TURING 图灵程序设计丛书 数据库系列

ELSEVIER

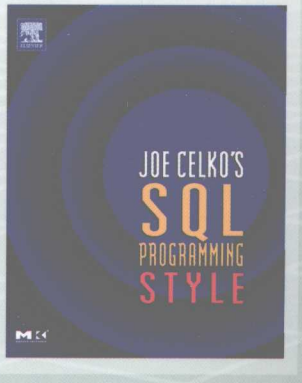
Joe Celko's SQL Programming Style

SQL编程风格

[美] Joe Celko 著

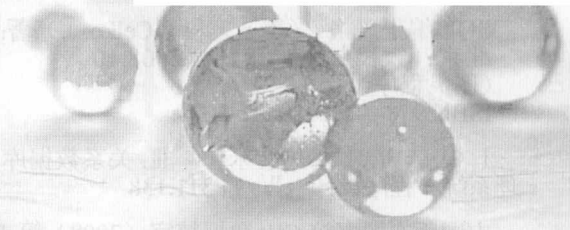
米全喜 译

- 世界级大师的SQL编程规范
- 讲述如何编写标准、高效、易于维护的SQL代码
- 教你像优秀的SQL程序员那样思考



人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 数据库系列



Joe Celko's SQL Programming Style

SQL编程风格

[美] Joe Celko 著
米全喜 译

人民邮电出版社
北 京

图书在版编目（CIP）数据

SQL 编程风格 /（美）塞科（Celko, J.）著；米全喜译 .
北京：人民邮电出版社，2008.10
（图灵程序设计丛书）
ISBN 978-7-115-18582-2

Ⅰ. S… Ⅱ. ①塞…②米… Ⅲ. 关系数据库－数据库管理系统－程序设计 Ⅳ. TP311.138

中国版本图书馆 CIP 数据核字（2008）第 115519 号

内 容 提 要

本书针对数据库的设计与编程提出了一系列规则和建议，内容涵盖命名规范、代码版式、键的设计、数据编码方案、编码风格、视图和存储过程的使用以及 SQL 中的思考方式和一些试探法等多方面。这些规则都给出了原理说明和例外情况，并列举了大量示例。通过阅读本书，读者可以加深对 SQL 思维方式的理解，改善 SQL 编程风格，并编写出可读性强、可移植且易于维护的 SQL 代码。此外，书中的规则对于公司内部制定编程规范也具有很好的借鉴作用。

本书适合数据库管理人员和开发人员阅读，也可作为高等院校计算机专业师生的参考教材。

图灵程序设计丛书

SQL编程风格

◆ 著 [美] Joe Celko
译 米全喜
责任编辑 刘艳娟

◆ 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 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京隆昌伟业印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：13
字数：256千字 2008 年 10 月第 1 版
印数：1－4 000册 2008 年 10 月北京第 1 次印刷

著作权合同登记号 图字：01-2008-2935号

ISBN 978-7-115-18582-2/TP

定价：39.00元

读者服务热线：(010)88593802 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

Joe Celko's SQL Programming Style
by Joe Celko
ISBN 0-12-088797-5

Authorized Simplified Chinese translation edition published by the Proprietor.
ISBN: 978-981-272-113-6

Elsevier (Singapore) Pte Ltd.
3 Killiney Road
#08-01 Winsland House I
Singapore 239519
Tel: (65)6349-0200
Fax: (65)6733-1817

First Published 2008
2008 年初版

Printed in China by POSTS & TELECOM PRESS under special arrangement with Elsevier (Singapore) Pte Ltd. This edition is authorized for sale in China only, excluding Hong Kong SAR and Taiwan.

译 者 序

数据库发展到今天，可以说是无处不在。作为程序员，我们或多或少都需要与数据库打交道。在设计数据库或使用SQL编程的时候，我们是否注意到了SQL与其他过程化或面向对象编程语言的差异？我们是否有一套可行的编程规范？我们考虑了代码的可读性、可移植性和性能吗？我们是在以SQL的思考方式使用SQL吗？

按照本书作者Joe Celko的观察，很多人都还做不到，而这也正是编写本书的目的。本书的第1～2章介绍了命名规范和代码的版式。第3～5章介绍了数据定义语言，说明了在设计键、编写约束、选择数据类型、设置默认值时应当注意的事项，并介绍了测度论、编码方案及其在数据库系统设计中的应用。第6～8章介绍了数据操作语言，从可读性、可移植性和性能等方面说明了编写程序和使用视图、存储过程时的注意事项。第9章和第10章介绍了SQL中的思考方式和试探法。这本书的书名虽然是编程风格，但它更像是一本SQL编程指导书。作者从事SQL咨询工作20多年并长期活跃在SQL社区，他知道程序员们经常犯什么错误，需要在哪些方面进行改进。本书通过大量的示例以及作者的见闻和亲身经历，详细说明了编程时应当遵守的规则——什么是应该做的，什么是不应该做的。这不是一本入门书，讲述的不是如何使用SQL编程的基础内容，但是通过对本书的学习，可以帮助我们更好地理解SQL的思考方式，从而设计出更好的数据库和SQL程序。

人民邮电出版社最近引进了Joe Celko的几本著作，有这样一位大师的指引，必然会提高我们对SQL的理解和应用水平。非常感谢图灵公司几位编辑对我的信任，将其中两本的翻译工作交给了我。本书篇幅不大，但是涉及的内容较广，译者在翻译过程中尽量使得译文准确、流畅，但其中难免有疏忽之处，欢迎读者批评指正，请将意见发送到miquanxi@hotmail.com。

译 者

2008年6月

前　言

本书不是一本SQL入门书。真的，如果你需要的是学习如何使用SQL进行编程，有其他更好的书。本书应该是你买的第二本书，而不是第一本。

本书假设你已经能够编写一定水平的SQL，并且希望进一步提高。如果你想要学习SQL编程技巧，可以买一本我编写的Joe Celko's *SQL for Smarties*（2005年第3版）[①]。在本书中，我想教给读者的，是如何以逻辑和说明性的方式编程，而不是以过程化或面向对象的方式——也就是要“用查询的思维看数据库”[②]。

绝大多数SQL程序员都是在有了几年的过程化语言或面向对象语言编程经验之后才开始接触SQL的。他们拿到某个SQL产品，然后只能自学，使用的书都是“SQL for Brain-Dead Morons（SQL傻瓜书）”、“Learn SQL in Ten Easy Lessons or Five Hard Ones（用10节容易的或5节复杂的课程学会SQL）”或其他更烂的书。

这太荒唐了！成为熟练的木匠或厨师都至少需要5年。为什么你会相信人们在一个周末内就能变成SQL高手？他们会变成糟糕的SQL程序员，只会使用本地SQL产品中的方言，并带有他们从前使用的编程语言的浓重口音。你可能需要读一下Peter Norvig写的“Teach Yourself Programming in Ten Years”一文（www.norvig.com/21-days.html）或Fred Brooks写的“No Silver Bullets”（*Computer*，20(4):10-19，1987年4月），以了解真实情况。

可怕的是这些人常常不知道自己是拙劣的程序员。一个极端情况是，整个公司的人都做得很差，他们从来没有见过高手。另外一个极端情况是，如果某人尝试告诉他们所

① 中文版《SQL权威指南》将由人民邮电出版社出版。——编者注

② 此句原文“query eye for database guy”套用了美国流行语“…eye for…guy”，最初源于一个流行电视节目。——编者注

存在的问题，他们会极力辩解或恼羞成怒。看看SQL新闻组中的帖子，你会发现许多程序员只是需要对问题立即得到一个权宜解答，并不想真正获得一个长期有效的解决方案。

如果这些是木工行业的新闻组，他们的问题将等同于“将螺丝钉敲进精致的家具时，使用什么样的石头最好？”如果有人告诉他使用一大块花岗岩，他们会很高兴，但是如果你告诉他们使用螺丝刀，他们就愤怒了。

你可能需要阅读由Justin Kruger和David Dunning（康奈尔大学心理学系）写的有关这个现象的一篇文章“Unskilled and Unaware of It: How Difficulties in Recognizing One's Own Incompetence Lead to Inflated Self-Assessments”（www.apa.org/journals/psp/psp7761121.pdf）。或者是看一下美国高中生在数学和自然科学的自我评估结果和实际结果的对比，这是布什政府的*No Child Left Behind Act*（《不让一个儿童落后法》）的一部分。

本书的目的

当那些过时的程序员当道的时候，我们这些老家伙们是如何成为更好的程序员的呢？在20世纪70年代末掀起结构化编程革命的时候，对我们最有帮助的是由Henry Ledgard和他在MIT的一些同事编写的一系列名为“[PASCAL | FORTRAN | COBOL | BASIC] with Style: Programming Proverbs”的书，封面上有天使、饰带和老式的印刷元素，看上去就像维多利亚时代的小说一样。这些书的副标题是“Principles of Good Programming with Numerous Examples to Improve Programming Style and Proficiency”。对我们大多数人来说，这些书与其他书有很大差别，因为它们教会了我们如何像优秀程序员那样思考。

本书的目的是改善SQL编程风格和提高SQL编程水平。更准确地说是以下几点。

(1) 帮助程序员编写不带口音或方言的标准SQL。积习难改，但并非不可能，不过最好还是从一开始就学习正确的方式。业余人士是为自己编写代码，而专业人士编写的代码则由其他人维护和使用。我的经验是需要一整年的SQL编程才能领悟其中的真谛，突然领悟到SQL领域可以分为3部分：值的逻辑、数据模型和集合。

(2) 为使用SQL的公司提供一个内部使用的编程规范。我认真地尝试着为我的每个规则都给出一个原理，同时给出我能够想到的例外情况。你可能不赞成我的某些选择，

但是你必须提供研究结果和示例来捍卫你的观点。简单地把“这是我们在FooTran中编写代码的方式，那它一定是上帝的意愿”作为论据是不够的。

如果你是小组负责人，你现在有了一本书（及其作者）可作为依据，对小组成员不喜欢做的事情挑错。即使后来证明我对于某个事物的观点是错误的，你们小组内部也能够保持一致风格。如果所犯的错误一致，会很容易纠正。

(3) 给程序员提供一些思考方法，用SQL来处理新的问题。我认为需要大约一年才能真正掌握，并去掉那些过程化编程的习惯。

致谢

Craig Mullins在www.DBAzine.com的一篇文章中提供了本书关于视图那一章的结构。代码的格式化风格取自我过去十多年间在CMP公司多个杂志以及其他出版物中使用的版式。Peter Gulutzan在www.DBAzine.com中的一篇文章中提供的数据说明了实际产品中的命名规范。第1章中的词缀规范基于Teradata公司的内部标准。有关尺度与测量及编码方案的资料出现在我在*DBMS*和*Database Programming & Design*杂志上的几个老专栏中，后来收录到我的另一本书*Joe Celko's Data & Database*（Morgan-Kaufmann出版社）中。我本来想在本书中感谢对本书有贡献的每一个人，但是这些年来参与新闻组的人太多了，就恕不一一列出了。

显然，我还要感谢Henry Ledgard和他的“Programming Proverbs”系列带给我的启发。

我也要感谢所有编写糟糕代码的程序员新手。听上去有点像是在讽刺，不过我不是这个意思。很多程序员新手没有接受过训练，也没有配备一位有经验的导师，就被管理人员派去从事DBA或SQL程序员的工作。除非他们真的不想把工作做得更好，否则我也不想责备这些受害者。他们在语法、语义和风格上犯的错误告诉了我他们是如何思考的。诊断是治疗的第一步。

校正、评论及未来的版本

对未来版本的校正和内容的增加可以直接发送给Morgan Kaufmann出版社，也可以发到我的电子邮箱jcelko212@earthlink.net。

目　录

第1章

名称与数据元素

有一个老笑话：

“在我小的时候，我们养了3只猫。”
“它们都叫什么名字？”
“猫、猫和猫。”
“这听着也太乱了。你是怎么区分它们的？”
“谁管这个？反正你叫它们的时候它们也不会过来。”

如果不能总是为数据起一个清晰的、可识别的名称，那么当你调用数据的时候，它们不会出来。这对于任何一个数据库项目来说都很重要。数据元素没有一个好的名称，将使代码很难甚至无法阅读。

说到无法阅读，我不是在开玩笑。以前，软件公司经常刻意将源代码的名称打乱并删除格式，以将算法对购买软件的人隐藏起来。这个传统似乎在有意无意间延续了下来。2004年8月，在一个SQL新闻组中出现了一个帖子，其中所有的名称都是由一个字母和一长串数字组成。

现在有了用于描述命名数据元素和注册标准的规则的ISO-11179元数据标准。因为它们是ISO标准，所以你不仅要在SQL中使用它们，在其他任何地方都要使用。

有了这个标准，再加上一点版式和一些常识，就可以给出入门所需要的规则了。

1.1 名称

以前，每个程序员都有一套自己的命名规范。但是，他们常常都太有创造性了。我特别喜欢举的一个例子是，有一个人使用某类主题词作为他的COBOL段名：一段程序可能使用国家名，另外一段可能使用花卉，等等。即使就程序员而言，这显然也是很奇怪的行为，但是很多程序员的个人命名系统只有他们自己明白，别人都无法理解。

例如，我使用的第一个FORTRAN版本只允许6个字母的名称，所以我变得善于使用和发明6个字母的名称。开始编程时使用弱类型或无类型语言的程序员们都喜欢使用匈牙利表示法（参见Leszynski和Reddick）。老的习惯很难放弃。

当软件工程变成规范后，每个公司都制定了自己的命名规范，并使用某种数据字典实施这些规范。使用最广泛的一套规则可能要算是由美国国防部建立的MIL STD 8320.1，但它在联邦政府之外却从未流行起来。这与先前缺乏有效组织的体系相比，已经有了很大进步，但是每个公司都有很大的不同：有些对于名称构造有正式的规则，而另外一些则只是将赋予数据元素的第一个名称登记一下。

现在，我们有了ISO-11179标准，它正变得越来越普遍，是某些政府工作所需要的，并且正在被放入到数据储存库产品中。一些工具和大量标准化编码方案也被放入到了这个标准中。考虑到这一点，并考虑到XML是一种标准交换格式，ISO-11179在今后将成为元数据参考的方法。

1.1.1 注意名称长度

原理　在SQL-92标准中，标识符最大长度为18个字符。这个长度是从以前的COBOL标准中得到的。现在，各种SQL的实现都允许更长的名称，但是如果用18个字符都描述不清楚，那就有问题了。表1-1按照ISO和几种流行的SQL产品，显示了几种最重要的SQL模式对象名称的最大长度。

2

表1-1　标识符长度

	SQL-92	SQL-99	IBM	MS SQL	Oracle
列	18	128	30	128	30
约束	18	128	18	128	30
表	18	128	128	128	30

表中的数字是字节或字符。如果使用的是多字节字符集，那么最大字符长度可能比最大字节长度短。

不要使用超长的名称。人们必须阅读它们、输入它们并把它们打印出来。当他们查看代码的时候，他们还必须能够理解这些名称、在数据字典中搜索它们，等等。最后，这些名称还必须在宿主程序中共享，而宿主程序可能不允许这个最大长度。

但是也不要走向另一个极端，使用那些不研究几个星期就无法理解的、高度浓缩的名称。老式的Bachman设计工具用于构造DB2数据库，那时列长度限制为18个字节。有时这个工具会通过删除所有的元音来将逻辑属性名称转换为物理列名。Craig Mullins将这个情况称为“Bachman动了DDL中的元音”。这是一个让名称具有更少字符的较差的方法。

例外 这些例外将随着具体情况的不同而变化，可能是具有不同命名限制的遗留系统的结果。

1.1.2 在名称中避免使用所有特殊字符

原理 名称中的特殊字符使数据库和宿主程序中很难或无法使用相同的名称，甚至将模式迁移到另外一种SQL产品上时也存在同样的问题。

表1-2显示了在标准中以及几种流行的SQL产品中，名称中允许用的字符。

表1-2 标识符字符集

	标准SQL	IBM	Oracle	Microsoft
第一个字符	字母	字母、$#@	字母	字母、#@
其后的字符	字母、数字、_	字母、数字、$#@_	字母、数字、$#_	字母、数字、#@_
区分大小写？	否	否	否	可选的
术语		普通标识符	非引用标识符	常规标识符

一般情况下，名称的第一个字符必须是字母，其后的字符可以是字母、数字或_（下划线）。任何一个数据库管理系统（DBMS）可能也都允许$、#或@，但是没有一个DBMS同时允许所有这3个，而且在任何情况下特殊字符不是在任何地方都可以使用的（Microsoft对以@或#开头的名称赋予了特殊含义，Oracle不允许在某些对象的名称中使用特殊字符）。

但是什么是字母呢？在最初的SQL中，所有字母都必须是大写拉丁字母，所以只有26种选择。现在范围更广了，但是需要注意Latin-1字符集之外的字符，原因如下。

(1) IBM不是总能够识别字母。它只是将除了空格之外的任何多字节字符都作为字母接受，并不会去确定它是大写还是小写。

(2) IBM和Oracle使用数据库字符集，这样对于外国字母的迁移可能有问题。Microsoft使用的是Unicode，所以没有这个问题。

中级SQL-92不允许标识符以下划线结尾。将多个下划线放在一起也不好，现代打印机让人很难数清连在一起的下划线的个数。

例外　无。

1.1.3　避免使用引号分隔标识符

4 **原理**　引号分隔标识符（Quoted Identifier）是在SQL-92中增加的。它的主要作用一直是为列名起别名，这样打印出来的内容会更像报表。这个权宜之计使分层架构失去了意义。它没有带来好处，相反，它破坏了代码的可移植性，容易导致结构不好的名称。表1-3显示了定界标识符的特征。

表1-3　引号分隔标识符字符集

	标准SQL	IBM	Microsoft	Oracle
定界符	""	""	""或[]	""
第一个字符	任意字符	任意字符	任意字符	任意字符
其后的字符	任意字符	任意字符	任意字符	任意字符
区分大小写?	是	是	可选的	是
术语	定界标识符	定界标识符	定界标识符	引号分隔标识符

如果发现名称的字符集约束过于繁杂，可以将标识符放到双引号的内部，这样就可以避免这些繁杂了。结果就产生了定界标识符（在Oracle术语中称为引号分隔标识符）。定界标识符可以由任意字符开头，也可以包含任意字符。对于如何在其中包含双引号(")，有点儿不确定。标准的方法是使用两个双引号，例如“Empl""oyees”，但是没有正式的文档记载。

对于定界名称的支持几乎是普遍存在的，只有两个主要的例外：(1)在存储过程中，IBM不允许标签和变量名是非字母数字的字符；(2) 如果QUOTED_IDENTIFIER选项是关闭的，那么Microsoft将不允许引号分隔标识符。第一个例外的原因可能是IBM在编译之前要将SQL过程转换成另外一种计算机语言。假设使用定界标识符创建一个表，例如：

```
CREATE TABLE "t" ("column1" INTEGER NOT NULL);
```

现在试着以常规标识符的方式访问这个表，如下：

```
SELECT column1 FROM t;
```

5

这样可以执行吗？按照SQL标准，不能执行，但是对于Microsoft，却可以执行。原因在于是否区分大小写，我们将在1.1.4节讨论该问题。

引号分隔标识符在宿主语言中不能很好地工作，特别是当有空格或特殊字符时。例如，下面是一个有效的插入语句：

```
INSERT INTO Table ([field with space]) VALUES (value);
```

ADO生成下面的代码：

```
INSERT INTO Table (field with space) VALUES (value);
```

这是一个语法错误。

例外 如果需要与之交流结果的人无法阅读或理解用Latin-1恰当构造的列名，那么可以使用引号分隔的别名将输出进行格式化。我曾经为母语是波兰语和汉语的人这样做过。

我也会在文档编制中使用带引号的名称，这样它们立即会被当作模式对象的名称而不是句子中一个普通的单词。

通常，发生这类错误的原因是程序员混淆了数据元素名称和显示的标题。在传统的过程化语言中，数据文件和应用程序在同一层中；在SQL中，数据库与显示数据的前台是完全分离的。

1.1.4 实施大写规则以避免大小写区分问题

原理 区分大小写的规则随着产品的不同而不同。

标准SQL、IBM和Oracle将常规标识符转换为大写，但是不会将定界标识符转换为大写。对于Microsoft，区分大小写的规则与名称是常规的还是定界的无关。标识符取决于默认排序规则。如果默认排序规则不区分大小写，那么t等于T。如果区分大小写，则t不等于T。

6

总之，一共有两种区分大小写的问题。第一种是，如果遵循SQL标准，则定界标识符“t”与常规标识符t不同。第二种是Microsoft不遵循SQL标准。这些问题使得一个命名规范很难适用于所有情况。

例外

在这里，将根据可读性原则和版式给出一组简单的规则，但是可能还存在其他规范。

(1) 避免使用定界标识符，这样就不会有问题。

(2) IBM只使用大写。但是，这样很难阅读，就好像你还在穿孔卡系统上编程似的。

(3) 除非小写看上去很奇怪，否则Microsoft和Oracle使用小写。但是，“看上去很奇怪”这个定义一点儿都不明确。保留字有时是大写的，有时是小写的，等等。

1.2　遵循 ISO-11179 标准命名规范

ISO-11179是有关元数据较新的ISO标准，它不太容易理解。幸好，SQL程序员需要了解的那部分相当清晰和简单。真正的问题是人们在以各种方式违反它们。在下列地址可以找到ISO元数据标准委员会对数据元素规则的简要介绍。

http://metadata-standards.org/11179

7

ISO-11179标准分成6个部分：

ISO-11179-1：数据元素定义的规格说明和标准化的框架；
ISO-11179-2：数据元素的分类；
ISO-11179-3：数据元素的基本属性；
ISO-11179-4：数据表示的规则和指导原则；
ISO-11179-5：数据的命名和标识原则；
ISO-11179-6：数据元素注册。