关系数据模型设计

李博杰 2011-12-03

声明式语言与过程式语言

数学家、物理学家、计算机学家分别拿到了一个球，要求测定球的体积

数学家测量直径，用球的体积公式计算；后来觉得球不够圆，用三重积分计算

物理学家将球放入装满水的烧杯中，测排水量

计算机学家从球生产商的数据库中找到了 datasheet

不同的编程模型，不同的思考方式

声明式语言与过程式语言

过程式编程语言：由流控制语句控制的一系列运算步骤
函数式编程语言：用一系列嵌套的函数调用（变换）解决问题
约束逻辑编程语言：在问题领域中添加约束，直到确定答案
SQL 是声明式语言，丢弃了算法的概念，只提供对问题的说明（做什么而非怎样做）
担心效率降低：如同担心高级语言比汇编语言效率低。现代 DBMS 已经优化得很好了。

关系模型不同于传统存储

列不是字段，没有顺序
行不是记录，表中的行是无序集合
语句不是过程，要使用约束条件而非计算选出需要的数据集合
数据完整性靠完整性约束而非应用程序实现
元数据（表结构）不是数据，不能任意修改

数据的原子性

分子型数据元素

(x,y) 表示坐标：应当分为两列

tag1,tag2,tag3：应当用关联表

拆散原子（将属性转换成元数据）

male_student 与 female_student 表：应当用属性存储性别，特定性别才有的属性用 NULL 标记

继承 linux 日志的传统，根据时间，每月一个表：应当用属性存储时间（除非表太大影响性能）

数据冗余

选课系统：课程、学生、时间、教室、教师
属性应当归属于唯一确定它的实体
关系（学生，学号，课程号，教室）不符合关系数据模型的范式，应当拆分为三个关系：

（学生，学号）

（学生，课程号）

（课程号，教室）：关联表

两个实体间多对多的关系：关联表

数据冗余

文章的标签

(article_id, content, tags), tags = “tag1,tag2,tag3”

(article_id, content, tag1, tag2, tag3)

(article_id, content), tag (article_id, tag)

第三种显然最合理

NULL

NULL 与任何值运算还是 NULL

NULL AND TRUE = NULL AND FALSE = NULL

NULL AND FALSE = FALSE

NULL OR TRUE = TRUE

(NULL = NULL) = (NULL + 1) = NULL

用 NULL（而不是 -1 等特殊值）来标志悬空值或非法值，尽管可能违背关系逻辑，不要轻易规定 NOT NULL

树形结构

邻接表： parent_id

无法遍历一棵树，当树的深度没有限制时

枚举路径

/usr/local/bin/bash ：类似文件系统的路径，通过比较字符串确定树中关系

效率低，字符串处理复杂，数据完整性无保证

关系数据库中尽量避免字符串处理，可能是数据非原子性的征兆

树形结构

祖先 - 后代表

Path (ancestor_id, descendant_id)

记录所有满足祖先 - 后代关系的元组

查询子树操作非常简单，祖先 - 后代表的数据冗余度不超过树的深度

允许一个结点存在多个祖先，允许森林

当在一个表内部需要维护复杂的关系时，不妨换用独立的关联表，善用 JOIN 操作。

可扩展的表结构

“实体 - 属性 - 值”模式

Attr (entity_id, attribute_name, value)

无法保证引用完整性

无法使用 SQL 数据类型（再加一列 datatype ？！）

难以直接取出一列，难于使用关系数据模型的连结等操作

一般需要借助应用层框架，提供非关系的接口

如果数据不需要复杂的关系处理，还不如用非关系数据库（一些特大网站就是这样干的）

可扩展的表结构

单表继承模式

所有属性都作为列，不需要的属性为 NULL

各类的独特属性很少时适用

每类一表模式

每个“类”定义一个单独的关系（表）

忽视了类之间的继承关系，不便于维护

关系模型的根本难题：缺乏灵活性

元数据是绝对静止的，数据是绝对运动的

不能与面向对象机制完全融合

可扩展的表结构

类表继承模式

子类所在的表只需保存与父类不同的部分

应用层与数据库之间仍然需要厚重的胶合层

妥协：半结构化模式

常用的属性保存在基类的表中

另一个（实体 - 属性 - 值）模式的表，用于存储各派生类的独特属性

在查询常用属性时便于使用关系操作

违反了关系理论，收获了可扩展性

格物网目前的数据存储模式

多态关联

引用完整性

一个实体可能引用自一个关系，也可能引用自另一个关系

一条评论可能引用自一个帖子，也可能引用自一篇文章

中介表：关联两边

post_comment (pid, cid), post (pid, content)

article_comment (aid, cid), article (aid, content)

comment (cid, content)

多态关联

本质问题：类的继承

post 和 article 都继承自 issue，凡是 issue 都可以有 comment。

使用上节所述“类表继承”模式，post 和 article（将来可能还有 blog，wiki）继承自 issue

issue 表采用统一的 id，便于维护引用完整性

post 表和 article 表只存放特别的信息

不要害怕关联查询！这是关系数据模型的生命所在。好数据结构上的关联查询比差数据结构上的若干简单查询更高效。

存储附件

直接在数据库中存储附件

数据库规模过大，不易于备份恢复

数据库适合关系查询，文件系统适合确定查询

在文件系统中存储附件，用数据库指向之

有一些数据置于数据完整性保护之外

文件不支持事务

较小的文件直接放进数据库，便于管理

较大的文件采用抽象层保持数据库与文件系统的一致性

数据库索引