Oracle技术丛书

Oracle 11g

Pro*C/C++ 编程艺术

王海凤　雷俊义　谷睿哲　王军杰　等编著

中国水利水电出版社
www.waterpub.com.cn

Oracle 技术丛书

Oracle 11g Pro*C/C++编程艺术

王海凤　雷俊义　谷睿哲　王军杰　等编著

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

本书为应用开发人员提供了使用 Pro*C/C++开发数据库应用的方法。针对各种不同类型的 Pro*C/C++应用，本书都提供了非常详细、具体的开发方法，并且为读者提供了大量的示例程序。本书不仅介绍了各种类型 Pro*C/C++应用的开发方法，而且还介绍了 Oracle 11*g* 在 Pro*C/C++方面所提供的各种新特征，包括使用大纲固定执行计划、DB2 数组插入和数组提取、隐含缓冲区插入、动态 SQL 语句缓存等。通过学习本书，读者可以快速掌握使用 Pro*C/C++开发数据库应用的方法。

图书在版编目（CIP）数据

Oracle 11*g* Pro*C/C++编程艺术 / 王海凤等编著. —北京：中国水利水电出版社，2009
（Oracle 技术丛书）
ISBN 978-7-5084-6616-3

Ⅰ. O… Ⅱ. 王… Ⅲ. ①关系数据库-数据库管理系统，Oracle 11*g*-程序设计②C 语言-程序设计 Ⅳ. TP311.138
TP312

中国版本图书馆 CIP 数据核字（2009）第 113127 号

书　　名	Oracle 技术丛书 Oracle 11*g* Pro*C/C++编程艺术
作　　者	王海凤　雷俊义　谷睿哲　王军杰　等编著
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路 1 号 D 座　100038） 网址：www.waterpub.com.cn E-mail：sales@waterpub.com.cn 电话：（010）68367658（营销中心）
经　　售	北京科水图书销售中心（零售） 电话：（010）88383994、63202643 全国各地新华书店和相关出版物销售网点
排　　版	北京零视点图文设计有限公司
印　　刷	北京纪元彩艺印刷有限公司
规　　格	184mm×260mm　16 开本　24.5 印张　643 千字
版　　次	2009 年 7 月第 1 版　2009 年 7 月第 1 次印刷
印　　数	0001—3000 册
定　　价	54.00 元

前　言

Pro*C/C++是 Oracle 公司提供的预编译开发工具，它使开发人员可以在 C/C++语言中直接内嵌 SQL 语句和 PL/SQL 块，从而降低了应用开发的难度。

当使用 C/C++语言开发 Oracle 数据库应用时，读者可以选择两种开发方法，第一种方法是使用 OCI（Oracle Call Interface）函数，第二种方法是使用 Pro*C/C++。当使用 OCI 开发应用时，开发人员除了需要掌握 C/C++、SQL 和 PL/SQL 外，还必须掌握大量 OCI 函数的作用和使用方法；而当使用 Pro*C/C++开发应用程序时，开发人员只需要掌握 C/C++、SQL 和 PL/SQL。综合以上原因，建议采用 Pro*C/C++。

本书非常系统、具体地为读者提供了开发 Pro*C/C++应用程序的方法和步骤，并且针对每部分内容都提供了完整的程序示例。通过学习本书，读者可以快速地掌握使用 Pro*C/C++开发应用程序的方法。

目标

在学习本书之后，读者应该达到以下目标：

- 掌握 Pro*C/C++各种相关工具的使用方法。
- 掌握内嵌 SQL 和 PL/SQL 块的方法。
- 掌握开发动态 SQL 应用、LOB 应用、对象应用以及集合应用的方法。

读者对象

本书是专门为 Oracle 应用开发人员所提供的编程指南。本书不仅为应用开发人员提供了内嵌 SQL 语句和 PL/SQL 块的方法，还介绍了动态 SQL 应用、LOB 应用、对象类型应用、集合应用的开发方法。另外，应用开发人员还可以掌握 Oracle11*g* 的新特征（使用大纲固定执行计划、DB2 数组插入和数组提取、隐含缓冲区插入、动态 SQL 语句缓存）。

本书内容及特点

- 第 1 章：Pro*C/C++及其相关工具。介绍预编译工具 proc、对象类型转换工具 ott、Net Configuration Assistant、SQL*Plus。以及 Visual C++工具的作用和使用方法。
- 第 2 章：Pro*C/C++基础。介绍 Pro*C/C++编程的指导方针和编程思路。
- 第 3 章：连接到数据库。介绍建立单个连接、使用自动连接、建立并发连接以及使用数据库链的方法。
- 第 4 章：内嵌 SQL 并使用宿主变量。本章不仅介绍宿主变量和指示变量的作用和使用方法，还介绍嵌入各种 SQL 语句的方法。另外，本章还介绍 Oracle 11*g* 的新特征——使用大纲固定执行计划。
- 第 5 章：错误处理。介绍 ANSI 错误处理方法（SQLCODE 和 SQLSTATE）、Oracle 错误处理方法（SQLCA 结构），以及使用 ORACA 结构诊断应用性能的方法。
- 第 6 章：处理字符数据。介绍预编译选项 CHAR_MAP 的不同设置对字符数据的影响，

还介绍使用 VARCHAR 变量处理字符数据的方法。

- 第 7 章：等同数据类型。介绍内部数据类型和外部数据类型的作用和区别，还介绍等同宿主变量和等同自定义类型的作用和使用方法。
- 第 8 章：使用宿主结构和指示结构。介绍宿主结构和指示结构的作用和使用方法。
- 第 9 章：使用宿主数组和指示数组。介绍宿主数组和指示数组的作用和使用方法，还介绍 Oracle 11*g* 的新特征——DB2 数组插入、隐含缓冲区插入。
- 第 10 章：使用结构数组。介绍宿主结构数组和指示结构数组的作用和使用方法。
- 第 11 章：使用游标。介绍非滚动游标、滚动游标、使用游标更新和删除数据的作用和使用方法，还介绍 Oracle 11*g* 的新特征——DB2 数组提取。
- 第 12 章：动态 SQL。介绍动态 SQL 方法一、方法二以及方法三的作用及使用方法，还介绍 Oracle 11*g* 的新特征——动态 SQL 语句缓存。
- 第 13 章：高级动态 SQL。介绍 Oracle 动态 SQL 方法四和 ANSI 动态 SQL 方法四的作用和使用方法。
- 第 14 章：开发多线程应用。不仅介绍多线程应用的作用，而且介绍共享上下文多线程应用、专用上下文多线程应用以及连接池应用的作用和开发方法。
- 第 15 章：开发 C++应用。介绍 Pro*C/C++所提供的 C++支持以及开发 C++应用的方法。
- 第 16 章：开发 LOB 应用。介绍 LOB 语句，以及使用 LOB 语句开发各种 LOB 应用的方法。
- 第 17 章：开发对象应用。介绍内嵌对象语句，以及使用联合接口和导航接口开发对象应用的方法。
- 第 18 章：开发集合应用。介绍集合语句，以及使用集合语句开发集合应用的方法。
- 第 19 章：内嵌 PL/SQL。介绍内嵌 PL/SQL 块、使用游标变量、建立和调用各种子程序的方法。
- 第 20 章：利用 PL/SQL 的强大功能。介绍使用 DBMS_LOB 包开发 LOB 应用、使用对象类型 ORDAudio 开发音频应用、使用对象类型 ORDVideo 开发视频应用、使用对象类型 ORDImage 开发图像应用、使用对象类型 ORDDoc 开发文献应用的方法。
- 第 21 章：开发 Windows API 应用。介绍使用 Pro*C/C++和 Visual C++开发 GUI 数据库应用的方法。

本书主要由王海凤、雷俊义、谷睿哲、王军杰等编著，王海凤组织编写第 1～6 章，雷俊义组织编写第 7～11 章，谷睿哲组织编写第 12～16 章，王军杰组织编写第 17～21 章。参加本书编写的其他人员还有王海亮、刘喜泉、宋和文、蒲建军、张建平、李新国、冯国庆、王乐天、武长毅、王宏斌、孙刚、刘瑞光、刘二乐、刘云松、赵亚军、刘洁瑛、张立民、郑忠、石磊、卫宝玉、王宝众、尹向民、王忠杰、杨举贤、高伟、梁师梅、王海霞、马新宇、尹文忠、张磊、李学刚、陈胜、徐星、姜大庆、张俊平、杨美霞、陈强、雷俊义、魏铁军、杜建云、赵建良、王文俊、富晓滨、赵新宪等。

本工作室人员具有丰富的 Oracle 应用开发、培训和技术支持经验，曾经为财政、电信、油田、银行、社保、证券期货、海关、教育等各种客户进行过 Oracle 培训，并且获得用户的一致好评。如果您有 Oracle 培训和技术支持需求，欢迎来电来函与我们联系。

由于时间紧迫和编者水平有限，书中难免有不足和疏漏之处，敬请广大读者批评指正。

编著者相关书籍

《Oracle 11*g* Pro*C/C++编程艺术》
《Oracle 11*g* 快速入门》
《Oracle 11*g* 管理备份恢复从入门到精通》
《Oracle 11*g* SQL 和 PL/SQL 从入门到精通》
《精通 Oracle 10*g* SQL 和 PL/SQL》
《使用 Oracle 10*g* Forms Builder 快速开发 Web 数据库应用》
《精通 Oracle 10*g* Pro*C/C++编程》
《Oracle 10*g* 快速入门》
《精通 Oracle 10*g* 备份与恢复》
《精通 Oracle 10*g* 系统管理》
《精通 Oracle 10*g* PL/SQL 编程》
《Oracle 9*i* 快速入门》
《Oracle 9*i* Pro*C/C++编程指南》
《Oracle 9*i* 系统管理培训教程》

编著者
2009 年 6 月
联系方式：0471-2210753
电子邮箱：whl88321@163.com
whl88321@21cn.com

目　　录

第 1 章　Pro*C/C++及其相关工具

Pro*C/C++是 Oracle 为应用开发人员所提供的预编译工具。除了该预编译工具之外，Oracle 还提供了 Pro*COBOL、Pro*FORTRAN、Pro*Ada 等 Pro*系列工具。通过这些预编译工具，Oracle 允许应用开发人员在高级语言中内嵌 SQL 语句和 PL/SQL 块，从而简化了数据库应用程序的开发难度。其中，Pro*C/C++用于在 C/C++语言中内嵌 SQL 和 PL/SQL 块，Pro*COBOL 用于在 COBOL 语言中内嵌 SQL 和 PL/SQL 块，Pro*FORTRAN 用于在 FORTRAN 语言中内嵌 SQL 和 PL/SQL 块，Pro*Ada 用于在 Ada 语言中内嵌 SQL 和 PL/SQL 块。本章介绍 Pro*C/C++及其相关工具的作用和使用方法，在学习本章之后应该完成以下任务：

- 掌握使用 proc 预编译 pc 源程序的方法。
- 掌握使用 ott 转换对象类型为 C 结构的方法。
- 掌握使用 Net Configuration Assistant 配置网络服务名的方法。
- 掌握使用 SQL*Plus 执行 SQL 和 PL/SQL 块的方法。
- 掌握使用 Microsoft Visual C++ 6.0 在 Windows 平台上建立控制台应用的方法。

1.1　Pro*C/C++简介

使用 C/C++开发 Oracle 应用程序有两种方法：使用 OCI（Oracle Call Interface）和使用 Pro*C/C++。当使用 OCI 开发应用程序时，应用开发人员不仅需要掌握 C/C++语言、SQL 和 PL/SQL，还必须掌握 Oracle 所提供的大量 OCI 函数；而当使用 Pro*C/C++开发应用程序时，应用开发人员只要掌握 C/C++语言，并能够熟练应用 SQL 和 PL/SQL 块就可以了。因为 Pro*C/C++比 OCI 更加简单，所以建议开发人员使用 Pro*C/C++。

当开发 Pro*C/C++应用程序时，第一步是编写 pc 源程序（包含内嵌 SQL 和 PL/SQL 块）；第二步是使用预编译工具 proc 将内嵌 SQL 语句和 PL/SQL 块转换为对 Oracle 运行库函数（SQLLIB）的调用，并生成 C/C++源代码文件（*.c 或*.cpp）；第三步是使用 C/C++编译工具编译 C/C++源文件，生成目标文件；第四步是使用 C/C++链接工具生成可执行文件；在生成可执行文件之后，应用开发人员可以运行该可执行文件，并完成应用程序的最终开发过程。具体步骤如图 1-1 所示。

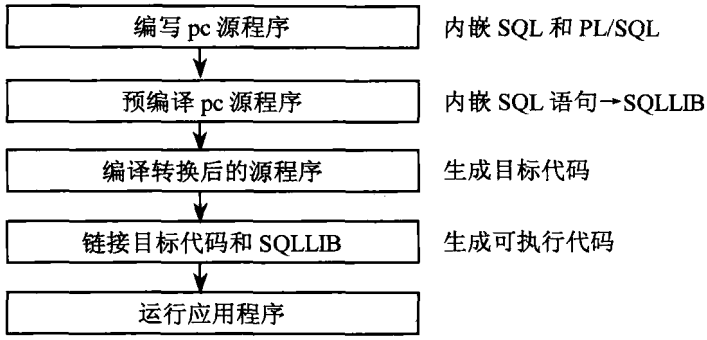

图 1-1　开发 Pro*C/C++应用程序的步骤

1. SQL

SQL 是关系数据库的基本操作语言，它是应用程序与数据库进行交互操作的接口。该语言将数据查询、数据操纵、数据定义和数据控制功能集于一体，从而使得应用开发人员、数据库管理员、最终用户都可以通过 SQL 语言对数据库进行操作。在 Pro*C/C++中可以嵌入各种 SQL 语句：

- 数据查询语言（SELECT 语句）：用于检索数据库数据。在所有 SQL 语句中，SELECT 语句的功能和语法最复杂、最灵活。
- 数据操纵语言（Data Manipulation Language）：用于改变数据库数据，它包括 INSERT、UPDATE 和 DELETE 三种语句。INSERT 语句用于将数据插入到数据库中，UPDATE 语句用于更新数据库数据，DELETE 语句用于删除数据库数据。
- 数据定义语言（Data Definition Language）：用于建立、修改和删除数据库对象。例如使用 CREATE TABLE 可以建表；使用 ALTER TABLE 可以修改表结构；使用 DROP TABLE 可以删除表。读者需要注意，DDL 语句会自动提交事务。
- 数据控制语言（Data Control Language）：用于执行权限授予和收回操作。例如，使用 GRANT 命令可以将权限授予用户，使用 REVOKE 命令可以收回用户的权限。读者需要注意，DCL 语句会自动提交事务。
- 事务控制语句（Transactional Control Statement）：用于维护数据的一致性，它包括 COMMIT、ROLLBACK 和 SAVEPOINT 三种语句。COMMIT 语句用于提交事务，ROLLBACK 语句用于回退事务，SAVEPOINT 语句用于设置保存点。

2. PL/SQL

PL/SQL 是 Oracle 在标准 SQL 语言上的过程性扩展，它支持过程结构、变量定义以及错误处理等，在运行 Oracle 的任何平台中都可以使用 PL/SQL。当编写 pc 源程序时，也可以将 PL/SQL 块内嵌到 C/C++程序中，其最大优点是可以降低网络开销、提高应用程序性能。当应用程序访问 RDBMS 时，如果不使用 PL/SQL，那么每次只能发送单条 SQL 语句，如图 1-2 所示。

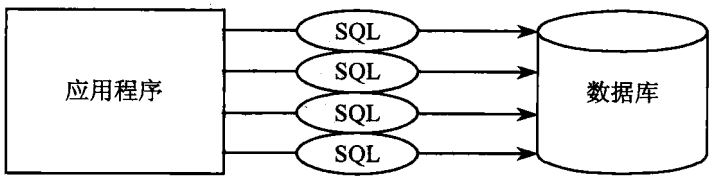

图 1-2 使用 SQL 访问数据库

通过使用 PL/SQL 块，可以将多条 SQL 语句组织到同一个 PL/SQL 块中，从而大大降低网络开销，进而提高应用性能，如图 1-3 所示。

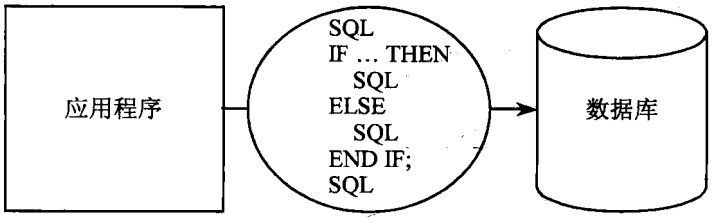

图 1-3 使用 PL/SQL 访问数据库

3. pc 源程序示例

当编写 pc 源程序时，如果嵌入 SQL 代码，那么可以使用 EXEC SQL 语句；如果嵌入 PL/SQL 块，那么可以使用 EXEC SQL EXECUTE 语句。读者需要注意，因为本书示例都是基于数据库用户 scott，所以在建立数据库时必须解锁 scott 账户。pc 源程序（demo01.pc）示例 1-1 如下：

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sqlca.h>
void    main()
{
  char username[10],password[10],server[10];
  char name[11],title[10];
  float salary;
  printf("输入用户名：");
  gets(username);
  printf("输入口令：");
  gets(password);
  printf("输入网络服务名：");
  gets(server);
  /* 内嵌连接语句 */
  EXEC SQL CONNECT :username IDENTIFIED BY :password USING :server;
  printf("请输入雇员名：");
  gets(name);
  /* 内嵌查询语句 */
  EXEC SQL SELECT sal,job INTO :salary,:title FROM emp
    WHERE UPPER(ename)=UPPER(:name);
  printf("岗位：%s,工资：%6.2f\n",title,salary);
  /* 内嵌提交事务，断开连接 */
  EXEC SQL COMMIT RELEASE;
}
```

示例 1-1　示例程序 demo01.pc 源代码

1.2　预编译工具 proc

因为 pc 源程序包含了内嵌的 SQL 语句和 PL/SQL 块，所以在编译和链接之前必须使用 proc 工具进行预编译。语法如下：

proc [OPTION_NAME=value]　[OPTION_NAME=value] ...

如上所示，OPTION_NAME 用于指定预编译选项，value 用于指定选项值。使用 proc 工具有三种方法：①在命令行直接指定预编译选项；②使用系统配置文件 pcscfg.cfg；③使用用户配置文件。读者需要注意，为了预编译 pc 源程序，必须确保已经安装了 C/C++工具和 Oracle。

1. 在命令行直接指定预编译选项

当预编译 pc 源程序时，至少需要提供 iname、parse 和 include 三个预编译选项。其中，iname 用于指定 pc 源程序名，parse 用于指定解析方法，include 用于指定头文件位置。示例 1-2 如下：

```
proc iname=d:\demo\demo01.pc parse=full
    include=%ORACLE_HOME%\precomp\public
    include="C:\Program Files\Microsoft Visual Studio\VC98\Include"
```

示例 1-2 在命令行直接预编译 demo01.pc

如上所示，环境变量 ORACLE_HOME 为 Oracle 软件安装路径。在预编译了 pc 源程序 demo01.pc 之后，会生成 C 程序 demo01.c，并将内嵌 SQL 语句转变为对 SQLLIB 库函数的调用。内嵌 EXEC SQL CONNECT 语句预编译后生成的代码段示例 1-3 如下：

```
/* EXEC SQL CONNECT :username IDENTIFIED BY :password
        USING :server; */
{
 struct sqlexd sqlstm;
 sqlstm.sqlvsn = 12;
 sqlstm.arrsiz = 4;
 sqlstm.sqladtp = &sqladt;
 sqlstm.sqltdsp = &sqltds;
 sqlstm.iters = (unsigned int  )10;
 sqlstm.offset = (unsigned int  )5;
 sqlstm.cud = sqlcud0;
 sqlstm.sqlest = (unsigned char  *)&sqlca;
 sqlstm.sqlety = (unsigned short)4352;
 sqlstm.occurs = (unsigned int  )0;
 sqlstm.sqhstv[0] = (         void  *)username;
 sqlstm.sqhstl[0] = (unsigned int  )10;
 sqlstm.sqhsts[0] = (         int  )10;
 sqlstm.sqindv[0] = (         void  *)0;
 sqlstm.sqinds[0] = (         int  )0;
 sqlstm.sqharm[0] = (unsigned int  )0;
 sqlstm.sqadto[0] = (unsigned short )0;
 sqlstm.sqtdso[0] = (unsigned short )0;
 sqlstm.sqhstv[1] = (         void  *)password;
 sqlstm.sqhstl[1] = (unsigned int  )10;
 sqlstm.sqhsts[1] = (         int  )10;
 sqlstm.sqindv[1] = (         void  *)0;
 sqlstm.sqinds[1] = (         int  )0;
 sqlstm.sqharm[1] = (unsigned int  )0;
 sqlstm.sqadto[1] = (unsigned short )0;
 sqlstm.sqtdso[1] = (unsigned short )0;
 sqlstm.sqhstv[2] = (         void  *)server;
 sqlstm.sqhstl[2] = (unsigned int  )10;
 sqlstm.sqhsts[2] = (         int  )10;
 sqlstm.sqindv[2] = (         void  *)0;
 sqlstm.sqinds[2] = (         int  )0;
 sqlstm.sqharm[2] = (unsigned int  )0;
 sqlstm.sqadto[2] = (unsigned short )0;
 sqlstm.sqtdso[2] = (unsigned short )0;
```

```
    sqlstm.sqphsv = sqlstm.sqhstv;
    sqlstm.sqphsl = sqlstm.sqhstl;
    sqlstm.sqphss = sqlstm.sqhsts;
    sqlstm.sqpind = sqlstm.sqindv;
    sqlstm.sqpins = sqlstm.sqinds;
    sqlstm.sqparm = sqlstm.sqharm;
    sqlstm.sqparc = sqlstm.sqharc;
    sqlstm.sqpadto = sqlstm.sqadto;
    sqlstm.sqptdso = sqlstm.sqtdso;
    sqlstm.sqlcmax = (unsigned int )100;
    sqlstm.sqlcmin = (unsigned int )2;
    sqlstm.sqlcincr = (unsigned int )1;
    sqlstm.sqlctimeout = (unsigned int )0;
    sqlstm.sqlcnowait = (unsigned int )0;
    sqlcxt((void **)0, &sqlctx, &sqlstm, &sqlfpn);
}
```

示例 1-3　预编译后代码段示例

2. *使用系统配置文件预编译 pc 源程序*

系统配置文件 pcscfg.cfg 用于存放常用的预编译选项及设置，当安装 Oracle 时会自动建立该文件，该文件被存放在%ORACLE_HOME%\precomp\admin 目录中。当在命令行直接预编译时，需要指定多个预编译选项（INAME、INCLUDE 和 PARSE）。因为 INCLUDE 和 PARSE 是常用的、相对固定的预编译选项，所以为了简化预编译语法，可以在 pcscfg.cfg 文件中配置这两个选项。pcscfg.cfg 配置示例 1-4 如下：

```
define=(WIN32_LEAN_AND_MEAN)
parse=full
include=%oracle_home%\precomp\public
include=%oracle_home%\oci\include
include=C:\Program Files\Microsoft Visual Studio\VC98\Include
```

示例 1-4　系统配置文件 pcscfg.cfg 内容

在编辑了系统配置文件 pcscfg.cfg 之后，在命令行预编译时将不需要指定 parse 和 include 选项。示例 1-5 如下：

```
proc iname=d:\demo\demo01.pc
```

示例 1-5　使用系统配置文件预编译

3. *使用用户配置文件预编译 pc 源程序*

用户配置文件也可以用于存放预编译选项及其设置。用户配置文件 user.cfg 内容示例 1-6 如下：

```
define=(WIN32_LEAN_AND_MEAN)
parse=full
include=%oracle_home%\precomp\public
include=C:\Program Files\Microsoft Visual Studio\VC98\Include
```

示例 1-6　用户配置文件 user.cfg 内容

在编写了用户配置文件之后，如果在预编译时要使用该配置文件，那么需要在命令行指定

config 选项。示例 1-7 如下：

```
proc iname=d:\demo\demo01.pc config=d:\demo\user.cfg
```

示例 1-7 使用用户配置文件预编译

1.3 对象类型转换工具 ott

当编写 pc 源程序时，如果访问普通数据类型（例如 NUMBER、VARCHAR2、CHAR、DATE 等），那么可以直接使用 C/C++数据类型；如果访问对象类型或者集合类型，那么必须使用 C 结构。尽管开发人员可以自己定义 C 结构，但这样不仅会花费大量时间，而且容易出错。为了避免这种问题，Oracle 为开发人员提供了对象类型转换工具 OTT（Object Type Translator），该工具用于将对象类型或者集合类型转换为宿主结构和指示结构。

1. OTT 命令行选项

- USERID：该选项用于指定用户名、口令和网络服务名。语法如下：
  USERID=username/password[@server]
- INTYPE：该选项用于指定对象类型输入文件（默认扩展名为.typ），该输入文件用于指定要转换的对象类型和集合类型。读者需要注意，如果不指定该选项，那么会转换用户的所有对象类型。语法如下：
  INTYPE=<filename>
- OUTTYPE：该选项用于指定对象类型输出文件（默认扩展名为.typ），该输出文件不仅包含了 INTYPE 文件的对象类型，而且可能会包含其他需要转换的对象类型。读者需要注意，该选项必须指定。语法如下：
  OUTTYPE=<filename>
- CODE：该选项用于指定要生成的 C 代码格式。读者需要注意，该选项必须指定。语法如下：
  CODE= C|KR_C|ANSI_C
- HFILE：该选项用于指定要生成的 C 头文件名（默认扩展名为.h），该头文件包含了对象类型转换后所生成的 C 结构。读者需要注意，该选项必须指定。语法如下：
  HFILE=<filename>
- CONFIG：该选项用于指定 OTT 配置文件名。OTT 选项既可在命令行键入，也可存放到配置文件中。读者需要注意，如果在配置文件中存放命令行选项，那么每行只能设置一个选项。语法如下：
  CONFIG=<filename>
- ERRTYPE：该选项用于指定 OTT 错误日志文件（默认扩展名为.tls），该错误日志会包含 INTYPE 文件内容和错误消息。语法如下：
  ERRTYPE=<filename>
- CASE：该选项用于指定 OTT 所生成的 C 标识符的大小写。SAME 表示采用原有大小写，LOWER 表示全部转换为小写，UPPER 表示全部转换为大写，OPPOSITE 表示大写变小写、小写变大写。语法如下：
  CASE=[SAME|LOWER|UPPER|OPPOSITE]
- SCHEMA_NAMES：该选项用于指定在 OUTTYPE 文件的类型名前是否带有方案名。ALWAYS（默认）表示带有方案名，IF_NEEDED 表示需要时带有方案名，

FROM_INTYPE 表示根据 INTYPE 文件确定是否带有方案名。语法如下：
SCHEMA_NAMES=[ALWAYS|IF_NEEDED|FROM_INTYPE]

- TRANSITIVE：该选项用于指定是否转换在 INTYPE 文件中未列出的相关对象类型，默认值为 TRUE。语法如下：
  TRANSITIVE＝[TRUE|FALSE]

2. INTYPE 文件

INTYPE 文件用于列出要转换的对象类型。当编写 INTYPE 文件时，可以指定 TYPE（必需）和 CASE 选项。读者需要注意，如果在 INTYPE 文件中指定 CASE 选项，那么必须放在 TYPE 选项之前。TYPE 选项用于指定要转换的对象类型，语法如下：

TYPE [<schema_name>.]type_name [AS <type_identifier>]
[VERSION [=] <version_string>] [HFILE [=] <hfile_name>]
[TRANSLATE{<member_name> [AS <identifier>]}...]

如上所示，schema_name 用于指定方案名，type_name 用于指定对象类型名，type_identifier 用于指定转换后的 C 结构名，version_string 用于指定对象类型版本，hfile name 用于指定转换后的 C 头文件名，member_name 用于指定对象属性名，identifier 用于指定转换后的 C 结构成员名。

3. 使用 ott

为了在 pc 源程序中操纵对象类型和集合类型，需要使用 ott 工具转换对象类型和集合类型。下面以将示例对象类型 employee_type 转换为 C 结构为例，说明使用 ott 的方法。对象类型 employee_type 的建立示例 1-8 如下：

```
CREATE OR REPLACE TYPE employee_type AS OBJECT(
  eno NUMBER(6),name VARCHAR2(10),salary NUMBER(6,2),
  job VARCHAR2(10),dno NUMBER(2));
```

示例 1-8　建立对象类型 employee_type

为了将该对象类型转换为 C 结构，需要编写 INTYPE 文件。INTYPE 文件 in.typ 的内容示例 1-9 如下：

```
CASE=lower
TYPE employee_type
```

示例 1-9　in.typ 内容

使用 ott 工具有三种方法：①在命令行直接指定 ott 选项；②使用默认配置文件 ottcfg.cfg；③使用自定义配置文件。

（1）在命令行直接指定 ott 选项。

当使用 ott 命令时，必须指定 USERID、INTYPE、OUTTYPE、CODE 和 HFILE 选项，而其他选项可以根据需要进行设置。示例 1-10 如下：

```
ott userid=scott/tiger@demo intype=d:\demo\in.typ
outtype=d:\demo\out.typ CODE=C hfile=d:\demo\employee.h
```

示例 1-10　在命令行指定 ott 选项

在执行了 ott 命令之后，会生成头文件 employee.h 和 OUTTYPE 文件 out.typ 两种输出文件。其中，头文件 employee.h 包含对象类型转换后生成的 C 结构，内容示例 1-11 如下：

```
#ifndef D:\DEMO\EMPLOYEE_ORACLE
```