

## <<SQL沉思录>>

### 图书基本信息

书名：<<SQL沉思录>>

13位ISBN编号：9787115213952

10位ISBN编号：711521395X

出版时间：2009-11

出版时间：人民邮电出版社

作者：Joe Celko

页数：270

译者：马树奇

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

## &lt;&lt;SQL沉思录&gt;&gt;

## 前言

本书讨论的是使用表而不是过程式代码的各种SQL编程技术。

我一直跟大家说，学习SQL编程最重要的就是要摒弃原有的过程式编程，但对于那些一直都在以文件和过程代码的方式来思考的人们而言，仅仅这样说不行，还得具体说明如何采用声明式关系语言来完成相关工作。

因此我编写了本书，展示实际技术，阐述思想方式。

与我的其他作品相同，本书的读者对象是那些希望掌握良好SQL编程技术的专业SQL程序员。

本书假定读者对SQL语言足够了解，已经能够编写可以运行的代码，具有一年的SQL使用经验。

为什么需要具有一年的经验呢？

我通过几十年讲授SQL课程发现，大多数人在由过程式编程语言（如FORTRAN、Cobol、Pascal、C系列语言）、面向对象编程语言等转向SQL语言时，都要经过多个阶段，而这段时间基本为一年。

声明式语言与他们此前使用的所有编程语言都不相同。

学习一门新的编程语言就如同学习一门外语。

一开始，人们会念错单词，并且总想使用自己母语的词序和语法。

接下来，人们会通过努力，根据一种固定模式来正确地造句。

最后，人们才能轻松地以该语言来思考和表述，并且不需要再刻意关注语言本身。

SQL编程的初级阶段只不过是闭着眼睛从别人的程序中把现有的代码复制过来，这并不是真正的编程。

人们可能还会使用一种具有图形用户界面的工具来从文本文件中把SQL语句组合起来，甚至根本不需要显示自己的实际代码。

## <<SQL沉思录>>

### 内容概要

《SQL沉思录》通过大量的实例，详细说明了为提高SQL编程技术而必须面对的思想方法上的根本转变——由以过程式编程方式思考转变为以数据集的方式来思考。

此外，《SQL沉思录》还讨论了关于SQL编程中查找表、视图、辅助表、虚拟表的应用，并独到地阐明了如何在SQL系统中正确地处理时间值以及SQL编程中的其他技术难点。

《SQL沉思录》适合广大数据库编程人员和SQL程序员学习参考。

## <<SQL沉思录>>

### 作者简介

Joe Celko, 世界著名的数据库专家, 曾担任ANSI SQL标准委员会成员达10年之久, 他也是世界上读者数量最多的SQL图书作者之一。他曾撰写过一系列专栏, 并通过他的新闻组支持和推动了数据库编程技术以及ANSI/ISO标准的发展。除本书外, 他还撰写了多部SQL经典著作, 包括《SQL编程风格》、《SQL解惑》和《SQL权威指南》, 上述作品的中文版均已经或即将由人民邮电出版社出版。

## 书籍目录

第1章 SQL是声明式语言，不是过程式语言 11.1 不同的编程模型 11.2 不同的数据模型 31.2.1 “列”不是“字段” 41.2.2 行不是记录 61.2.3 表不是文件 91.2.4 关系键不是记录定位器 111.2.5 键的类型 121.2.6 关系键的理想属性 141.2.7 唯一，但并非不变 151.3 表作为实体 151.4 表作为关系 161.5 语句不是过程 161.6 分子、原子和亚原子型数据元素 171.6.1 分割表 171.6.2 分割列 181.6.3 时间值的分割 191.6.4 假造的非第一范式数据 191.6.5 分子型数据元素 211.6.6 异构数据元素 211.6.7 检验分子型数据 22第2章 硬件、数据量和维护数据库 232.1 并行处理技术 232.2 廉价的主存储器 252.3 固态硬盘 252.4 更廉价的二级存储器和三级存储器 252.5 数据也在改变 262.6 思维方式并未改变 26第3章 数据访问和记录 293.1 顺序访问 293.2 索引 303.2.1 单表索引 313.2.2 多表索引 313.2.3 索引的类型 323.3 散列 323.3.1 数字选择 333.3.2 除法散列 333.3.3 乘法散列 333.3.4 合并 333.3.5 表的查找 333.3.6 冲突 343.4 位向量索引 343.5 并行访问 343.6 行和列存储 353.6.1 基于行的存储 353.6.2 基于列的存储 353.7 联结算法 363.7.1 嵌套循环联结算法 373.7.2 排序合并联结算法 373.7.3 散列联结算法 373.7.4 Shin算法 38第4章 查找表 394.1 数据元素的名称 404.2 多参数查找表 424.3 常量表 434.4 OTLT或MUCK表问题 454.5 正确表的定义 48第5章 辅助表 495.1 序列表 495.1.1 创建序列表 515.1.2 序列构造器 515.1.3 替换迭代循环 525.2 排列 545.2.1 通过递归进行排列 545.2.2 通过CROSS JOIN进行排列 555.3 函数 575.4 通过表实现加密 595.5 随机数 605.6 插值 63第6章 视图 666.1 Mullins视图使用原则 666.1.1 高效访问和计算 676.1.2 重命名列 686.1.3 避免增生 686.1.4 视图同步原则 686.2 可更新视图和只读视图 696.3 视图的类型 716.3.1 单表投影和限制 716.3.2 计算列 716.3.3 转换列 726.3.4 分组视图 726.3.5 联合视图 736.3.6 视图的联结 746.3.7 嵌套视图 756.4 用表构建类模型 766.4.1 SQL中类的层次结构 776.4.2 通过ASSERTION和TRIGGER工作的子类 796.5 数据库系统如何处理视图 796.5.1 视图列的列表 796.5.2 视图的物化 806.6 嵌入式文本扩展 806.7 WITH CHECK OPTION子句 816.8 删除视图 866.9 过时的视图用法 876.9.1 域的支持 876.9.2 表表达式视图 886.9.3 表级CHECK()约束的视图 886.9.4 每个基表一个视图 88第7章 虚拟表 907.1 派生表 907.1.1 列的命名规则 917.1.2 作用域规则 917.1.3 公开的表名 937.1.4 LATERAL()子句 947.2 CTE 967.2.1 非递归CTE 967.2.2 递归CTE 977.3 临时表 987.3.1 ANSI/ISO标准 997.3.2 厂商的模型 997.4 信息模式 997.4.1 INFORMATION\_SCHEMA声明 1007.4.2 视图及其用途的快速列表 1017.4.3 域的声明 1027.4.4 定义模式 1027.4.5 INFORMATION\_SCHEMA断言 105第8章 用表实现的复杂函数 1068.1 没有简单公式的函数 1068.2 用表实现校验位 1078.2.1 校验位的定义 1078.2.2 检错与纠错的对比 1088.3 算法的分类 1098.3.1 加权和算法 1098.3.2 幂和校验位 1118.3.3 Luhn算法 1128.3.4 Dihedral Five校验位 1138.4 声明不是函数，不是过程 1148.5 用于辅助表的数据挖掘 118第9章 时态表 1209.1 时间的本质 1209.1.1 时间段，不是时间子 1219.1.2 细分程度 1229.2 ISO半开放时间模型 1239.2.1 用NULL表示永远 1259.2.2 单时间戳表 1259.2.3 重叠的时间间隔 1279.3 状态转换表 1349.4 合并时间间隔 1389.4.1 游标和触发器 1399.4.2 OLAP函数解决方案 1409.4.3 CTE解决方案 1419.5 Calendar表 1429.5.1 用表提供星期值 1429.5.2 节假日列表 1439.5.3 报告期 1459.5.4 自更新视图 1459.6 历史表 147第10章 用非第一范式表清理数据 14910.1 重复的组 14910.2 设计清理表 15510.3 清理操作使用的约束 15710.4 日历清理 15810.5 字符串清理 15910.6 共享SQL数据 16110.6.1 数据的发展 16210.6.2 数据库 16210.7 提取、转换和加载产品 16310.7.1 加载数据仓库 16410.7.2 全部用SQL来完成 16510.7.3 提取、转换并加载 166第11章 以SQL的方式思考 16811.1 热身练习 16811.1.1 整体，不是部分 16911.1.2 特征函数 16911.1.3 尽早锁定解决方案 17111.2 启发式方法 17211.2.1 将规范表达为清晰的语句 17211.2.2 在名词前面添加“所有……的集合”几个字 17211.2.3 删除问题语句中的行为动词 17311.2.4 仍然可以使用存根 17311.2.5 不要担心数据的显示 17411.2.6 第一次尝试需要专门处理 17511.2.7 不要害怕抛弃自己在DDL中的首次尝试 17511.2.8 克制使用DML的冲动 17611.2.9 不要以方框和箭头的方式思考 17611.2.10 画圆和数据集示意图 17711.2.11 学习具体的产品 17811.2.12 把WHERE子句看做“超级变形虫” 17811.2.13 使用新闻组、博客和因特网 17811.3 不要在SQL中使用BIT或BOOLEAN标记 17911.3.1 标记位于错误的层 17911.3.2 标记使用不当使正确属性难以理解 181第12章 组特征 18412.1 并不是按是否相等来分组 18512.2 使用组，不看里面是什么 18612.2.1 半面向数据集的方式 18712.2.2 分组的解决方案 18812.2.3 解决方案总结 18912.3 根据时间分组 19012.3.1 渐进式解决方案 19012.3.2 整体数据解决方案 19212.4 其他使用HAVING子句的技术 19212.5 GROUPING、ROLLUP和CUBE 19412.5.1 GROUPING SET子句 19412.5.2 ROLLUP子句

## &lt;&lt;SQL沉思录&gt;&gt;

19512.5.3 CUBE子句 19612.5.4 关于超级组的脚注 19612.6 WINDOW子句 19612.6.1 PARTITION BY子句  
19712.6.2 ORDER BY子句 19812.6.3 RANGE子句 19812.6.4 编程技巧 199第13章 将技术规范变为代码  
20013.1 不良SQL的标志 20013.1.1 代码的格式是否像另一种语言 20013.1.2 顺序访问假设 20113.1.3 游标  
20113.1.4 糟糕的内聚度 20113.1.5 表值函数 20213.1.6 同一数据元素有多个名称 20213.1.7 数据库中的格式  
20213.1.8 将日期保存到字符串中 20313.1.9 BIT标记、BOOLEAN及其他计算列 20313.1.10 跨列的属性分  
割 20313.1.11 跨行的属性分割 20313.1.12 跨表的属性分割 20313.2 解决方法 20413.2.1 基于游标的解决方  
案 20413.2.2 半面向数据集的解决方案 20513.2.3 完全面向数据集的解决方案 20713.2.4 面向数据集代码的  
优点 20713.3 解释含糊的说明 20713.3.1 回归到DDL 20913.3.2 修改问题说明 211第14章 使用过程及函数调  
用 21314.1 清除字符串中的空格 21314.1.1 过程式解决方案#1 21314.1.2 函数解决方案#1 21414.1.3 函数解  
决方案#2 21714.2 聚合函数PRD() 21814.3 在过程和函数中使用长参数列表 220第15章 对行编号 22315.1  
过程式解决方案 22315.2 OLAP函数 22615.2.1 简单的行编号 22615.2.2 RANK()和DENSE\_RANK() 22715.3  
节 228第16章 保存计算数据 23116.1 过程式解决方案 23116.2 关系式解决方案 23216.3 其他种类的计算数  
据 233第17章 约束类触发器 23417.1 计算类触发器 23417.2 通过CHECK()和CASE约束实现的复杂约束  
23517.3 通过视图实现复杂约束 23717.4 用约束实现视图操作 23917.4.1 3个基本操作 23917.4.2 WITH  
CHECK OPTION子句 24017.4.3 WITH CHECK OPTION与CHECK()子句 24317.4.4 视图的行为 24417.4.5  
联合视图 24617.4.6 简单的INSTEAD OF触发器 24717.4.7 关于INSTEAD OF触发器的告诫 250第18章 过程  
式解决方案和数据驱动的解决方案 25118.1 删除字符串中的字母 25118.1.1 过程式解决方案 25218.1.2 纯  
粹的SQL解决方案 25218.1.3 不纯粹的SQL解决方案 25318.2 数独的两种求解方法 25418.2.1 过程式解决方  
案 25418.2.2 数据驱动的解决方法 25418.2.3 处理已知数字 25518.3 数据约束方法 25718.4 装箱问题  
26118.4.1 过程式解决方法 26118.4.2 SQL方式 26218.5 库存成本随时间的变化 26418.5.1 库存中使用的  
UPDATE语句 26718.5.2 回到装箱问题 268

## 章节摘录

第1章 SQL是声明式语言，不是过程式语言 前言里，我谈到了一些FORTRAN程序员和一名LISP程序员的事，前者只会使用循环来解决问题，后者只会使用递归方式解决问题。这种情况并不少见，因为人们都喜欢使用自己了解的工具。

下面讲一个笑话，不是真事：有人给一个数学家、一个物理学家和一个数据库程序员各发了一个橡皮球，并且让他们确定球的体积。

数学家认真地测量了直径，然后用球体积公式计算出了球的体积，或者认为这个球不很圆，就用三重积分计算了球的体积。

物理学家则在一个大烧杯中接满了水，把球放入水中，测量出排水量。他并不关心这个球是什么形状。

数据库程序员呢，他到橡皮球生产商的在线数据库里查了这个球的型号和产品序列号，根本不关心这是不是球。

他获得了这个球的制造公差、设计形状和尺寸以及其他许多与整个橡皮球生产过程有关的参数。

这个故事说明：数学家知道如何计算，物理学家知道如何测量，而数据库技术人员知道如何查找数据。

每个人都采用自己的工具来解决问题。

现在我们把问题扩展到仓库中成千上万个橡皮球。

数学家和物理学家因此会花费大量的手工劳动完成任务，而数据库技术员只要下载一些信息，就能够得出橡皮球的工业标准（假设有这种标准）以及详尽得可以用于法庭辩论的文档。

1.1 不同的编程模型 自我完善的过程就是在学习新知识的同时，忘记老的习惯。

——Edsger Dijkstra 编程模型有多种。

过程式编程语言使用的是由流控制语句（WHILE—DO、IF—THEN—ELSE、BEGIN—END）控制的一系列过程步骤，借此把输入数据转换成输出数据。

这是对编程的一种传统认识，因为这是著名的数学家约翰·冯·诺伊曼归纳出来的，后来也常被称为冯·诺伊曼模型。

同样的源代码经相同的编译器编译之后，每次都生成相同的可执行模块。

该程序在每次调用时都以完全相同的方式工作。

这种模型中的关键字是可以预测和确定的。

由于这种模型具有可确定性，所以主要用于一些数学分析。

另外，还有一些变化。

一些语言使用了不同的流控制语句。

FORTRAN和HCOBOL会在程序一开始就为数据分配全部存储区。

后来的Algol系列编程语言会根据数据在程序块结构中的作用域动态地分配存储区。

Edsger Dijkstra（参见文献[www.CS.utexas.edu/users/EWD/](http://www.CS.utexas.edu/users/EWD/)）发明了一种非确定性语言。

语句，又称为保护命令，既可以阻止语句的执行，也可以允许该语句的执行，而且在打开的语句之间没有确定的执行顺序。

这种模型没有在商业化产品中实现，但它表明人们原来在编程中认为必备的因素（确定性）可以被丢弃。

函数式编程语言的基础是用一系列嵌套的函数调用来解决问题。

在这些语言中，高阶函数可以转换自身的功能，这个概念非常重要。

导数变换和积分变换就是这种高阶函数在数学上应用的实例。

这种语言的目标之一是避免在程序中出现副作用，保证它们能够以代数的方式进行优化。

特别是，一旦某个表达式与另一个表达式相等（某种意义上的相等），它们就可以替换，而不会影响整个运算结果。

<<SQL沉思录>>

媒体关注与评论

“ Joe Celko写的所有的书我都买了，这是其中写得最好的一本……强烈推荐！” ——Amazon.com读者评论

## <<SQL沉思录>>

### 编辑推荐

SQL是数据库的标准数据查询语言，在广大程序员的日常工作中已必不可少。但是，大部分的SQL程序员都是由过程式编程人员和面向对象编程人员转变而来的，他们已经习惯了原有的思维方式，若要充分利用SQL必须学会以数据集的方式思考。

《SQL沉思录》中，世界级SQL专家、Joe Celko通过展示实用技术及思想方法，教会大家如何完成这个转变。

书中通过大量实例详细介绍了各种SQL编程技术，内容涉及查找表、辅助表、虚拟表、时态表、视图、SQL的思考方式等多个方面，值得所有SQL程序员仔细研读和思考。

世界级SQL专家经典著作    深入揭示SQL编程本质    教你不同于过程和OO编程的全新思考方式

<<SQL沉思录>>

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>