ORACLE 数据库系统
高级应用技术

刘志斌　杜小勇　孟小峰　编著

清华大学出版社

需要全本请访问 www.ertongbook.com

(京)新登字 158 号

内 容 简 介

本书是我社出版的《ORACLE 丛书》的第三册。在前两册系统分析、讲解 ORACLE,并给出大量应用实例的基础上,本书以 ORACLE 的"高级应用"为出发点,再一次系统而有层次地分析了 ORACLE 系统应用开发中的一些较有针对性、较有难度也较为重要的技术问题。内容包括:ORACLE 系统的功能及其体系结构;主流工具(SQL * Plus,Pro * C,SQL * Forms)的重点及难点分析;解决应用开发中的有共性的复杂问题,如长正文处理、报表制作、系统性能优化等;应用实例的分析。

本书文笔流畅,针对性强,可供计算机应用系统开发人员参考,可作为 ORACLE 各类培训班的教材,也可作为高等院校计算机相关专业数据库课程的参考书。

版权所有,翻印必究。

本书封面贴有清华大学出版社激光防伪标志,无标志者不得销售。

图书在版编目(CIP)数据

ORACLE 数据库系统高级应用技术/ 刘志斌,杜小勇,孟小峰编著 .-北京:清华大学出版社,1994.4
(ORACLE 丛书)
ISBN 7-302-01450-7

Ⅰ.O… Ⅱ.①刘… ②杜…③孟… Ⅲ.数据库,ORACLE-计算机应用
Ⅳ.TP392

中国版本图书馆 CIP 数据核字(94)第 03358 号

出版者:清华大学出版社(北京清华大学校内,邮编 100084)
印刷者:北京密云胶印厂
发行者:新华书店总店北京科技发行所
开 本:787×1092 1/16 印张:20 字数:473 千字
版 次:1994 年 6 月第 1 版 1994 年 6 月第 1 次印刷
本社分类号:TP·570
印 数:0001—8000
定 价:19.00 元

前　言

本书是清华大学出版社出版的 ORACLE 系列丛书中的第三本，书名《ORACLE 数据库系统高级应用技术》反映了我们的写作意图。“应用”两字说明本书的许多素材来自应用问题，是应用中的一些共性问题。我们在进行 ORACLE 数据库使用、培训，以及参加一些鉴定和讲学中，经常有实际应用单位的同志提出许多应用中的问题和我们商讨，研究其解决方案。本书作者也或多或少地实际参加过一些项目的开发工作，对开发过程中所遇到的问题进行了多方面的抽象提炼，我们还从一些介绍实际项目的文章中分析整理出较有新意的技术手段。所以可以说本书的大多数问题是有其实际背景的。我们希望通过整理成书，奉献给读者，使其从应用中来到应用中去，为应用开发服务。“高级”两字并不单纯地表示复杂或高深，我们用这两个字想表达如下的两层意思。一是“综合”：本书涉及的问题是通常的 ORACLE 使用手册或教科书中没有的。在使用手册或教科书中，主要是为了解释每个命令或语句的使用，或者解释某个概念。而本书的写作是面向问题的，需要综合地利用我们已掌握的知识来提出解决方案。二是“系统”：有些应用问题，例如一个极其普遍的问题是 ORACLE 系统的性能问题，这种问题的解决，既与对实际问题的分析理解有关，也与 ORACLE 系统本身的特性有关，还与我们的设计开发经验有关，它涉及的因素很多，我们只有系统地分析、考察，才能有效地提出解决方案来，单靠一点“技巧”是远远不够的。

本书由于其面向问题的特点，决定了在写作时容易造成结构的零碎和松散。为了避免这一问题，我们在内容组织上下了点功夫，将内容组织成四个层次，见下图。

第一个层次是 ORACLE 系统概述,力争在一个更高的层次上、更广的范围中来讨论一下 ORACLE 数据库的功能及其体系结构,并通过对主流产品的评价,提出我们对应用开发中选用产品的一些看法。第二个层次是主流工具的重点与难点分析。准备重点对 ORACLE的 SQL 语言,Pro * C, SQL * Forms 中一些难以理解和掌握的概念和功能(难点)以及对应用开发至关重要的内容(重点)进行分析和讨论。第三个层次是对应用中一些具有共性的问题进行分析和讨论。这个层次的内容比较多,限于篇幅,我们仅选择了比较大的、也相对比较重要和困难的超长正文处理、复杂报表制作、数据库性能优化等问题展开讨论,以期有更多的读者来关心这一层次的技术问题。第四个层次想通过开发实例来介绍一些技术的综合应用。例子是典型的数据密集型的应用问题,我们侧重于对其数据库的设计过程展开讨论。

本书是系列书的第三本,因此我们假定读者已经初步学习并具体使用过 ORACLE 数据库系统,并且具有基本的数据库系统常识。本书适用于广大的高等院校学生、教师和应用开发人员。

ORACLE 系统引入国内已有较长的时间,在国内的应用面很广,限于我们的接触面不是很广泛,不一定能全面地反映这方面的经验和成果。尽管我们做了很大的努力,恐怕仍然是挂一漏万。同时,随着数据库实践的不断发展,我们相信肯定还有更多的新问题从应用中提出来,并被众多的 ORACLE 开发高手所解决。我们诚恳地希望本书能起到抛砖引玉的作用,引来诸多 ORACLE 高手与富于开创精神的清华大学出版社计算机编辑室的诸位编辑先生合作,写出更好的“高级应用开发”的书。

本书的第二章高级 SQL 技术由孟小峰同志撰写,其余章节由刘志斌、杜小勇完成。书中所有例子都经过上机验证,运行环境为微机 DOS 系统,ORACLE 的版本为第 5 版和第 6 版。为加快 ORACLE 的普及和应用水平的提高,我们愿意为大家提供书中全部例子的源程序软盘,需要的同志可以与作者直接联系(电话:(01)2574499 - 7350),也可与清华大学出版社软件部联系(2594891)。

中国人民大学信息系王珊教授审阅了本书的组织结构和内容提纲,并给作者提出了许多宝贵的指导性意见。在本书的写作过程中,清华大学出版社编辑姜峰同志始终给作者予大力的支持和鼓励。祝琳、李劲、徐玉珏等同志为本书作了大量辛苦细心的抄写和录入工作。在此谨向他们致以最衷心的感谢!

尽管我们付出了很多努力,但是因为数据库系统内容博大精深,加之我们水平有限,书中一定还有许多错误和不足,我们衷心地希望读者批评指正。作者愿为本书中的观点和错误负全部责任。

作　者

1994 年 2 月于北京

目　录

前言
第一章　ORACLE 系统概述 …… 1
1.1　ORACLE 系统的功能及其特点 …… 1
1.1.1　具有完整的数据管理功能 …… 1
1.1.2　优秀的超完备关系型产品 …… 3
1.1.3　实用的分布式数据库产品 …… 6
1.1.4　具有非常丰富的开发工具 …… 10
1.1.5　具有优良的开放性 …… 11
1.2　ORACLE 系统的体系结构 …… 12
1.2.1　数据库系统的构成及其结构 …… 12
1.2.1.1　ORACLE RDBMS 与操作系统的关系 …… 13
1.2.1.2　ORACLE RDBMS 与网络系统的关系 …… 14
1.2.1.3　ORACLE RDBMS 与高级语言的关系 …… 14
1.2.1.4　ORACLE RDBMS 与其工具的关系 …… 15
1.2.1.5　数据库应用系统 …… 15
1.2.2　ORACLE 系统进程结构 …… 16
1.2.3　ORACLE 系统的产品结构 …… 16
1.2.4　ORACLE RDBMS 的软件层次结构 …… 17
1.2.5　ORACLE RDBMS 的进程结构 …… 18
第二章　高级 SQL 技术 …… 20
2.1　SQL 语言概述 …… 20
2.1.1　SQL 数据类型、运算符与谓词 …… 21
2.1.2　数据定义语言(DDL) …… 22
2.1.3　数据操纵语言(DML) …… 23
2.1.4　数据控制语言(DCL) …… 25
2.2　连接查询 …… 25
2.2.1　相等连接和不等值连接 …… 26
2.2.2　外连接 …… 29
2.2.3　表的自身连接 …… 32
2.3　子查询 …… 32
2.3.1　不相关子查询 …… 33
2.3.2　相关子查询 …… 33
2.3.3　带有比较符的子查询 …… 35
2.3.4　带有 ANY,ALL 的子查询 …… 35
2.3.5　使用存在量词 EXISTS 的子查询 …… 37
2.3.6　多层嵌套子查询 …… 38

需要全本请访问 www.ertongbook.com

2.3.7 其他形式的子查询 …… 39
2.3.8 子查询小结 …… 40
2.4 集合查询 …… 40
2.4.1 并操作 UNION …… 42
2.4.2 交操作 INTERSECT …… 42
2.4.3 差操作 MINUS …… 43
2.5 空值的显示和处理 …… 44
2.5.1 空值查询 …… 44
2.5.2 按顺序排列有空值的行 …… 45
2.5.3 在表达式和函数里的空值 …… 46
2.5.4 空值函数 NVL …… 47
2.6 视图的建立、查询与更新 …… 48
2.6.1 视图的建立 …… 49
2.6.2 视图的查询 …… 50
2.6.3 视图的更新 …… 51
2.6.4 基表定义的改变对视图的影响 …… 52
2.7 如何写出高效率的查询语句 …… 52
2.7.1 ORACLE 系统中的优化原理 …… 53
2.7.1.1 单表查询的优化 …… 53
2.7.1.2 多表连接的优化处理 …… 55
2.7.2 查询优化实例及分析 …… 56
2.7.2.1 EXPLAIN PLAN 语句 …… 57
2.7.2.2 示例查询所用的表和索引 …… 58
2.7.2.3 举例与分析 …… 60
2.7.3 小结 …… 68
第三章 嵌入 SQL 技术 …… 69
3.1 嵌入 SQL 概念和实现原理 …… 69
3.1.1 嵌入 SQL 引入 …… 69
3.1.2 嵌入 SQL 的执行过程 …… 69
3.1.3 嵌入 SQL 形式 …… 70
3.1.4 嵌入 SQL 实现原理 …… 71
3.2 使用嵌入 SQL 完成数据库的各种基本操作 …… 75
3.2.1 数据定义、控制、操纵语句 …… 75
3.2.2 数据查询语句和游标 …… 76
3.2.3 使用 SQLCA …… 78
3.2.4 使用宿主变量数组 …… 82
3.3 高级动态 SQL 技术 …… 86
3.3.1 动态 SQL 基本概念 …… 86
3.3.1.1 动态 SQL 引入 …… 86
3.3.1.2 动态 SQL 的处理过程 …… 87
3.3.1.3 动态 SQL 的一般要求 …… 87
3.3.1.4 动态 SQL 分类 …… 88

3.3.2 前三种动态 SQL 方法 …… 88
3.3.2.1 第一种动态 SQL 方法 …… 89
3.3.2.2 第二种动态 SQL 方法 …… 90
3.3.2.3 第三种动态 SQL 方法 …… 91
3.3.3 第四种动态 SQL 方法 …… 93
3.3.3.1 前三种方法的限制 …… 93
3.3.3.2 第四种方法实现原理 …… 94
3.3.3.3 定义 SQLDA …… 96
3.3.3.4 第四种动态 SQL 方法编程基本步骤 …… 98
3.3.3.5 编程步骤详解 …… 100
3.4 应用嵌入 SQL 的复杂程序举例 …… 110
3.4.1 使用第四种动态 SQL 方法的完整程序 …… 110
3.4.2 LONG 类型数据存取程序 …… 117
第四章 SQL * Forms 设计技术 …… 125
4.1 SQL * Forms 的基本设计技术 …… 125
4.1.1 FORM 概念 …… 125
4.1.2 SQL * Forms 设计方法 …… 128
4.1.3 SQL * Forms 基本设计内容 …… 129
4.2 触发器机制剖析 …… 130
4.2.1 触发事件(触发器类型) …… 130
4.2.1.1 触发事件 …… 130
4.2.1.2 触发时间 …… 133
4.2.1.3 触发器之间的关系 …… 135
4.2.1.4 触发事件的识别与触发器的执行 …… 135
4.2.2 触发行为 …… 135
4.2.2.1 触发器命令 …… 135
4.2.2.2 功能键、功能码、键触发器 …… 142
4.2.2.3 控制机制 …… 142
4.2.2.4 触发器的执行流程 …… 144
4.2.3 触发器执行的后果 …… 145
4.2.4 触发器实现评价 …… 146
4.2.5 实现 SQL * Forms 的数据库表 …… 146
4.3 触发器设计技术 …… 153
4.3.1 触发器简单应用 …… 154
4.3.1.1 复杂的数据验证 …… 154
4.3.1.2 赋值和计算 …… 160
4.3.1.3 自动生成序列号 …… 161
4.3.1.4 数据库查询和操纵 …… 165
4.3.1.5 重定义功能键 …… 167
4.3.2 实现参照完整性 …… 171
4.3.3 块间协调 …… 174
4.4 用户出口的基本原理及编写过程 …… 181

4.4.1 用户出口概述 …… 181
4.4.2 用户出口的生成过程 …… 182
4.4.3 编写用户出口 …… 185
4.4.3.1 用户出口与一般宿主程序的区别 …… 185
4.4.3.2 参数传递 …… 186
4.4.3.3 返回值 …… 186
4.4.3.4 读写域值 …… 187
4.4.3.5 用户出口实例 …… 188
4.4.3.6 注意事项 …… 189
4.5 应用 SQL*Forms 的复杂程序举例 …… 192
4.5.1 定义显示前一组记录功能键 …… 192
4.5.2 合法域值列表窗口 …… 195
4.5.3 定义扩展功能键 F0—F9 …… 197
4.5.4 查询制表打印 …… 201
4.5.4.1 问题的提出 …… 201
4.5.4.2 实现结构 …… 202
4.5.4.3 编译连接和使用过程 …… 205
4.5.4.4 源程序清单 …… 206
第五章 超长正文处理技术 …… 226
5.1 超长正文处理问题概述 …… 226
5.1.1 数据库中的长正文问题 …… 226
5.1.2 ORACLE 对超长正文的支持和限制 …… 227
5.2 SQL*Forms 中长正文处理方法 …… 228
5.2.1 “检索型”长正文处理 …… 229
5.2.2 “编辑型”长正文处理 …… 229
5.3 实例介绍 …… 230
5.3.1 实现环境和特点 …… 230
5.3.2 实现结构 …… 230
5.3.3 程序清单 …… 234
第六章 复杂报表制作技术 …… 244
6.1 报表问题概述 …… 244
6.1.1 引言 …… 244
6.1.2 报表的一般特性 …… 244
6.1.3 报表数据准备 …… 245
6.1.4 数据格式化 …… 247
6.1.5 报表工具分析 …… 248
6.2 ORACLE 报表工具分析 …… 251
6.2.1 SQL*ReportWriter …… 251
6.2.2 SQL*Report …… 252
6.2.3 SQL*Plus …… 254
6.2.4 Pro*C …… 258
6.2.5 SQL*Calc …… 259

6.2.6 SQL*Forms …… 260
6.3 ORACLE 报表制作实用技术 …… 261
6.3.1 FORM 前端技术 …… 261
6.3.2 利用通用报表处理系统进行数据格式化 …… 262
6.3.3 设计自己的数据格式化工具 …… 262
6.3.4 查询制表 …… 263
第七章 数据库应用系统性能优化技术 …… 264
7.1 数据库应用系统性能问题 …… 264
7.1.1 应用系统的评价标准 …… 264
7.1.2 影响系统性能的主要因素 …… 264
7.1.3 提高应用系统性能的主要措施 …… 265
7.2 数据库物理设计技术 …… 266
7.2.1 确定数据存储结构 …… 266
7.2.2 确定存储空间分配 …… 269
第八章 获奖项目管理系统 GQ40 …… 274
8.1 数据库系统设计的过程、方法与工具 …… 274
8.1.1 数据库系统设计的特点 …… 274
8.1.2 数据库设计方法学 …… 275
8.1.2.1 需求分析阶段 …… 275
8.1.2.2 概念设计阶段 …… 276
8.1.2.3 逻辑设计阶段 …… 276
8.1.2.4 物理设计阶段 …… 280
8.1.2.5 实施与运行维护阶段 …… 280
8.2 系统背景描述与需求分析 …… 280
8.2.1 系统背景 …… 280
8.2.2 需求分析 …… 281
8.3 概念模型设计 …… 285
8.4 系统逻辑模型设计 …… 289
8.4.1 数据结构及约束的设计 …… 289
8.4.1.1 表的结构 …… 289
8.4.1.2 完整性约束 …… 290
8.4.2 系统模块结构的设计 …… 290
8.4.3 其他设计 …… 291
8.4.3.1 数据共享设计 …… 291
8.4.3.2 安全性设计 …… 291
8.4.3.3 界面设计 …… 292
8.4.3.4 代码设计 …… 292
8.5 物理设计阶段 …… 292
8.5.1 数据库的物理设计 …… 292
8.5.2 模块的设计 …… 293
附录 A ORACLE 第 7 版新特性简介 …… 295
A.1 ORACLE 7 支持关键使命应用的特性 …… 295

A .2 ORACLE 7 支持主动应用的特性 …… 298
A .3 ORACLE 7 支持应用集成的特性 …… 301
A .4 ORACLE 第 6 版的独有特性 …… 304
A .5 将应用升级到 ORACLE 7 …… 306
A .6 ORACLE 7 产品选件情况 …… 307
参考文献 …… 309

第一章　ORACLE 系统概述

本章将在一个较高的层次上来研究 ORACLE 系统,即把它作为一个数据库系统,甚至一个软件系统来讨论它的功能、体系结构等有关问题。希望读者在初步学习和使用了 ORACLE 数据库系统以后,通过本章的学习,能对 ORACLE 数据库系统有个更加全面、更加清晰的认识,同时也为阅读以后章节作些准备。

在我们的工作中常常有这样一种不良的倾向,即过分注重于系统的功能,常常被功能所左右。看到一个系统的功能宣传介绍的小册子,马上联想到自己的系统也正需要这样的功能,于是就采用了这个系统。这是一种功能导向的作法,有许多弊病。我们认为,合理的作法应当是问题导向的,即首先认真考察实际系统,弄清其基本的需求和问题的本质,然后研究解决问题的最佳方案,并从现有系统中选择能较好地支持这一解决方案的系统来。同样,了解一个系统,也不应单纯地了解功能,而应弄清它是解决什么问题的,采用什么方法解决的。

1.1　ORACLE 系统的功能及其特点

ORACLE 数据库系统是美国 ORACLE 公司提供的以分布式关系数据库为核心的一组软件产品。作为一个通用的数据库系统,它具有较完整的数据管理功能;作为一个关系数据库系统,它是一个十分优秀的完备关系产品;作为分布式数据库系统,它实现了简单实用的分布式处理功能;作为一个应用开发环境,它提供了一组界面友好、功能齐全的开发工具,使用户拥有一个良好的应用开发环境;作为一个现代的软件产品,它又具有很好的开放性。

下面我们就从这五大方面来详细地讨论。

1.1.1　具有完整的数据管理功能

认识 ORACLE 系统,首先要把它作为一个通用的数据库系统来认识。现在人们一谈起数据库这个词,就马上联想到它有一个复杂的管理系统软件、负责数据的存储/存取控制,还有一大堆的开发工具等等。这些固然都对,而且关于 ORACLE 系统的功能清单我们可以列出长长的一串。但是作为数据库的一个用户,我们的目的并不仅仅是沾沾自喜于会用一个新的玩意儿,而是要用它来为我们的应用服务,来提高应用的开发质量、速度和效率。我们要弄清楚为什么需要这样一些功能而不是另外一些功能。这就涉及一个问题,作为数据库系统,它要解决什么问题呢?在你的应用中是否也有这种性质的问题正好可以使用数据库去解决的呢?只有弄清了这些问题,我们才能成为软件工具的主人而不是软件工具的奴隶。

大家知道,数据库技术是随着计算机技术的发展,尤其是在数据处理领域中的广泛应

需要全本请访问 www.ertongbook.com

用而发展起来的,具有很强的实用性,它主要是为了解决应用中具有如下特征的数据管理问题。

(1) 大量。应用中的许多问题通常都需要涉及大量的数据。数据是对系统中各种对象的性质、特征、状态等的描述。将实物对象或具体概念对象抽象成一系列的数据,是应用计算机技术的关键一步,我们不可能把一个实物装入计算机,因此数据的大量性是任何一个数据处理问题,也可以说是大多数计算机应用问题的共同特征。当然这里"大量"还有个级别,至少是超过了你的计算机系统的内存容量。否则我们可以利用任何一种高级程序设计语言,在内存空间中开辟一个足够大的数组来装载数据,数据处理就可以在内存中完成,问题也就简单多了,无需数据库来帮忙。因此可以说数据库是用来管理大容量数据的。

(2) 持久。持久的意思是数据要永久地保存。这是数据库中的数据处理有别于一般的程序中的数据处理的重要区别所在。在使用普通的程序语言写成的某个程序中,数据的生命周期是该程序的一次运行期——即该程序对应进程的生命期。一旦进程终止,数据也就消失,不能重复使用。若想重复使用,则必须使用赋值语句来显式地给出数据。显然,这对于具有大量数据的情况是不可想象的。而且大多数的应用问题中都不允许程序运行结束后丢失数据。因此可以说数据库是管理持久性数据的。

(3) 共享。即多用户、多程序需要共享一份数据存储。在我们以手工方式进行信息处理时,由于人的计算能力的限制,难以作到共享一份存储,而是各人根据自己的具体需要,收集一份相对完整的数据,各自为政地完成信息的收集、分类、排序、计算、统计等处理工作。这种从工作需要出发,条块分割、各自为政的作法必然会造成同一个对象在多个场所存在其描述信息,从而极易造成信息的不一致。我们知道,信息的正确性是计算机能对解决实际问题有所帮助的根本前提。试想如果不能保证机器中的数据的基本正确性,那么你的计算机再先进、数据库系统再优良、程序编得再精巧,也只能是"南辕北辙",越走越远。不仅解决不了实际问题,还可能造成严重的危害。因此可以说数据库是管理共享信息的。共享有两个紧密相关的表现:一是多种用户要共享某一数据;二是多个用户要同时存取某一数据。

(4) 可靠。一个实际的系统对数据必须有其基本的安全可靠保证。本来在实际生活中可以通过规章制度、条款条例、甚至警卫保险等措施来保障一个系统的合理运行。现在当客观对象以数据形式存入计算机系统,成为一种计算机系统时,这种可靠性、合理性、正确性由谁来保障?如果没有这种基本的可靠保障,又有谁敢于真正地让计算机系统在实际工作中发挥作用? 因此,数据库系统必须提供对数据的可靠性的支持。

至此可以说,需要使用数据库来解决的实际问题应当是那些其数据具有明显的大量、持久、共享和可靠特征的问题,这可以作为我们是否应当考虑选用数据库系统解决某一问题的标准。

数据库管理对象的特征在很大程度上也决定了一个好的数据库系统所应具有的功能。例如:

(1) 数据量大。要求数据库系统具有对外存储器上的数据的管理能力,包括提供可选的、有效的文件存储结构以及对存储数据的有效存取路径,以保证系统有较好的性能。

(2) 永久保存。要求系统能提供处理临时性数据的程序接口,包括将数据库中的数据传给程序变量以及将存储在变量中的内存数据传到数据库中保存起来使之永久化,以便为其他应用和用户使用。由于目前大多数的数据库系统是关系数据库系统,其数据操纵语言是一次一个集合式的描述性语言,而高级语言大多是以一次一个记录方式工作的过程性语言。这二种系统之间的接口设施,我们已经从以前的教材中看到了,并不是一件容易的事,需要精心设计,精心使用。

(3) 多用户共享。我们说过数据共享有两层含义。一是多个用户共享同一份数据,但是同一个系统中的数据并非都是同等重要的,对不同的用户应当有不同的机密性。这就要求系统在提供信息共享的同时,要实施信息的安全性控制。做到既最大限度地共享,又各取自己所需信息(“最大共享、知必所需”)。数据共享的另一个含义是多用户同时存取同一数据,我们在“操作系统”课程学习中,甚至在日常生活中早已知道,对共享资源的竞争必须加以严格的控制,否则将造成极其严重的后果。因此数据库要提供对多用户并发数据存取的控制,保证操作结果的合理性。

(4) 可靠。要求系统具有抗干扰、抗故障能力。例如对一个违背合理性的数据操作请求能够予以拒绝;出现系统的软、硬件故障时能够尽快恢复等等。

人们在长期的数据库研究与实践中已经总结出了管理大量、持久、共享、可靠数据的系统所应具备的完整功能主要包括:

- 外存数据的存储/存取功能
- 数据对象的定义与操纵功能
- 并发控制
- 安全性控制
- 完整性控制
- 故障恢复
- 与高级语言接口

ORACLE 数据库作为一个通用的数据库系统,可以说具有上述的全部功能,是一个功能完整的数据库系统。

1.1.2 优秀的超完备关系型产品

什么是关系系统? 简单地说就是支持关系模型的系统,即支持关系模型的所有特征。那么这些特征具体是些什么呢?关系数据库系统的创始人 E. F. Codd 博士,曾提出衡量一个数据库是否为关系型产品的十二条基本准则,其基本内容如下:

准则 0: 一个关系型的 DBMS 必须能完全通过它的关系能力来管理数据库。也就是说不管你的系统提供了多少“功能强大”的数据管理能力(可能是一些非关系的成分),必须至少可以在关系层次上(即以一次一个集合的方式)操纵和管理数据库。准则 0 是关系系统的基本前提,是下面十二条准则的基础。

准则 1: 信息准则。关系型 DBMS 的所有信息都应在逻辑一级上用一种方法,即表中的值显式地表示。不仅用户的所有数据是以表中元组形式表示的,而且系统的所有数据,

如描述表名、列名、用户权限等内容的信息,也是用表(即数据字典)中的值来表示的。

准则 2: 保证访问准则。仅依靠表名、主码和列名的组合,就能保证以逻辑方式访问关系数据库中的每个数据项。这说明关系系统中至少应有一种访问方式,能够访问数据库中的每个单元,而且独立于关系数据库中的具体物理结构。

准则 3: 空值系统化处理准则。不能用一个特殊的具体值来替代空值。

准则 4: 基于关系模型的动态联机数据字典准则。授权用户可以像使用普通数据库表那样使用数据字典,而且可以动态地随时查询字典,了解系统状态。

准则 5: 统一的数据子语言准则,系统至少应提供一种统一的、语法严格的语言,支持以下功能:

· 数据定义
· 数据操纵
· 完整性约束定义
· 授权
· 事务处理功能

准则 6: 视图更新准则。所有理论上可更新的视图应该允许更新。

准则 7: 高级数据操纵准则。能支持在关系级别上(即将关系作为一个单一的操作对象)进行数据的插入、删除和修改操作。

准则 8: 数据物理独立性准则。数据库的数据在存储表示或存取方法上的任何变化,不应影响到应用程序和终端活动。

准则 9: 数据逻辑独立性准则。当对基本关系进行理论上信息不受损的任何变化时,应用程序和终端活动应当保持逻辑上的不变性。

准则 10: 数据完整性的独立性准则。关系数据库的完整性约束条件必须是用数据子语言定义并存放在系统数据字典中的,而不是在应用程序中加以定义的。

准则 11: 分布独立性。系统无论是在最初的数据分布还是以后的数据的重新分布时,都能使其应用程序和终端活动保持逻辑不变性。

准则 12: 无破坏性准则。若系统具有低级(即一次一个记录)的数据语言,那么它不应违背或绕过依照完整性准则已定义的完整性约束。

(关于这十二条基本准则的详细内容可参见萨师煊、王珊所著《数据库系统概论》(第二版),高等教育出版社,1991 年 4 月,第 148～151 页。)

读者不难发现,关系系统的有些要求是十分高的,要全部实现困难很大。事实上到目前为止还没有一个系统是完全满足这些要求的,只能说是一个追求的目标。而且实际上这些准则的重要性并不一样。因此各系统在自己的实现中根据各自的考虑对这些准则有不同程度的体现。下面我们来分析一下 ORACLE 系统对上述准则的贴近程度。

(1) ORACLE 数据库使用表这样一种单一的数据结构来表示和组织所有的对象,所有的信息(包括用户信息和系统信息)都使用表中的值来表示。因此它满足“信息准则”。

(2) ORACLE 数据库使用 SQL 语言以完全逻辑的方式访问数据库中的每个数据项。因此它满足“保证访问准则”。

(3) ORACLE 数据库支持空值的概念,而且通过空值不予存储这一作法,使之与任何一个实际可能的特殊值相区别,而且对所有操作都系统地考虑了空值的影响,这是一系统化的处理方法。因此 ORACLE 系统满足“空值系统化处理”准则。

(4) ORACLE 数据库的数据字典是基于关系模型的主动数据字典。授权用户可以在其权限范围内,使用 SQL 语言像访问普通用户表一样访问数据字典。因此它满足有关数据字典的准则 4。

(5) ORACLE 提供了标准 SQL 语言,而且在标准 SQL 之上还进行了一系列有益的扩充。在同一个 SQL 语言中可以完成:数据定义,数据操纵,完整性约束、授权和事务的开始、提交、滚回等处理功能。尽管完整性约束的定义在其支持强度上还不够,但其 6 版、7 版已逐步对此作了加强。因此我们也可以说 ORACLE 系统是具有统一的数据子语言的。

(6) ORACLE 支持视图这一概念。但是由于判断什么是“理论上可更新的视图”目前还没有有效的办法,存在着困难。所以 ORACLE 只是允许对能比较容易地判断为可更新视图的行列子集视图进行更新。所谓行列子集视图是指从单个表中导出并且只是去掉了基本表的某些行和列(视图中必须包括基本表的码字段及说明为 NOTNULL 的字段)。因此 ORACLE 对“视图更新准则”只提供有限度的支持。这种支持的有限性也必然影响到“数据逻辑独立性准则”的实现。

(7) ORACLE 数据库使用的 SQL 语言是界于关系代数和关系演算之间的一种语言,它完全在关系层次上实施数据操纵。只要求用户提出“干什么”,而无需指出“怎么干”。用户不必指出使用什么存取路径,存取策略完全由系统决定。这不仅给用户带来了很大的方便,使他们可以集中精力去思考应用所要解决的问题上,而且给系统提供了很大的余地来进行查询的优化。在 ORACLE 5 版中 ORACLE 已将人工智能的有关技术收入查询优化,通过给存取路径打分的方法来选择最佳存取路径,规划最佳处理策略。在 ORACLE 6 版中又进一步将一些系统状态信息考虑在优化因素中,使得优化器更加精巧有效。ORACLE 7 版可以自动利用数据库对像的统计信息进行查询优化。事实上,追求查询优化器的精巧、有效一直是 ORACLE 数据库努力的一个重要目标。因此 ORACLE 系统满足“高级数据操纵准则”。

(8) ORACLE 系统支持三级模式结构。因此满足“数据物理独立性准则”。

(9) ORACLE 系统对数据逻辑独立性的支持是有限度的,数据的逻辑独立性主要是通过视图机制来实现的。而我们知道 ORACLE 对视图的支持是有限度的,这就不能保证系统具有完全的数据逻辑独立性。举例来说,当我们需要将职工表 EMP 分解成两个子表 EMP1, EMP2,它们分别记载着职工的部分描述信息(注:这种分解在许多应用中是合理的,例如将稳定不变的部分与频繁变化的部分分开成两个表,对性能会有很大的贡献),当然这两个表的码是一样的,都是职工号。显然这种分解是不会丢失任何信息的。但是对于原来建立在 EMP 上的视图,现在需要修改成 EMP1 和 EMP2 这两个表连接操作的结果。显然 ORACLE 是不允许在新的视图上作更新操作的,这便破坏了数据逻辑独立性准则。所以说 ORACLE 对数据逻辑独立性的支持是有限度的。

(10) 所谓完整性条件,即为保证系统中的数据的正确合理性而预先定义的一组约束条件。主要有两类:一是关系模型固有的完整性条件,包括实体完整性约束和参照完整性

约束;另一类是应用有关的完整性条件。ORACLE 5 版对数据完整性的支持比较差,并不直接支持实体完整性和参照完整性。实体完整性是通过非空(NOT NULL)属性字段和唯一性索引(UNIQUE INDEX)来间接实现的。如果从使用高级数据库语言来定义完整性条件这一角度看 ORACLE 5 版,它不具备数据完整性的独立性。ORACLE 6 版从此作了较大的改进,允许在 SQL 语言中定义主码字段(从而自动维护实体完整性),也允许定义参照完整性和一些描述性的应用完整条件。但是除了主码的实体完整性以外,其他完整性条件并未真正实施。ORACLE 7 版真正完全实现了实体完整性和参照完整性。

(11) ORACLE 数据库自 5.1 版之后提供了分布式处理功能,而且可以平滑地从集中式系统过渡到分布式系统。在 ORACLE 系统中我们可以在 SQL* Plus 中定义一个称之为 DATABASE LINKS 的系统对像来自由地建立与远程数据库的联系(当然需要有 SQL* Net 等软件及通讯软件的支持)。ORACLE 数据库的分布式方案对于普通用户而言是具有完全分布透明性的。但是 ORACLE 的第 5.1,第 6.0 版仍然局限在分布式查询处理功能上,不支持分布式更新(但这并不意味着 ORACLE 不能作远程更新)。这限制了它在许多应用场合的使用。ORACLE 7 版以两段提交机制实现了分布式更新的功能。

(12) ORACLE 只提供唯一的 SQL 语言与数据库接口。任何程序或工具想存取数据库都必须使用 SQL 语言(尽管 SQL 有多种使用方式)。因此自然地保证了 ORACLE 系统遵循"无破坏性准则"。

通过上面的分析,可以看到 ORACLE 数据库系统尽管没有达到上述全部十二条准则的要求,但是除了像数据逻辑独立性,数据完整性的独立性等少数几条准则外,大多数的准则都能得到较好的满足。事实上,E. F. Codd 在比较评价了最著名的几种 DBMS 产品与关系模型的贴近度后,指出 ORACLE 是对 12 项基本准则贴近度最高的关系 DBMS 产品。E. F. Codd 的这一权威性评价无疑是对 ORACLE 系统关系功能的最好的肯定。

既然现存的关系产品无一达到上述 Codd 标准,因此它只能算是一种理想境界(我们称为全关系系统)。为了比较关系系统,我们把目光放得稍低一些,讨论一下关系系统对关系模型的支持级别。我们知道关系数据模型包括数据结构、操作和完整性约束这三个方面。我们可以根据系统对这三个方面的支持程度不同,将关系系统做一分类(见表 1.1)。

ORACLE 不仅满足完备关系系统的全部要求,而且在许多方面还大大超过了,是界于关系完备系统与全关系系统之间的关系产品,因此我们可称之为超完备关系系统。

1.1.3 实用的分布式数据库产品

ORACLE 数据库自第 5 版起就开始提供分布式处理功能,是世界上第一个真正具有分布式处理功能的关系数据库产品。尽管一开始仅提供了分布式查询能力,但在其后继的产品中已逐渐增强了这方面的功能,ORACLE 第 7 版已具有较完善的分布式数据库功能。

对于什么是分布式数据库,人们常常存在一些模糊甚至是不正确的认识,以为只要是地理上是分散的,又有远程结点间的相互访问就可以称得上是分布式数据库的应用问题了,就必须采用分布式数据库技术来解决问题。例如某总局有若干个地理位置分散的分

表 1.1　关系系统的划分

模型 / 分类	数据结构	数据操作	完整性约束	实例
表式系统	关系	——	——	dBASE
最小关系系统	关系	支持选择、投影、连接操作，与存取路径无关	——	许多微机数据库系统
完备关系系统	关系	支持所有关系代数操作(集合运算、关系运算)	——	DB2, ORACLE SYBASE ……
全关系系统	关系	支持所有关系代数操作(集合运算、关系运算)	实体完整性 参照完整性 应用有关完整性	——

局，各分局在工作性质上有很强的独立性。平时的工作多半在自己的分局范围内解决，但各分局不时地需要从总局查询一些信息。由于该应用问题中存在地理分散特性和信息远程访问特性，自然地想到应当用分布式数据库技术。但是实际上我们可以用更加简单的方案来解决这一问题。又比如某电厂有些处室，他们已有各自的系统在运行，现在总厂决策层(厂长、总工们)希望动态快速地掌握一些全局信息，为此决定建立一个信息系统从各处室中收集必要的信息。在这个问题中既有地理分散性，也有全局性应用，因此也自然想到采用分布式数据库技术，而事实上我们也可以用更加简单的方案加以解决。诚然，上述问题都可以采用分布式数据库技术来加以解决，但是这多少有点“杀鸡用牛刀”的滋味，无论在资金、实现难度等各方面都是不合算的。

因此我们觉得也有必要先弄清分布式数据库所要解决问题的特征，然后再谈谈 ORACLE 分布式数据库是采用怎样的方案和技术来解决这些问题的，它具有些什么功能。

分布式数据库除了具有通常的数据管理问题的一切特征外，还有以下两个显著的特征:

(1) 数据分布。系统要管理的数据是分布在不同的场地的，“分布”这个词不同于分散，它有两层意思。一是物理上的分布，即数据在物理上是分散在计算机网络的不同结点(场地)上的；一是逻辑上的集中，即数据在逻辑上相互有语义上的联系，是属于一个系统的，而不仅仅是各场地的数据的集中。数据分布特征要求分布式数据库系统一方面要有通信网络技术的支持，要与有关通信协议接口；另一方面要能在逻辑上以统一的观点看待各场地的数据，并向用户屏蔽有关物理场地的信息(即所谓分布独立性要求)。

值得注意的是随着多处理机技术的迅速发展，可以相信基于多处理机系统的数据库技术也会相继出现(事实上 ORACLE 已有相应产品)。对于不共享存储器结构的多处理

需要全本请访问 www.ertongbook.com