The Pragmatic Programmers

TURING 图灵程序设计丛书

SQL Antipatterns

Avoiding the Pitfalls of Database Programming

SQL反模式

- 深入剖析数据库编程常见错误
- 提升SQL功力的实用宝典
- 大师指点令人茅塞顿开

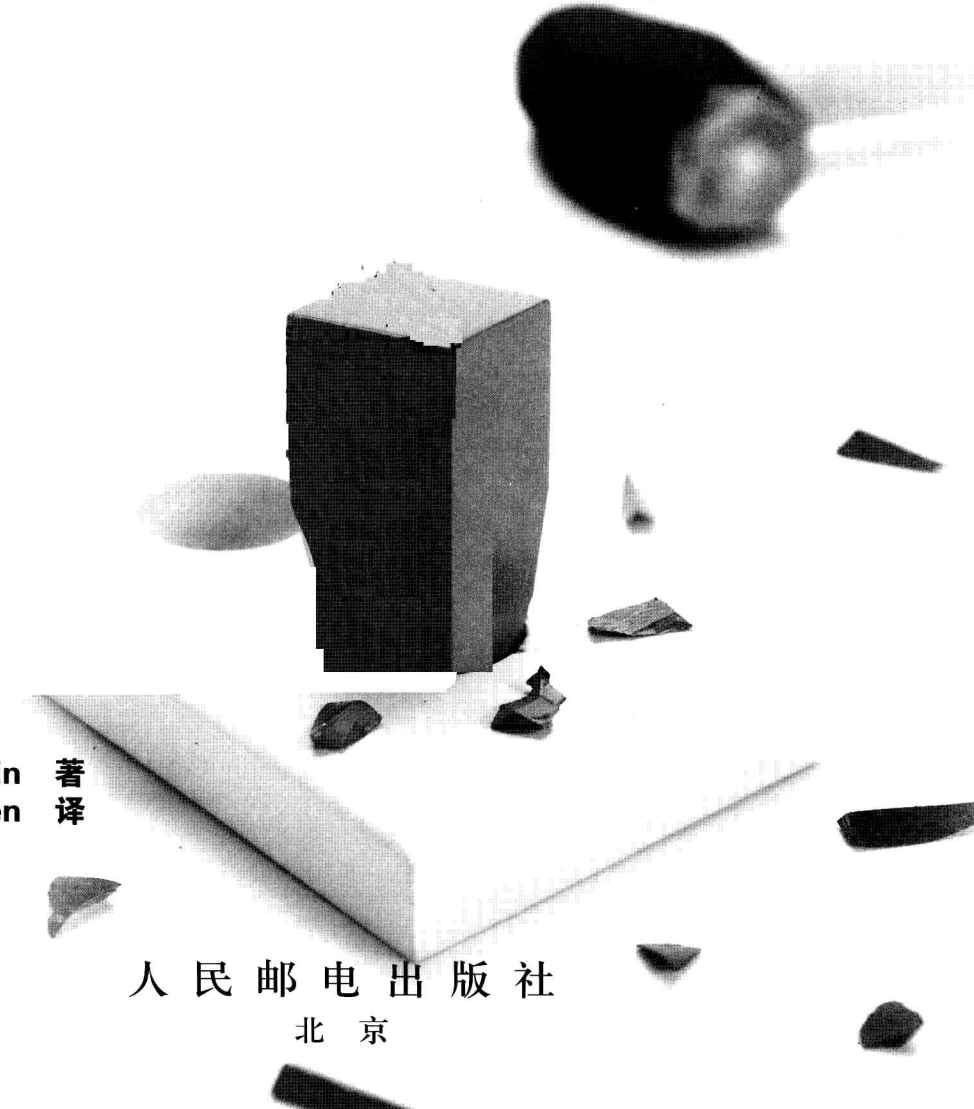
[美] Bill Karwin 著
谭振林 Push Chen 译

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

SQL Antipatterns
Avoiding the Pitfalls of Database Programming

SQL反模式



[美] Bill Karwin 著
谭振林 Push Chen 译

人民邮电出版社
北京

图书在版编目（CIP）数据

SQL反模式 / (美) 卡尔文 (Karwin, B.) 著 ; 谭振林译. -- 北京 : 人民邮电出版社, 2011.9
（图灵程序设计丛书）
书名原文: SQL Antipatterns:Avoiding the Pitfalls of Database Programming
ISBN 978-7-115-26127-4

Ⅰ. ①S… Ⅱ. ①卡… ②谭… Ⅲ. ①关系数据库－数据库管理系统，SQL Server Ⅳ. ①TP311.138

中国版本图书馆CIP数据核字(2011)第156320号

内 容 提 要

本书是一本广受好评的SQL图书。它介绍了如何避免在SQL的使用和开发中陷入一些常见却经常被忽略的误区。它通过讲述各种具体的案例，以及开发人员和使用人员在面对这些案例时经常采用的错误解决方案，来介绍如何识别、利用这些陷阱，以及面对问题时正确的解决手段。另外，本书还涉及了SQL的各级范式和针对它们的正确理解。

本书适合SQL数据库开发人员与管理人员阅读。

图灵程序设计丛书

SQL反模式

◆ 著 [美] Bill Karwin
译 谭振林 Push Chen
责任编辑 傅志红
执行编辑 李 盼

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：16.5
字数：355千字 2011年9月第1版
印数：1－3 000册 2011年9月北京第1次印刷
著作权合同登记号 图字：01-2010-7681号
ISBN 978-7-115-26127-4

定价：59.00元

读者服务热线：(010)51095186转604 印装质量热线：(010)67129223
反盗版热线：(010)67171154

读者感言

对于经常碰到本书中表述的那些数据库设计决择的软件开发人员来说，这本书是必读的。因为它将帮助开发团队理解数据库设计达成的结果，并且基于实际的需求、预期、测定做出最合理的决定。

——Darby Felton，DevBots Software Development 的联合创始人

我非常喜欢 Bill 在书中采用的写作方式，展示了他独一无二的风格和幽默感，这对讨论一大堆枯燥的话题太重要了。Bill 成功运用了一种很好的表述方式，让技术以易于理解的面貌示人，而且便于以后查阅。简而言之，这将是你书架里又一极好的资源。

——Arjen Lentz，Open Query
（http://openquery.com）执行总监，
High Performance MySQL 第二版作者之一

对于有 SQL 基础，但是在为项目设计 SQL 数据库时希望寻求基础之外帮助的软件工程师来说，这本书尤其有用。

——Liz Neely，资深数据库程序员

Bill 捕捉到了我们在 SQL 不同方面碰到过的很多陷阱的关键所在，而我们有些时候甚至没有意识到已被困住了。Bill 的反模式涵盖从“不敢相信我又犯了一遍”的事后感叹，到最佳方案与伴你一路走来的 SQL 教条相左的诡异情况。这是一本对 SQL 骨灰和新手都不错的书。

——Danny Thorpe，
Microsoft 总工程师；
Delphi Component Design 作者

译者序一

毫无疑问，数据库领域当下最热门的概念是 NoSQL，我正在公司最新大型社区项目中实践 NoSQL 产品，并准备在更大范围内推广优秀的 NoSQL 产品，而另一译者——陈魏明，则自己研发了一个 NoSQL 产品，应用在另一大型社区项目中，提供 Feed 系统的支持。

但是，正如 NoSQL 自身所宣扬的一样，任何一种 NoSQL 产品，甚至所有的 NoSQL 产品合在一起，它们的设计初衷绝不是解决掉所有的数据处理需求，它们追求的是为某一种或某几种数据处理场景选择最优的 CAP[①]组合，提供最合适的解决方案。

因此，SQL 并没有因为 NoSQL 的流行而变得不重要，它仍然跟以前一样重要，并且因为它们长期以来在开发人员中建立的深厚基础，以及丰富的支持工具，特别是强大的查询功能，将使其长期在广泛的数据处理场景中作为主要的解决方案而存在。比如你总不能用 Redis[②]来处理运营团队天天变着戏法要的运营分析数据吧。

这本语言略显啰嗦的书，是一本非常实用的书，因为它每一章的内容都源自于最常见、最普通的 SQL 应用场景，每一章中描述的问题，都是全世界的 SQL 应用人员犯得最多的错误。总之，我译完这本书后，就有一个强烈的感触："原来我犯了这么多错误！"

所以，这本书中的知识与教训，应该对很多人（比我资浅的那一小部人和比我资深的那一大部分人）都有帮助，而且长期有效。

本书作者知识渊博，原作中不少地方引经据典，我们在翻译过程中尽量通过 Google 等办法查找对应的典故，不过文化的差异和有限的水平，造成译稿中必定还有不少差强人意的地方，请各位读者原谅。

谭振林

2011 年 5 月

---

① CAP 是指：一致性（Consistency），可用性（Availability），分区容忍性（Partition tolerance）。CAP 原理认为这三个要素最多只能同时实现两点，不可能三者兼顾。

② Redis 是一款优秀的、基于内存处理数据的、原生支持多种数据结构的 Key-Value 型 NoSQL 产品。

译者序二

我本人并非DBA，但工作中时刻都要和数据打交道，无论是SQL还是NoSQL。在一个产品的生命周期里，设计与开发总是只占很小的一部分，大量的时间都用在后续的维护、优化和调整中。互联网服务更是如此。而维护、性能调优阶段，就是一个不断地犯错—纠正—归纳的循环。本书作者将他所归纳整理出的这些“错误”写在了这本书中，让我们这些后来者能够更早地发现程序中的问题，从而节省大量的时间成本。

无论时下NoSQL技术有多么的火爆，终究有一个无法解决的问题——内存开销。因而几乎所有的网站、社区，其后台必然有大量使用SQL进行存储的模块。

在翻译本书的过程中，我也回顾了以前所接触到的或者自己做的数据库设计，很多都没有从数据库本身出发去解决问题，而是直接在前端增加了一个缓存。

本书中的很多问题解决方案都很优雅，在不触碰程序员追求程序整洁度的洁癖神经的情况下，利用SQL本身就有的特性和功能解决了问题。这些案例非常值得我们学习。

本书作者知识丰富，在很多方面上都有所涉及，书中提到很多文化方面的内容，翻译不到位之处，敬请指正。

Push Chen

2011年5月

目　录

第 1 章 引 言

所谓专家，就是在一个很小的领域里把所有错误都犯过了的人。

▶ 尼尔斯·玻尔

我曾经拒绝过第一份关于 SQL 的工作。

获得加州大学计算机与信息科学的本科学位后不久，我接到了一个工作邀请，来自于一位曾经在加州大学工作过的经理，我们在一次校园活动相识。他当时刚刚建立了自己的软件公司，致力于使用 shell 脚本和诸如 awk 的相关工具（诸如 Ruby、Python、PHP，甚至 Perl 等现代动态语言，在当时都还不甚流行），来开发适用于众多 UNIX 平台的数据库管理系统。这个经理邀请我是因为他需要一个人来开发一些代码，识别并且执行功能有限版本的 SQL 语言。

他说："我不需要支持完整的 SQL 语言，那样工作量太大了。我只需要支持一个 SQL 语句：SELECT。"

我在学校里并没有学过 SQL。数据库在当时并不像现在这样普遍，并且当时也没有诸如 MySQL 和 PostgreSQL 之类的开源程序。但我用 shell 脚本开发过完整的应用程序，并且了解一些语法分析的技术，也做过一些关于编译器设计和计算语言学的课程项目。因此我在想是否要接受这个工作。只解析像 SQL 这样的专业语言中单独的一类语句会有多困难呢？

我找了一份 SQL 的参考资料并且立刻意识到它不同于那些支持 `if()`、`while()`语句、变量定义、表达式以及可能还有函数调用的语言。说 SELECT 只是此类语言中的一个语句，等同于说引擎只是汽车的一部分。从字面上来看，这两句话都是正确的，但是都完全掩盖了这两个主体的复杂性和深度。我意识到仅仅为了支持 SELECT 这一个语句的执行，就必须要实现一个完整的关系型数据库管理系统以及其查询引擎的全部代码。

我拒绝了这个用 shell 脚本实现 SQL 解析与关系型数据库管理系统引擎的工作机会。那个经理并没有充分理解他的项目的复杂度，可能他并不理解什么是关系型数据库管理系统（RDBMS）。

我早期使用 SQL 的经历和普通的软件开发人员并没有什么区别，和那些从计算机科学专业毕

业的学生相比也是如此。大多数人学习 SQL 语言都是因为项目所需而不得不自学的，而不是像其他的编程语言那样从头开始认真地学习。不论是对 SQL 爱好者、专业程序员或者拥有博士学位的娴熟的研究人员，SQL 就好像是程序员的一个不经过训练就学会了的软件技能。

此后，当我了解了一些 SQL 方面的知识后，我惊讶于它和那些过程式的编程语言（诸如 C、Pascal 和 shell）或是面向对象的编程语言（诸如 C++、Java、Ruby 或 Python）都有着显著的区别。SQL 是一门*声明式的编程语言*，就像 Lisp、Haskell 或者 XSLT。SQL 使用集合（set）作为根本的数据结构，而面向对象的语言使用的是对象（object）。受过传统培训的软件开发人员被所谓的“*阻抗失配*”[①]所阻碍，因此很多程序员转而使用现成的面向对象的库，以此来避免学习如何高效地使用 SQL。

自 1992 年以来，我在工作中大量使用 SQL。我在开发应用程序的过程中要使用它，我也为 InterBase RDBMS 产品提供技术支持、开发培训课程以及文档，并且开发了 Perl 和 PHP 中用于 SQL 编程的库。我在那些网络邮件列表和新闻讨论组中回答了成千上万的问题。大量重复的问题表明程序员总是一遍又一遍地犯同样的错误。

1.1 谁需要这本书

不管你是初学者还是资深人员，这本《SQL 反模式》[②]都能帮助需要使用 SQL 的程序员更有效地使用它。我和所有不同经验层次的人交流过，他们都能从这本书中获益。

你可能已经阅读过 SQL 语法的参考资料，知道所有的 SELECT 语句的子句，并且能够用它来做一些事情了。渐渐地，你可以通过阅读别人的程序代码或者文章来提升你的 SQL 技能。但你怎么能区分优劣？怎么能确定你正在学习的是最佳方法，而不是另一个可能使你陷入困境的方法？

你可能会在本书中找到一些熟悉的话题。即使你之前已经找到了解决方案，仍可以发现新的看待问题的方法。重新审视那些广为流传的错误，将能更好地确认和巩固你对优秀范例的理解。还有一些话题可能对你来说比较新鲜，我希望你能通过阅读它们来改善自己的 SQL 编程习惯。

如果你是一个训练有素的数据库管理员，可能已经知道了如何用最好的方法来避免本书所描述的 SQL 编程中易犯的错误。然而这本书也能从软件开发人员的角度来帮助你。开发人员和 DBA 之间为项目争论是很常见的，但相互尊重和团队合作能够帮助我们更有效地工作。你可以使用本书来向你负责开发的同事解释一些好的做法，以及不这么做会有怎样的后果。

1.2 本书内容

什么是“反模式”？*反模式*是一种试图解决问题的方法，但通常会同时引发别的问题。反模

① 阻抗失配（impedance mismatch）原意为当滤波器输出阻抗 Z0 和与之端接的负载阻抗 ZL 不相等时，在该端口 A 上产生反射，引入到计算机科学中指面向对象应用程序向关系型数据库存储数据时的数据不一致问题。——译者注

② 反模式英文为 Antipattern，在 SQL 中，pattern 有时译为范式，本书中采用模式一词。——译者注

式虽以不同的形式被广泛实践，但这其中仍存在一定的共通性。人们可能独立地，或是在一个同事、一本书或者一篇文章的帮助下想出一个反模式的主意。很多面向对象的软件设计与项目管理方面的反模式都记录在 Portland Pattern Repository[①]中，或是记录在 1998 年出版的《反模式》[②]（William J. Brown 等）一书中。

本书描述了我做技术支持、做培训课程、和程序员一起开发软件以及在网络论坛上回答问题时遇到的最常见的错误。这其中很多错误我自己也犯过，除了在深夜花好几个小时找到并解决掉之外并没有更好的办法。

1.2.1 本书结构

这本书将反模式分类成如下四部分。

逻辑数据库设计反模式

在你开始编码之前，需要决定数据库里存储什么信息以及最佳的数据组织方式和内在关联方式。这包含了如何设计数据库的表、字段和关系。

物理数据库设计反模式

在知道了需要存储哪些数据之后，使用你所知的 RDBMS 技术特性尽可能高效地实现数据管理。这包含了定义表和索引，以及选择数据类型。你也要使用 SQL 的“数据定义语言”，比如 CREATE TABLE 语句。

查询反模式

你需要向数据库中添加然后获取数据。SQL 的查询是使用“数据操作语言”来完成，比如 SELECT、UPDATE 和 DELETE 语句。

应用程序开发反模式

SQL 应该会用在使用 C++、Java、Php、Python 或者 Ruby 等语言构建的应用程序中。在应用程序中使用 SQL 的方式有好有坏，该部分内容描述了一些常见错误。

很多反模式章节都采用了一些比较幽默或是能产生共鸣的标题，比如金锤子、重新造轮子或由委员会设计。通常来说，会同时给出正面的设计模式和反模式的名字作为隐喻或帮助记忆。

附录提供了一些对关系数据库理论实践的描述。本书中的很多反模式其实都是对数据库理论的误解造成的。

---

① Portland Pattern Repository：http://c2.com/cgi-bin/wiki?AntiPattern。

② 本书由人民邮电出版社于 2007 年出版。——编者注

1.2.2 反模式分解

每个反模式章节都包含了如下的子标题结构。

目的

这是你可能要去尝试解决的任务。意图使用反模式提供解决方案，但通常会以引起更多问题而告终。

反模式

这一部分表述了通常使用的解决方案的本质，并且展示了那些没有预知到的后果，正是这些使得这些方案成为反模式。

如何识别反模式

一些固定的方式会有助于你辨识在项目中使用的反模式。你遇到的特殊障碍，或是你自己和别人说的一些话，都能使你提前识别出反模式。

合理使用反模式

规则总有例外。在某些情况下，本来认为是反模式的设计却可能是合理的，或者说至少是所有的方案中最合理的。

解决方案

这一部分描述了首选的解决方案，它们不仅能够解决原有的问题，同时也不至于引起由反模式导致的新问题。

1.3 本书未涉及的内容

我不打算讲解 SQL 的语法或者相关术语。有大量关于这些基础内容的书籍或者网络资料。我假设你已经学习了足够多的 SQL 的语法来使用这门语言并且能够做好一些事情。

性能、可伸缩性以及优化对于很多开发数据驱动应用程序，特别是网络应用的设计人员来说是非常重要的。市面上也有很多和数据库开发性能相关的书籍。我推荐 *SQL Performance Tuning*[GP-3]和 *High Performance MySQL, Second Edition*[SZT+08][①]。本书的一些主题和性能相关，但这不是本书最主要的目的。

我尝试把问题表述成适用于所有厂商的数据库，并且这些解决方案也将会适用于所有的数据库。SQL 语言被规定为一种 ANSI 和 ISO 标准，所有的数据库都支持这一标准，因此我尽可能中

① 《高性能 MySQL（第二版）》由电子工业出版社于 2010 年出版。——编者注

立地使用 SQL 而不偏向任何品牌，并且我会明确说明不同厂商 SQL 数据库的特定扩展。

数据访问框架和对象关系映射（ORM）库是非常有用的工具，但这些也并不是本书的重点。我用最普通、最直白的方式写过很多 PHP 的代码范例。本书的范例足够简单，从而能够在大部分编程语言中适用。

数据库的管理和操作任务，诸如服务器磁盘分配、安装和配置、监控和备份、日志分析以及安全都是非常重要并且值得用一整本书来描述的，但本书更倾向于那些使用 SQL 的开发人员而不是数据库管理员。

这本书是关于 SQL 和关系型数据库的，以及其他的替代技术，诸如面向对象数据库、键/值存储、面向列的数据库、面向文档的数据库、分级数据库、网络数据库、Map/Reduce 框架或者语义数据存储。比较这些技术的优缺点并在不同的数据管理解决方案中恰当地使用它们是一个很有趣的课题，但这并不是本书的议题。

1.4 规约

下面描述了本书中使用的一些规约。

排版

SQL 关键字都大写并且使用等宽字体，以使得它们在上下文中更突出，就像 SELECT。

SQL 的表名，也用等宽字体，并且首字母大写，如 Accounts 或者 BugsProducts。SQL 的列，也使用等宽字体，全都使用小写，并且使用下划线分词，如 account_name。

术语

SQL 的正确发音是 S-Q-L，不是 C-QL。虽然我对通俗用法并没有什么反对意见，但我更倾向于使用正式用法。

在数据库相关的用法中，“索引”一词指的是一个有序的信息集合。在其他情况下“索引”可能指的是指示器。

在 SQL 中，术语*查询*（query）和*语句*（statement）有时指的是同一个意思，指的都是任何可执行的一句完整的 SQL 指令。为了描述清晰，我使用“查询”表示 SELECT 语句，而“语句”用来表达其他语句，包括 INSERT、UPDATE 和 DELETE 以及数据定义语句。

数据实体关系图

最常见的关系数据库图表就是*实体关系图* ERD（Entity Relationship Diagram）。表格使用矩形表示，关系使用链接各个矩形的线段来表示，并且在两端使用各种符号来表述关系的基数。具体事例可以参考下一页的图 1-1。

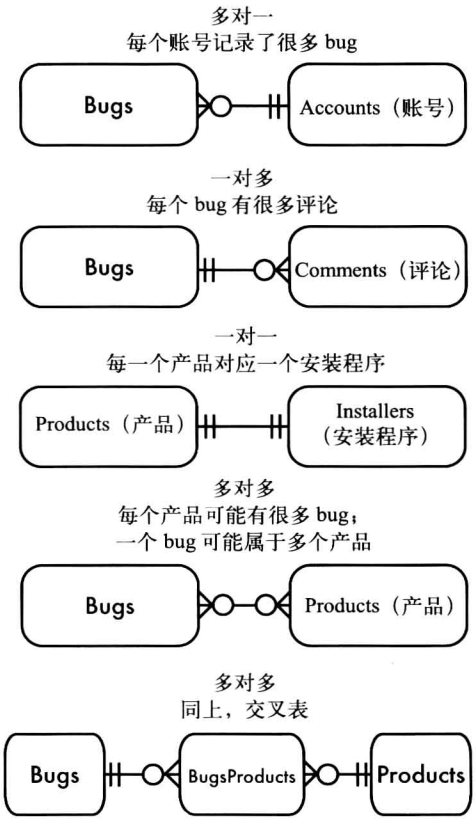


图 1-1 实体关系图 ERD 示例

1.5 范例数据库

我使用一个假想的缺陷跟踪程序的数据库来展示大部分与 SQL 反模式相关的话题。图 1-2 是该数据库的 ERD。请注意，Bugs 表和 Accounts 表之间有 3 个连接，代表 3 个不同的外键。

接下来的数据库定义语句则展示出如何定义这些表。有些时候所做的选择只是为了照顾后面的范例，因而它们可能并不是人们在实际应用程序中所做的选择。我尽力使用标准 SQL 来定义这个数据库，使其能适用于任一数据库产品，但也会出现一些 MySQL 数据类型，诸如 SERIAL 和 BIGINT 语句。

Introduction/setup.sql

```
CREATE TABLE Accounts (
  account_id        SERIAL PRIMARY KEY,
  account_name      VARCHAR(20),
  first_name        VARCHAR(20),
  last_name         VARCHAR(20),
```