数据库反规范化设计的探讨与实现

The Discussion and Realization of Denormalization Design of Database

邓小善　罗大庸　（中南大学信息科学与工程学院　湖南长沙 410000）

摘要：本文通过比较规范化与反规范化设计的特点与优劣，结合实例分析了反规范化设计的几个指导原则，阐明了反规范化设计的实现方法。

关键词：规范化 反规范化 指导原则 实现方法

1 引言

数据库设计者的一个重要职责就是让数据库达到人们的需求与最佳性能，而数据库的逻辑设计直接影响着数据库的性能。针对数据库的逻辑设计，存在着两种不同的观点：一是实现高规范化（normalization）的数据库设计，这种设计方法目前占据着主导地位，它常能达到数据存储空间的最佳利用和存储数据较少响应时间的较好平衡。二是实现智能的非规范化（denormalization）设计。近年来，随着数据库表中的数据量由 MB 级向 GB 甚至 TB 级的急速膨胀，数据响应速度变得越来越慢，一个完全规范化的设计有时会影响数据库的运行效率，而合理地使用反规范化设计，可较好地优化数据库性能。因而，非规范化的关系模式设计引起了业界的关注并得到了逐步的应用。本文将结合按规范化设计的员工表（Employee）、员工工资表（Employeepay）两个数据库表实例（图 1），来阐述如何进行数据库的反规范化设计。

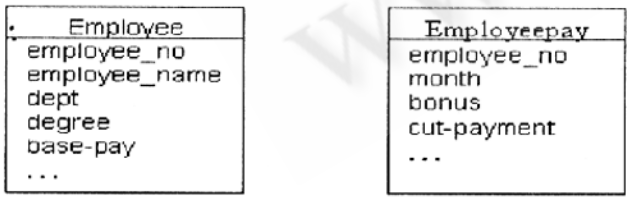

图 1　员工表（Employee）与员工工资表（Employeepay）

其中 employee_no、employee_name、dept、degree、base－pay、month、bonus、cut－payment 分别指员工编号、员工姓名、所在部门、学历、基本工资、月份、奖金、扣款。

2 规范化与反规范化设计的比较

规范化通过对表结构的一系列测试来决定它是否满足或符合给定范式。目前遵循的主要范式包括 1NF、2 NF、3 NF、BCNF、4NF、5NF 等 6 种，在工程中 3NF、BCNF 应用得最广泛，我们一般采用 3NF 作为标准。范式的等级越高，需满足的约束集条件也越严格。

1NF：关系中的所有属性都是单一的，即组和多值的列不能重复。

2NF：一个关系如果是 1NF 的，且每一非主键元素表现完全依赖于主键，而不是依赖于主键中的一部分，则该关系 2NF 的。

3NF：一个关系如果是 2NF 的，且每一非主键列都不传递依赖于主键，则该关系是 3NF 的。

反规范化设计通过合理增加冗余数据、数据分片等策略来改进原有的规范化关系模式，以达到改善数据库的性能的目的。规范化与反规范化设计的优缺点比较如表 1。

从表 1 可以看出，规范化设计有效地控制了数据冗余，数据更新的速度较快，但常因查询时的多表连接（join）而导致查询性能降低。而反规范化设计的主要优势在于可以有效减少查询时的多表连接，提升查询的效率，但会较大地影响更新的速度，需要维护数据库中数据的完整性。

3 反规范化设计的指导原则

目前，在指导关系模式的设计中，我们倡导进行规

范化设计，且通常 3NF 在性能、扩展性和数据完整性方面达到了较好平衡。反规范化在一定程度上带来性能优化的同时，在系统完整性维护上存在着较大的困难。因此，我们要认真分析反规范化的利弊，在进行反规范化设计时务须谨慎，可考虑以下几条指导原则：

3.1 确信自己对系统逻辑设计有全面的理解

表 1 规范化与反规范化设计的特点与优劣比较

规范化(3NF)	反规范化
表的数据列较少，表的数量较多	表的数量可能会适当减少
数据冗余少，节约了磁盘空间	增加了冗余数据
数据的完整性维护比较简单	难以保证数据的完整性
加快更新的速度，可能影响查询速度	加快了查询速度，较大地影响更新的速度
查询时需要更多的多表连接	减少了对连接运算的依赖
有利于索引的创建和更快的排序	排序与索引的效率受到适当影响
简化了大型数据库的组织规划	加大了大型数据库的组织规划的复杂度

这些知识将帮助自己确定当改变了部分系统时，其它部分所受到的影响是什么。

3.2 尽量采用索引、数据存储技术等方法来提升数据库性能

用户在进行数据库设计时，明确是否一定需要反规范化？通过规范化设计的该数据库是否一定会遇到较大的性能问题？同时，还要确定更新的频率，如果更新频繁，那么完整性维护所化的代价将抵消掉冗余数据所带来的性能提高。若该性能问题能够通过优化查询、良好的索引设计或通过 RAID、SQL 文件组等数据存储技术等方法得到解决时要尽量避免采用反规范方法。如对于数据量不太大的工资管理系统，建议通过良好的索引技术来解决性能问题将更令人放心。

3.3 先规范化再反规范化

不要试图立刻反规范化设计整个数据库，较佳的策略是以规范化的设计为出发点，然后再根据需要，尽量少范围地、谨慎地进行反规范化，并明确其带来的一些完整性维护的内容，力图以最少代价达到最大程度地提升系统性能的目的。

3.4 注重完整性维护

数据库的完整性是指数据的正确性和相容性。使用反规范技术所带来的最大问题是需要维护数据库的完整性，我们常采用应用事务逻辑维护、批处理维护、触发器维护等方法来保证数据库的完整性。

(1) 应用事务逻辑维护。该方法要求在同一事务中对所有涉及的表同步进行增、删、改操作。由于不同的事务所对应的相关表可能不同，而同一应用事务必须在所有的相关表中进行相应操作，容易造成遗漏，特别是在需求变化时更容易出错，因而采用应用事务逻辑来维护数据库的完整性风险较大。如 增、删、改“Employee”表中一数据行时，在“Employeepay”表中也需增、删、改相应的数据行。

(2) 批处理维护。批处理维护是指对于实时性要求不高的数据库，对冗余列或者派生列的修改积累一定的时间后，运行一个批处理作业或存储过程对冗余或者派生列进行修改以达到维护数据库的完整性的目的。对于“Employee”、“Employeepay”两表，由于数据的更改有一定的时段性，故可采取在每月初或月末由管理员分阶段运行批处理作业，进行数据库的批处理维护。

(3) 触发器维护。触发器是维护数据库完整性最强大的工具，通过创建触发器，对数据的任何修改可立即触发对冗余列或者派生列的相应修改，可使最终用户和程序员不受冗余数据的影响。触发器是实时的，也易于维护。例如在修改“Employee”表中“Employee_no”的值时，利用触发器可以同时修改“Employeepay”表中相应的“Employee_no”及其它相关属性列的值。

4 反规范化设计的方法

反规范化设计的方法主要有增加冗余列、增加派生列、重新组表、数据分片等。

4.1 增加冗余列

增加冗余列是指在数据表中增加因经常访问而需频繁连接的其它表中的某些属性列，以避免表之间的连接过于频繁。这将导致多个表中具有相同的列，即冗余列。增加冗余列常常应用于以下场合：当两个或多个表在查询时经常需要进行重复的连接时，特别是规范化设计产生了许多 4 路或更多路合并关系，就可以考虑在数据库表中加入重复列。增加冗余列可以在查询时避免连接操作，但它需要更多的磁盘空间，同时将增加表维护的工作量。如在“employeepay”表中，因需经常按姓名查询员工的工资而要连接“employ-

ee"与"employeepay"表,故可在"employeepay"表中增加冗余列"employee_name"。如图 2 所示:

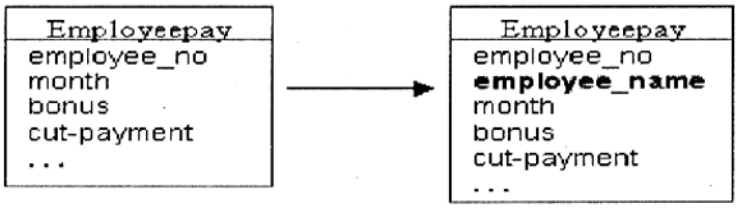

图 2　增加冗余列

4.2 增加派生列

增加派生列是指在查询某数据库表时,需频繁访问来自于其它表中的数据的计算结果,则可将该结果作为表中增加的派生列。它的作用是在查询时减少连接操作,避免使用集函数。增加派生列常应用于以下场合:常用的计算字段(如实发工资等)可以考虑存储到数据库表中。如"employeepay"表即可增加应发工资(total_pay)、所得税(income_tax)实发工资(factual_pay)三项。如图 3 所示。

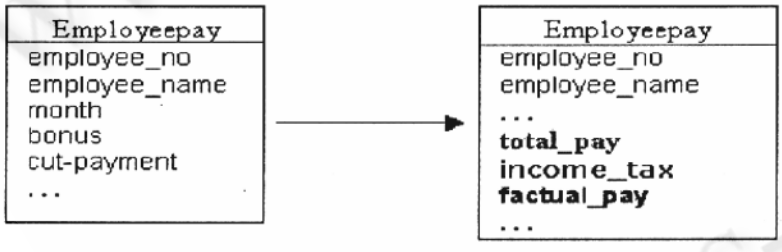

图 3　增加派生列

4.3 重新组表

重新组表是指将两个或多个表的某些数据列重新组成一个新表。重新组表常应用于以下场合:主要的应用程序在执行时需要经常将多个表连接起来进行查询,如经常使用两个或多个表中的某几列时,我们可把这些列重新组成一个表来减少连接,从而提高查询性能。重新组表需要更多的磁盘空间,同时也损失了数据在概念上的独立性。由于要经常发放工资表,可将上述两表中的某些属性重组成"pay – table"表,以避免经常性、重复性的多表连接。如下图 4 所示。

4.4 数据分片

数据分片主要有三种方式:水平分片、垂直分片与混合分片。

(1) 水平分片。水平分片是指按一定的条件把数据库的所有行划分成若干不相交的子集,即多个表。在数据库表比较庞大的情况下,水平分片方式通常在以下的几种情况下使用:一是行数据的访问频率差别很大时,我们常将频繁被访问的行数据放在一个表中,将较少被访问的行数据如历史数据放在另一个表中。如可将"employeepay"表中当月的工资数据与其它月的数据进行水平分片;二是若数据表中的数据本来就有独立性,如可将"employeepay"表根据"dept(部门)"属性将行数据水平分片为多个表;三是在数据量非常庞大,需要把数据存放到多个介质上时也可进行水平分片。如我们可将本年度的数据与其它年度(即历史数据)的数据进行水平分片,组成两个表。

(2) 垂直分片。把一个数据库的属性集分成若干子集,并在这些子集上作投影运算,每个投影称为垂直分片。我们常把主键和一些频繁访问的属性列放到一个表中,把主键和其它的属性列放到另一个表中。这有利于并行处理,并将产生列数较少的表,一个数据页能够存放更多的数据,同时,把频繁被访问的列数据同较少被访问的列数据分开,在查询时将会减少 I/O 次数。其缺点是需要管理冗余列,查询所有数据时需要多表连接(join)操作。如在"employee"表中还有一些其它属性列没有列出,如"sex(性别)"、"native_place(籍贯)"、"birthdate(出生日期)"等,当员工数达到几百万或以上时,可进行如下图 5 的垂直分片:

(3) 混合分片。在实际应用中仅仅进行单一的水平或垂直分片往往是不够的,常常同时用到这两种方法的综合,可以先水平分片再垂直分片,或先垂直分片再水平分片。

5 小结

规范化设计和反规范化设计都是数据库逻辑设计阶段中的重要方法,它们互有利弊。一方面,对于大数据量、查询操作频繁、更新操作较少的数据库表,如现在应用非常广泛的数据仓库,采用适当的反规范化设计常可提高其查询性能。另一方面,反规范技术也带

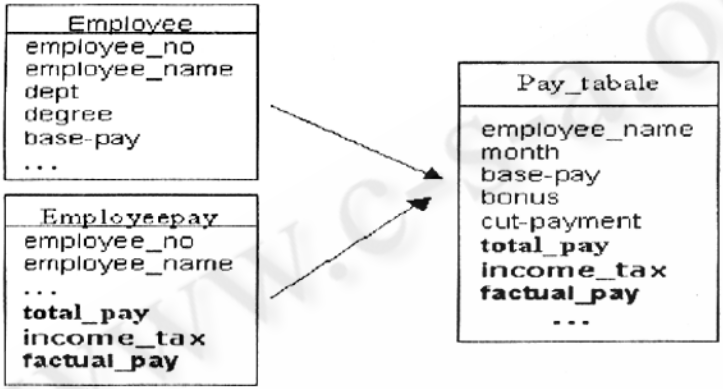

图 4　重新组表

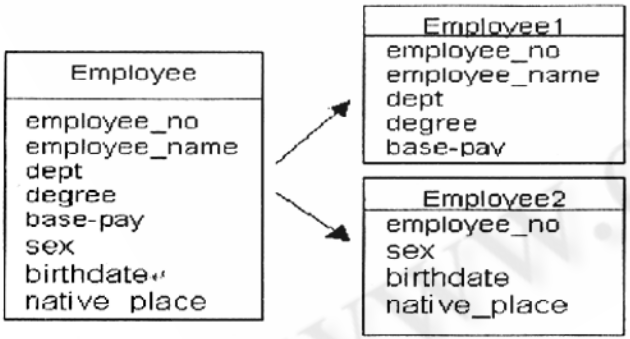

图 5　垂直分片

来了数据冗余、需要维护数据的完整性等问题。因此，我们在实际工作中，务须权衡利弊，谨慎使用反规范化设计，将两者合理结合、相互补充，充分发挥各自的优点。

参考文献

1 谷震离，关系数据库查询优化方法研究[J]，微计算机信息，2006.5.

2 尹萍，SQL Server 数据库性能优化[J]，计算机应用与软件，2005.3.

3 陶勇、丁维明，数据库中规范化与反规范化设计的比较与分析[J]，计算机技术与发展，2006.4.

4 尹永顺，MS SQL Server 中大数据量表的查询优化[J]，计算机系统应用，2005.2.

5 余金山、周敏龙等，SQL Server 2000 编程指南[M]，北京希望电子出版社，2001.6.

6 Thomas M. Connolly & Carolyn E. Begg 著，何玉洁译，数据库设计教程[M]，机械工业出版社，2005.1.