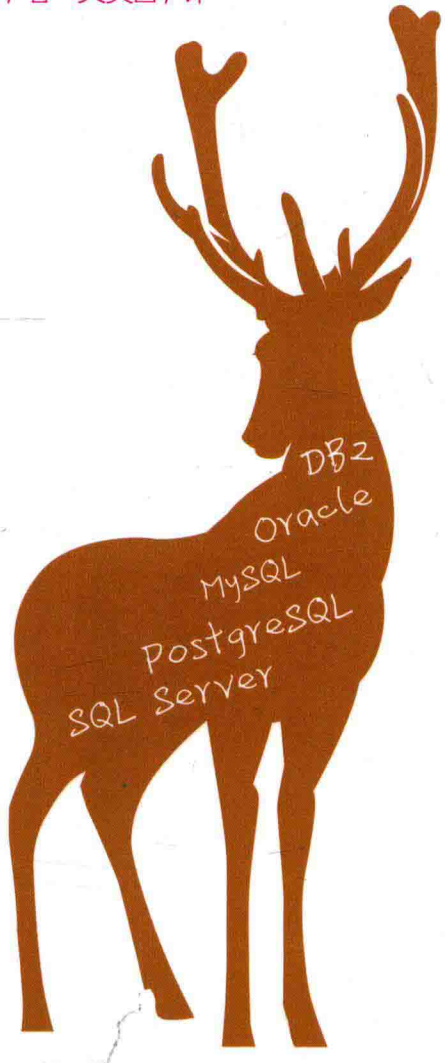
TURING
图灵程序
设计丛书

SE
SHOEISHA

SQL进阶教程

[日] MICK / 著　吴炎昌 / 译



进阶中级实用指南

挖掘SQL常见技术的新用法

基于标准SQL编写
示例程序均可下载

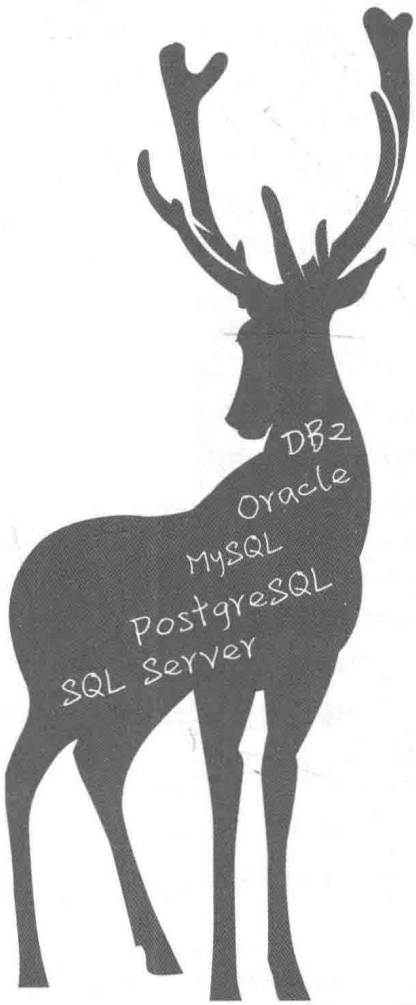
中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

TURING
图灵程序
设计丛书

SQL进阶教程

[日] MICK / 著　吴炎昌 / 译



人民邮电出版社
北　京

图书在版编目（CIP）数据

SQL进阶教程 /（日）MICK著；吴炎昌译. -- 北京：人民邮电出版社，2017.11

（图灵程序设计丛书）

ISBN 978-7-115-47052-2

Ⅰ. ①S… Ⅱ. ①M… ②吴… Ⅲ. ①关系数据库系统—教材 Ⅳ. ①TP311.138

中国版本图书馆CIP数据核字（2017）第253141号

内 容 提 要

本书是《SQL基础教程》作者MICK为志在向中级进阶的数据库工程师编写的一本SQL技能提升指南。全书可分为两部分，第一部分介绍了SQL语言不同寻常的使用技巧，带领读者从SQL常见技术，比如CASE表达式、自连接、HAVING子句、外连接、关联子查询、EXISTS……去探索新发现。这部分不仅穿插讲解了这些技巧背后的逻辑和相关知识，而且辅以丰富的示例程序，旨在帮助读者提升编程水平；第二部分着重介绍关系数据库的发展史，把实践与理论结合起来，旨在帮助读者加深对关系数据库和SQL语言的理解。此外，每节末尾均设置有练习题，并在书末提供了解答，方便读者检验自己对书中知识点的掌握程度。

本书适合具有半年以上SQL使用经验、已掌握SQL基础知识和技能、希望提升自己编程水平的读者阅读。

◆ 著　　[日]MICK

译　　吴炎昌

责任编辑　杜晓静

执行编辑　高宇涵　侯秀娟

责任印制　彭志环

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号

邮编　100164　　电子邮件　315@ptpress.com.cn

网址　http://www.ptpress.com.cn

北京鑫丰华彩印有限公司印刷

◆ 开本：800×1000　1/16

印张：19.5

字数：455千字　　2017年11月第1版

印数：1-4 000册　　2017年11月北京第1次印刷

著作权合同登记号　图字：01-2016-4463号

定价：79.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316

反盗版热线：(010)81055315

广告经营许可证：京东工商广登字20170147号

阅读本书时的注意事项

- 本书中出现的SQL语句都是尽可能按照标准SQL（SQL-92/99/2003）来写的，对于依赖具体数据库实现的地方，本书会有明确的说明。
- 按照标准SQL的要求，指定表的别名的关键字AS也应该写上，但本书省略了。这是为了避免SQL程序在Oracle数据库中出错（其他数据库里也一样，省略了就不会出错）。
- RANK、ROW_NUMBER这样的窗口函数（OLAP函数）目前还不能在MySQL数据库中运行。
- 正文里的代码在以下数据库中测试运行过。
  - Oracle 10*g*
  - SQL Server 2005
  - DB2 9.1
  - PostgreSQL 9.6
  - MySQL 5.0
- 正文里提到Oracle、MySQL等数据库而未指定版本时，请参照上述版本。
- 关于用于创建示例用表的SQL语句和示例代码等，请参考如下网站。

  http://www.ituring.com.cn/book/1813（请点击“随书下载”下载中文版相关资料）

  http://www.geocities.jp/mickindex/database/db_support_sinan.html（作者MICK的日文网站）

译者序

我曾在日本从事多年软件开发工作，工作中经常会跟各种数据库打交道，编写 SQL 代码也是常有的事情。但是对于 SQL 语言，当时也只是通过大学里的一门讲授数据库系统的课程了解了基本的语法，在工作中积累了一些实用的经验而已，并没有进行过非常深入的研究。于是我便打算找一本深入一些的书，最好是面向有一定编程经验的读者的，系统地学习一下。

后来我在书店遇见了 MICK 先生的这本书，翻看前言，尝试了他提出的检验读者水平的若干问题。非常遗憾，我只能回答出很少的几个，于是我便认为这本书正是我需要的，当场决定买下了。

几年过去，我由于个人原因回国了，工作中也不再使用日语，便想着借着业余时间翻译一些优秀的日语技术书。当图灵公司的老师问我是否有意向翻译这本书时，我立刻就答应了。当初回国时为了缩减行李，我只保留了几本日语原版的技术书，这本就是其中之一。这样一本多年前结缘、至今仍躺在我书架上的好书，当有机会将它翻译成中文版时，我实在没有什么理由放弃掉。

这本书，我认为是作者的用心之作。书中大部分内容都来自作者记录自己实践总结和日常思考的个人博客，最大的特点是理论与实践相结合，除了讲述应该怎么做，还解释了其背后的原理。全书包含两部分内容，第一部分介绍了 SQL 在使用方面的一些技巧，第二部分介绍了关系数据库相关的内容。第一部分在介绍 SQL 的技巧时，作者并没有上来就展示各种酷炫的招式，而是先以简单的问题或者例题引出将要讨论的内容，在讲解之后进一步扩展，由点及面地引出更深的话题或者背后的原理。这种由浅入深的讲述方式，符合一般的学习习惯，读者能在轻松愉悦的阅读过程中，跟着作者一起思考，自然而然地掌握相应的思考方式。第二部分在介绍关系数据库时，作者先介绍了关系数据库诞生的历史背景及其解决的问题。关系数据库已经诞生了几十年，为了让现在的读者理解当初的问题和背景，作者大量引用了关系数据库之父 E.F. Codd 和关系数据库领域权威专家 C.J. Date 的文献和言论，并按自己的理解给出了分析与解释，力图使读者体会到伟大人物们在革新技术之际的心路历程。除此之外，第二部分中作者还从逻辑学和集合论的角度讲述了 SQL 和关系模型的理论基础。该部分内容作者充分发挥了自己在相关领域的深厚积累，以深入浅出的方式进行了阐述，我认为非常精彩。

书中引用了许多经典的图书和文献，都在脚注和书末参考文献中给出了详细的出处，方便有需要的读者进一步研读。更加可贵的是，在大多数小节的末尾作者都提出了两三个精心设计的小问题，

这些问题是正文内容的扩展和延伸，非常利于读者巩固相应的知识点。而且，针对这些问题，作者也给出了详细的解答，并指出了读者容易犯的错误。

本书推荐数据库工程师、经常需要和数据库打交道的软件工程师，以及所有希望提升 SQL 水平的读者阅读。在翻译过程中，我尽力表达出原著的意图，但是由于水平有限，难免存在问题，欢迎读者批评指正。读者在阅读中有任何问题，都可以通过电子邮件和我取得联系（ensho_go@hotmail.com）。

2017 年 9 月

于北京

前言

编写本书的目的在于架起两座桥梁：一是让数据库工程师从初级向中级进阶的桥梁，旨在帮助初级工程师提升自己；二是理论（原理）和实践之间的桥梁。这里所说的“初级”，具体是指已经掌握了 SQL 的基础知识和技能，具有半年到一年左右的使用经验这种水平。

我们来做一个测试，帮助大家了解一下自己处于何种水平。下面有 10 个问题，请回答 Yes 或 No。

1. 没有在聚合函数中使用过 CASE 表达式。
2. 想象不出自连接是如何工作的。
3. 感觉 HAVING 子句不是很常用。
4. 感觉 IN 比 EXISTS 好用，所以更喜欢用 IN。
5. 听到布尔类型，脑海里浮现出的只有 true 和 false。
6. 设计表的时候不加 NOT NULL 的约束。
7. SQL 全部用大写字母或全部用小写字母来写。
8. 不能用一句话说出 GROUP BY 和 PARTITION BY 的区别。
9. 不知道 SQL 里的高阶函数的名字。
10. 试着读过 Joe Celko 的《SQL 权威指南》[①] 和《SQL 解惑（第 2 版）》[②]，但是感觉太难而没能读完（或者压根儿没有读过）。

大家的回答如何呢？如果全部都回答了 No，那很好，不要担心什么，请合上本书，立刻踏上成为一名高级工程师的道路吧（也许只有本书 3-2 节“参考文献”值得略读一下）。相反，如果一半以上都回答了 Yes，那么本书将照亮大家的前进之路——这正是编写本书的目的，相信大家读后一定会有收获。

但是，接下来要说的内容可能会让大家觉得有点前后矛盾。因为，这本书即将介绍的技术绝不是多么新潮的东西，而是遵循标准 SQL 的非常普通的技术。关于这一点，相信扫一眼目录你就会明白。CASE 表达式、自连接、HAVING 子句、外连接、关联子查询、EXISTS……这些都是数据库工程师日常工作中经常用到的技术。

---

① 原书名为 *Joe Celko's SQL for Smarties: Advanced SQL Programming*，本书共有五版。国内引进了第 4 版，书名为《SQL 权威指南（第 4 版）》，朱巍等译，人民邮电出版社，2013 年。——编者注

② 米全喜译，人民邮电出版社，2008 年 4 月。——编者注

编写本书的目的就是从新的角度把光照向这些“并没有什么特别的、谁都知道的技术”，照亮它们迄今都没有被看到的一面。相信大家读完本书时，会从那个一直以来都被认为平淡无奇的关系数据库的世界里，看到一些不一样的光辉。

下面，就让我们立刻前往博大精深的关系数据库的世界，开始探险之旅吧。

目 录

第1章 神奇的SQL

第2章 关系数据库的世界

第 1 章

神奇的SQL

1-1 CASE 表达式

▶在 SQL 里表达条件分支

CASE 表达式是 SQL 里非常重要而且使用起来非常便利的技术，我们应该学会用它来描述条件分支。本节将通过行列转换、已有数据重分组（分类）、与约束的结合使用、针对聚合结果的条件分支等例题，来介绍 CASE 表达式的用法。

写在前面

CASE 表达式是从 SQL-92 标准开始被引入的。可能因为它是相对较新的技术，所以尽管使用起来非常便利，但其真正的价值却并不怎么为人所知。很多人不用它，或者用它的简略版函数，例如 DECODE（Oracle）、IF（MySQL）等。然而，正如 Joe Celko 所说，CASE 表达式也许是 SQL-92 标准里加入的最有用的特性。如果能用好它，那么 SQL 能解决的问题就会更广泛，写法也会更加漂亮。而且，因为 CASE 表达式是不依赖于具体数据库的技术，所以可以提高 SQL 代码的可移植性。这里强烈推荐大家改用 CASE 表达式，特别是使用 DECODE 函数的 Oracle 用户[1]。

本节，我们将通过具体的例题来学习优点众多的 CASE 表达式。

注❶

DECODE 是 Oracle 用户很熟悉的函数，它有以下四个不如 CASE 表达式的地方。

- 它是 Oracle 独有的函数，所以不具有可移植性。
- 分支数最大支持 127 个（参数上限 255 个，一个分支需要 2 个参数）。
- 如果分支数增加，代码会变得非常难读。
- 表达能力较弱。具体来说，参数里不能使用谓词，也不能嵌套子查询。

CASE 表达式概述

首先我们来学习一下基本的写法，CASE 表达式有简单 CASE 表达式（simple case expression）和搜索 CASE 表达式（searched case expression）两种写法，它们分别如下所示。

■ CASE 表达式的写法

```
-- 简单 CASE 表达式
CASE sex
  WHEN '1' THEN '男'
  WHEN '2' THEN '女'
ELSE '其他' END
```

```
-- 搜索 CASE 表达式
CASE WHEN sex = '1' THEN '男'
     WHEN sex = '2' THEN '女'
ELSE '其他' END
```

这两种写法的执行结果是相同的，“sex”列（字段）如果是'1'，那么结果为男；如果是'2'，那么结果为女。简单 CASE 表达式正如其名，写法简单，但能实现的事情比较有限。简单 CASE 表达式能写的条件，搜索 CASE 表达式也能写，所以本书基本上采用搜索 CASE 表达式的写法。

我们在编写 SQL 语句的时候需要注意，在发现为真的 WHEN 子句时，CASE 表达式的真假值判断就会中止，而剩余的 WHEN 子句会被忽略。为了避免引起不必要的混乱，使用 WHEN 子句时要注意条件的排他性。

■剩余的 WHEN 子句被忽略的写法示例

```
-- 例如，这样写的话，结果里不会出现“第二”
CASE WHEN col_1 IN ('a', 'b') THEN '第一'
     WHEN col_1 IN ('a')      THEN '第二'
ELSE '其他' END
```

此外，使用 CASE 表达式的时候，还需要注意以下几点。

注意事项 1：统一各分支返回的数据类型

虽然这一点无需多言，但这里还是要强调一下：一定要注意 CASE 表达式里各个分支返回的数据类型是否一致。某个分支返回字符型，而其他分支返回数值型的写法是不正确的。

注意事项 2：不要忘了写 END

使用 CASE 表达式的时候，最容易出现的语法错误是忘记写 END。虽然忘记写时程序会返回比较容易理解的错误消息，不算多么致命的错误。但是，感觉自己写得没问题，而执行时却出错的情况大多是由这个原因引起的，所以请一定注意一下。

注意事项 3：养成写 ELSE 子句的习惯

与 END 不同，ELSE 子句是可选的，不写也不会出错。不写 ELSE 子句时，CASE 表达式的执行结果是 NULL。但是不写可能会造成“语法没有错误，结

果却不对”这种不易追查原因的麻烦，所以最好明确地写上 ELSE 子句（即便是在结果可以为 NULL 的情况下）。养成这样的习惯后，我们从代码上就可以清楚地看到这种条件下会生成 NULL，而且将来代码有修改时也能减少失误。

将已有编号方式转换为新的方式并统计

在进行非定制化统计时，我们经常会遇到将已有编号方式转换为另外一种便于分析的方式并进行统计的需求。

例如，现在有一张按照“‘1：北海道’、‘2：青森’、……、‘47：冲绳’”这种编号方式来统计都道府县❶人口的表，我们需要以东北、关东、九州等地区为单位来分组，并统计人口数量。具体来说，就是统计下表 PopTbl 中的内容，得出如右表“统计结果”所示的结果。

■ 统计数据源表 PopTbl

pref_name（县名）	population（人口）
德岛	100
香川	200
爱媛	150
高知	200
福冈	300
佐贺	100
长崎	200
东京	400
群马	50

■ 统计结果❷

地区名	人口
四国	650
九州	600
其他	450

大家会怎么实现呢？定义一个包含“地区编号”列的视图是一种做法，但是这样一来，需要添加的列的数量将等同于统计对象的编号个数，而且很难动态地修改。

而如果使用 CASE 表达式，则用如下所示的一条 SQL 语句就可以完成。为了便于理解，这里用县名（pref_name）代替编号作为 GROUP BY 的列。

```
-- 把县编号转换成地区编号 (1)
SELECT  CASE pref_name
                WHEN '德岛' THEN '四国'
                WHEN '香川' THEN '四国'
                WHEN '爱媛' THEN '四国'
                WHEN '高知' THEN '四国'
```

注❶ 日本的省级行政单位有都、道、府、县，包含一都（东京都）、二府（京都府和大阪府）、一道（北海道）和诸多的县，统称都道府县。多个较近的县被划归到一个地区，如关东地区、九州地区等，类似我国的华北地区、华南地区等概念。——译者注

注❷ 在“统计结果”这张表中，“四国”对应的是表 PopTbl 中的“德岛、香川、爱媛、高知”，“九州”对应的是表 PopTbl 中的“福冈、佐贺、长崎”。——编者注

```
                WHEN '福冈' THEN '九州'
                WHEN '佐贺' THEN '九州'
                WHEN '长崎' THEN '九州'
       ELSE '其他' END AS district,
       SUM(population)
  FROM  PopTbl
 GROUP BY CASE pref_name
                WHEN '德岛' THEN '四国'
                WHEN '香川' THEN '四国'
                WHEN '爱媛' THEN '四国'
                WHEN '高知' THEN '四国'
                WHEN '福冈' THEN '九州'
                WHEN '佐贺' THEN '九州'
                WHEN '长崎' THEN '九州'
          ELSE '其他' END;
```

这里的关键在于将 SELECT 子句里的 CASE 表达式复制到 GROUP BY 子句里。需要注意的是，如果对转换前的列“pref_name”进行 GROUP BY，就得不到正确的结果（因为这并不会引起语法错误，所以容易被忽视）。

同样地，也可以将数值按照适当的级别进行分类统计。例如，要按人口数量等级（pop_class）查询都道府县个数的时候，就可以像下面这样写 SQL 语句。

```
-- 按人口数量等级划分都道府县
SELECT  CASE WHEN population <  100 THEN '01'
             WHEN population >= 100 AND population < 200  THEN '02'
             WHEN population >= 200 AND population < 300  THEN '03'
             WHEN population >= 300 THEN '04'
       ELSE NULL END AS pop_class,
       COUNT(*) AS cnt
  FROM  PopTbl
 GROUP BY CASE WHEN population <  100 THEN '01'
               WHEN population >= 100 AND population < 200  THEN '02'
               WHEN population >= 200 AND population < 300  THEN '03'
               WHEN population >= 300 THEN '04'
          ELSE NULL END;
```

```
pop_class  cnt
---------- ----
01            1
02            3
03            3
04            2
```

这个技巧非常好用。不过，必须在 SELECT 子句和 GROUP BY 子句这两处写一样的 CASE 表达式，这有点儿麻烦。后期需要修改的时候，很容易发生只改了这一处而忘掉改另一处的失误。

所以，如果我们可以像下面这样写，那就方便多了。

```
-- 把县编号转换成地区编号（2）：将 CASE 表达式归纳到一处
SELECT  CASE pref_name
                WHEN '德岛' THEN '四国'
                WHEN '香川' THEN '四国'
                WHEN '爱媛' THEN '四国'
                WHEN '高知' THEN '四国'
                WHEN '福冈' THEN '九州'
                WHEN '佐贺' THEN '九州'
                WHEN '长崎' THEN '九州'
        ELSE '其他' END AS district,
        SUM(population)
FROM  PopTbl
GROUP BY district;
```

GROUP BY 子句里引用了 SELECT 子句中定义的别名

没错，这里的 GROUP BY 子句使用的正是 SELECT 子句里定义的列的别称——district。但是严格来说，这种写法是违反标准 SQL 的规则的。因为 GROUP BY 子句比 SELECT 语句先执行，所以在 GROUP BY 子句中引用在 SELECT 子句里定义的别称是不被允许的。事实上，在 Oracle、DB2、SQL Server 等数据库里采用这种写法时就会出错。

不过也有支持这种 SQL 语句的数据库，例如在 PostgreSQL 和 MySQL 中，这个查询语句就可以顺利执行。这是因为，这些数据库在执行查询语句时，会先对 SELECT 子句里的列表进行扫描，并对列进行计算。不过因为这是违反标准的写法，所以这里不强烈推荐大家使用。但是，这样写出来的 SQL 语句确实非常简洁，而且可读性也很好。

用一条 SQL 语句进行不同条件的统计

进行不同条件的统计是 CASE 表达式的著名用法之一。例如，我们需要往存储各县人口数量的表 PopTbl 里添加上“性别”列，然后求按性别、县名汇总的人数。具体来说，就是统计表 PopTbl2 中的数据，然后求出如表“统计结果”所示的结果。