−48.

[21] Chiang F, Miller RJ. A unified model for data and constraint repair. In: Proc. of the ICDE. Iscataway, NJ: IEEE Computer Society, 2011. [doi: 10.1109/ICDE.2011.5767833]

[22] Silberschatz A, Korth H, Sudarshan S. Database System Concepts. 6th ed., McGraw-Hill Education, 2010.

附中文参考文献:

[4] 宫学庆,金澈清,王晓玲,张蓉,周傲英.数据密集型科学与工程:需求和挑战.计算机学报,2012,35(8):1−16.

[5] 敖莉,舒继武,李明强.重复数据删除技术.软件学报,2010,21(5):916−929. http://www.jos.org.cn/1000-9825/3761.htm [doi: 10.3724/SP.J.1001.2010.03761]

[15] 李建中,刘显敏.大数据的一个重要方面:数据可用性.计算机研究与发展,2013,50(6):1147−1162.

[17] 刘辉平,金澈清,周傲英.一种基于模式的实体解析算法.计算机学报,2015,38(9):1796−1808.

金澈清(1977－),男,浙江文成人,博士,教授,博士生导师,CCF 会员,主要研究领域为海量数据管理,包括基于位置的服务,数据质量,不确定数据管理.

刘辉平(1990－),男,博士生,主要研究领域为基于位置的服务,数据挖掘,实体解析.

周傲英(1965－),男,博士,教授,博士生导师,CCF杰出会员,主要研究领域为Web数据管理,数据密集型计算,内存集群计算,大数据基准测试和性能优化.