第13章 调 试

在建立了应用程序并运行它的时候，可能会有失败。有的时候，失败还会是明显的——遗漏了块中的一个域或者在某些需要逗点的地方输入了一个“ i ”。但是多数情况遇到的失败 (错误)是不明显的，不得不去寻找它。这种查找处理或通过找到并修正代码中的错误来解决问题的技巧叫做调试。

不确切地说，调试是测试的补充。通过简单地检查代码可以找到错误，这是一个格外有效的调试途径。测试和调试结合在一起是一种强有力的技术，可以在开发过程中尽早发现并修正错误。如果开发者彻底地测试其对象并使用本章介绍的技术查找和修正运行时引起失败的错误，那么在进行alpha测试和beta测试时花的调试时间要少得多。

使用Oracle Developer来制造软件应用程序，避免了许多可能在标准程序设计语言中遇到的错误源。不存在指针和内存分配的错误，甚至 SQL语法错误也不会发生。因为用户的 SQL语句立刻被分析，并且马上就能得到错误反馈。这意味着，如果通过对另外一种程序设计语言的学习进行调试，所学的与Oracle Developer的应用会没有关系。其实， Oracle Developer也使用了相当多的优秀的调试技巧。

13.1 调试技巧

建立测试模型是彻底了解一段代码、要求或设计组件的极好的方法。正如在本章要看到的，了解一段代码也是在调试时要做的根本问题。如果为组件对象建立对象模型，通常会发现隐藏在组件中的大部分错误。实际上，既不用运行测试和不用开发测试示例也可以发现它们。建立测试模型的技巧通常对于一个好的开发者找出在对象测试中的逻辑和代码缺陷是足够了。

然而，有些错误需要进一步调查。这种情况经常发生，因为有些错误包含容易被忽略的问题，某些问题在不止一处或一个对象中发生，某些问题建立在隐含假设的基础上，而不是开发的测试模型的一部分。对于这些问题，才是调试技巧真正发挥作用的时候。

本部分的标题来自一本可能是现有的最好的关于测试和调试软件的书： Glenford Myers的《The Art of Software Testing》(Wiley,1979)。尽管这本书比较早，并且逐渐在测试和调试的技术和理论方面有些过时，但这本书对其主题提供了非常实用的介绍。没有哪个从事软件开发的人在开发软件时不读这本书的。本部分总结了 Myers介绍的调试技巧。还介绍了如何把Oracle Developer调试工具添加到调试板中和如何使用它们有效地支持调试。

13.1.1 通过归纳法定位错误

归纳的过程是从局部到整体的推理 (见图13-1)。那就是，利用故障的数据，发现故障的本质。

1) 定位相关故障数据 不要只是说“ 那里有问题 ”就开始修正。要通过留心观察精确地确定故障如何发生；如何使其再次发生；如何在相似但不同的测试情况下避免问题。要研究

故障发生的来龙去脉。

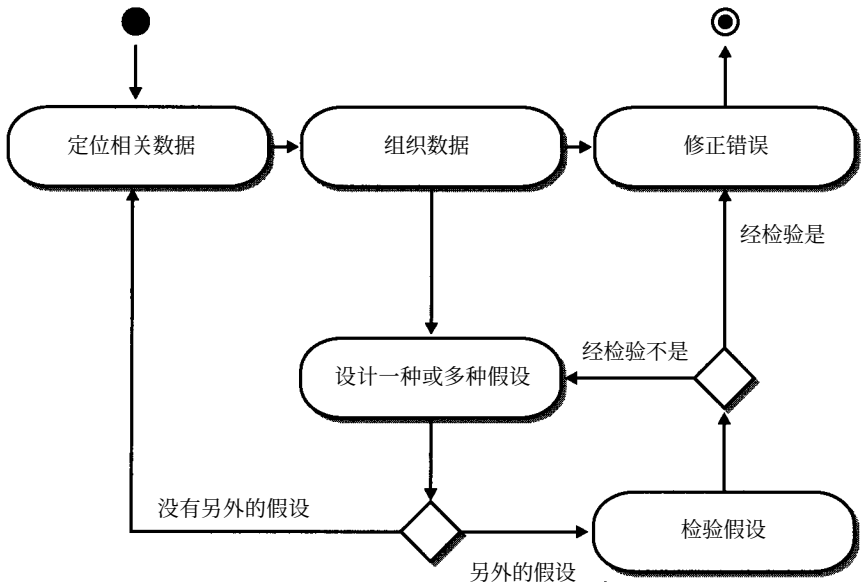

图13-1 归纳调试过程

2) 组织故障数据 通过模型分析进行归纳。如果数据组织得好，看出数据中的模型和关系就容易得多。Myers用一个表来显示发生了什么，在哪里发生，什么时候发生，以及在发生和不发生的环境下在哪个范围发生。把事件按剧本组织也十分有用。因此，要研究数据显露的关系模型。一个尤其有效的方法是回溯法：从失败的位置开始，按照进程反方向通过数据处理。另外一种可行的方法是用一些其他的测试情况进行处理，把模型看得更清楚。

3) 设计一种假设 研究数据的关系之后，对错误提出一种或多种假设。如果还不能提出假设，则需要获取更多的数据。

4) 检验假设 选择最可能的假设，然后通过它的表现检验其是否正确。首先检验它是否符合已知数据，第二，检验它在代码中如何起作用。

5) 修正错误 按照假设的建议修改代码，然后再测试代码，检验并修正那些确认了的错误，并且确认没有引起另外的错误。

当有一组数据并且相对来说不太了解程序如何工作，不太了解错误的根源时，可以使用归纳法。归纳法与后面要介绍的演绎法相比，需要进行彻底的数据分析，花大量时间和努力。

13.1.2 通过演绎法定位错误

演绎的过程是从整体到局部的推理 (见图13-2)。那就是利用对应用程序和系统的了解，通过排除和提练假设来推理错误的根源。

1) 列出可能引起失败的错误 最好与其他开发者集体讨论引起失败的错误原因。实际上，部分假设是很有用的。有的假设来自常规程序设计推测，例如查看最近改变的代码 (“我只改变了一个地方，它不可能引起故障！”)；非常复杂的代码(“我必须这样做，否则无法实现其功能！”)；或者曾经有错误的代码(“不可能再有错误了！”)。有的假设来自程序设计经验，例如边界条件错误或对SQL结构的误解，例如GROUP BY HAVING及其他外部连接(“哪一个

使用加号？”)。

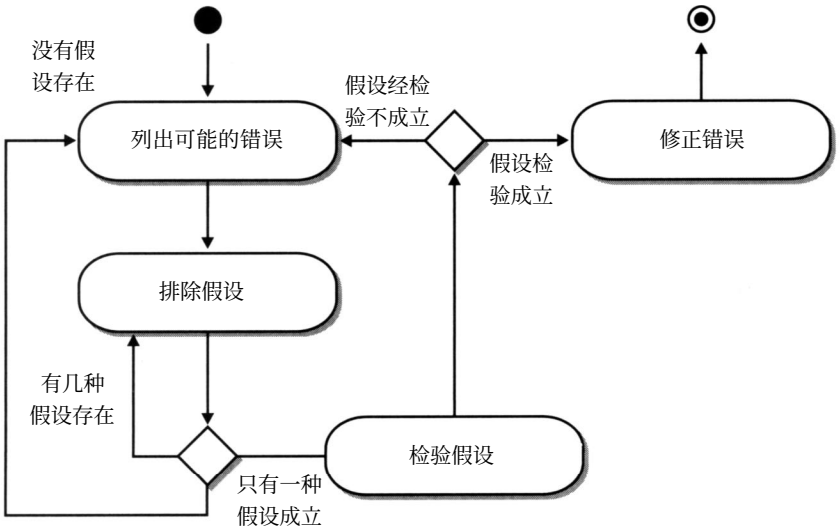

图13-2 演绎调试过程

2) 排除假设 分析数据及其关系。如果只能排除一种假设，那么着眼于剩下的假设搜集更多的数据。通常，采用某些附加测试条件会做得更好。如果排除了所有的假设，再集体讨论新的假设。

3) 精练并检验假设 使用可利用的数据尽可能地使错误推测具体化。到底要发生什么？查看所提练的假设是否与已知数据相符合，它在代码中如何起作用。

4) 修正错误 按照假设的建议修改代码，然后再测试，检验修正是否防止了错误的发生，并且没有引起另外的失败。

如果很充分了解程序的作用及失败的根源时，应该使用演绎法。这种充分了解使你能够提出一系列假设。这种方法的关键部分是通过数据分析排除并提练假设。跳过数据分析通常会导致无目的的乱撞(删去一句幽默比喻 —— 译者注)。应该检查数据，然后检验假设。

13.1.3 通过跟踪定位错误

发现错误的强制方法是跟踪程序，一个事件接一个事件，直到发现某处有错误。在调试(也就是跟踪)时，大多数开发者对要找什么有良好的直觉。但是，应该清楚这种方法比归纳法或演绎法的效率要低得多，因为在处理过程中没有“思考”。 在处理过程中会产生大量的数据，其中多数是与错误不相干的。并且程序越大，越是这样。而技巧不能够递增适应性。

当然，如果不知道在程序中发生了什么并且对错误的了解不充分，那么跟踪是找到错误的唯一途径。对于Oracle Developer的程序来说，只有在不了解Oracle Developer如何处理时，这种情况才会发生。例如，不了解第五章介绍的处理过程，并且即使通过归纳处理也不能可靠地重演失败，那么就可能要通过调试程序进行单步调试，直到有异常的事情发生。这要花费一定的时间。如果没有什么发生，可能会认为Oracle Developer使用演绎法来代替强制跟踪的方法更好。

13.1.4 错误定位技术

下面是归纳法和演绎法的一些思想，没有特别的顺序：

- 使用大脑 这可能是对程序设计最有益的东西。
- 使用下意识 把错误放在一边，去作其他的的事。通常会灵机一动来解决你的忧虑。
- 使用朋友 和朋友讨论问题。讨论可能会使你找到一种新的假设，或者有人能够提出一种不同的假设。
- 避免强行修改 尤其尽量不要随便地改变应用程序来看要发生的事情。这比跟踪更坏，因为它分裂了执行环境，潜在地增加了新的故障。

13.1.5 修正错误

修正错误还有一些其他的处理，不仅仅是改变代码。

- 在修正错误时，注意周围的代码。
- 要确认修正了全部错误。意思是需要了解全部的问题是什么。换句话说，在没有彻底弄明白应用程序如何失败和由失败的衍生错误之前，不要开始修正错误。归纳法和演绎法调试是最有可能使你了解全部失败的方法。另外，不要只修正了表面错误而忽略了根源。软件故障比人为的故障容易消除。
- 确信在修正错误时没有破坏其他部分。尤其在应用程序是如此庞大和复杂时，每次修改可能引入新错误的概率很高。避免这种问题的最好方法是回归测试，测试能够证明所有的部分在做它应该做的事。另外一种方法是检查代码。用这种方法，需要检查所有与被修改代码相关的代码，甚至是不直接相关的代码。
- 应该研究代码中的错误模式，从错误中吸取教训。如果代码中有某个错误需要修改，那么就修改它。如果发现某种错误好象经常在 Oracle Developer的代码中发生，可在适当的位置上放置一个标准样品来看是否能够防止它们。如果发现某些错误比其他错误发生得频繁，就把工作集中到这些错误上，来减少这些错误。

13.2 调试Oracle Developer对象

假设现在完全怀疑自动调试工具作为调试方法，本章剩余部分将介绍如何有效地使用这些工具。主要是在调试的假设-检验层上使用这些工具。

Oracle Developer的环境可以以几种不同的途径进行调试。内置的调试程序可以跟踪整个代码，并且事实上能够随意地查看任何事。表单和报表也能够使用跟踪工具进行跟踪，能够知道现在程序执行到什么地方。

13.2.1 调试程序

调试程序可以在应用中以交互的现代跟踪技术跟踪 PL/SQL：

- 在源代码中交互处理断点和触发器。
- 交互查看、处理PL/SQL和变量。
- 查看PL/SQL调用堆栈。
- 交互执行PL/SQL命令。

每种工具以不同的方式建立应用程序的调试程序。

1. Forms的调试程序

在Oracle Developer Forms的应用程序中使用调试程序，必须使用调试选项生成表单和菜

单可执行文件。单击Object Navigator Tool Palette上的Run Form-Debug工具，或使用Program|Run Form|Debug菜单项在 Debug mode(调试方式)运行。每当生成应用程序时， Oracle Developer/Forms嵌入源代码符号到应用程序中。

要在运行应用程序时运行调试程序，或者在生成器中用 Run Form Debug工具或菜单项运行，或者在调试信息编译之后，用特定的调试可执行文件 IFDBG60带Forms Runtime命令行参数DEBUG = YES运行:

```
IFDBG60 workerskill talbot/george@orcl debug=YES
```

运行系统在加载表单之前显示调试程序窗口，以便设置任何调试行为 (见图13-3)。

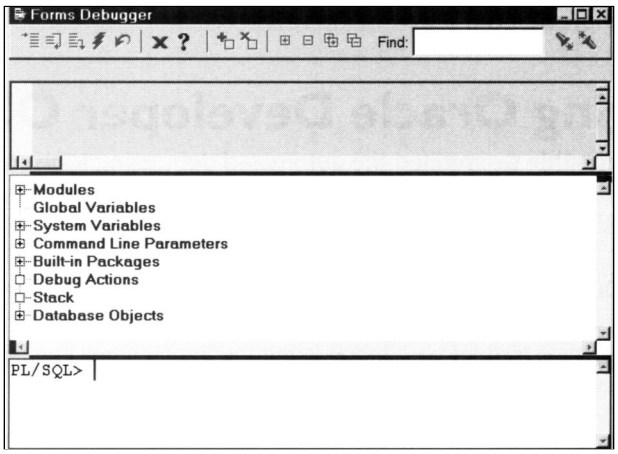

图13-3 Forms Debugger窗口

在这里，可以在任何触发器或应用的程序单元中设置断点，并且可以建立调试触发器 (由调试程序事件激活的触发器)。

在顶部窗格中，调试程序显示当前执行块的 PL/SQL源程序，这是源程序窗格。由于当前调试程序还没有执行一个块，所以是空的。可以在这个窗格中设置只有在调试中才存在的断点。中间窗格是Object Navigator窗格，除了少数对象，这个窗格与 Object Navigator很像。这个Object Navigator有一组在运行调试时使用的对象，但它们不是在生成器的 Object Navigator中的那些。使用这个窗格来查看调试行为和变量 (全局、系统、局部)和调用堆栈中的参数。最下层的窗格是Interpreter(注释器)窗格，输入执行的命令在这里显示。注释器窗格用于输入命令并执行命令。

注意 如果在Forms Runtime中运行到一个好象要出错但又看不出错误和异常的地方，可以检查表单中的On-Error触发器。如果触发器用WHEN OTHERS子句捕捉了所有异常，而不是只处理特定异常并把其他的异常传递的话，那么触发器就吸收了所有的错误，而不给出任何反馈。通过注释WHEN OTHERS子句来消除这样的子句。通过插入适当的异常处理来修正问题(异常处理参见第13章)。

2. Reports调试程序

与Forms不同， Reports的调试程序把其调试行为存储在报表本身中。在 Report Builder的Object Navigator中,可以看到两个标题： Debug Actions和Stack。生成器存放断点和触发器行为，后者描述运行时调用的堆栈。

要建立断点，在选择Report Builder Object Navigator中的程序单元后通过Program|PL/SQL Interpreter(注释器)菜单项调用Report调试程序。显示一个非模态的两个窗格（Source程序和Interpreter窗格)的调试程序(见图13-4)。

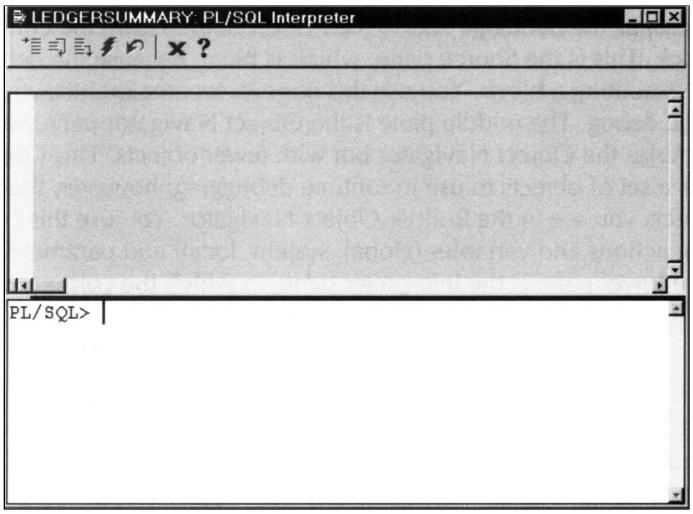

图13-4 两个窗格的调试程序

当在Object Navigator中单击一个程序单元或触发器时，PL/SQL注释器显示PL/SQL代码。然后就可以在代码的任何地方设置断点(在下面)，然后保存报表。当运行报表时，在到达断点时，运行时调试程序弹出并高亮度显示断点(不需要滚动窗口)。

注意 如果重新编译程序单元，需要重设受影响的断点到其新的行号。

3. Graphic调试程序

Oracle Developer Graphics调试程序的工作方式与在Oracle Developer Reports中的十分相似。在Graphics Builder中通过Tools|PL/SQL Interpreter(注释器)菜单项调用非模态的调试程序。

大多数的调试工作是找出在图形显示和图表中的哪个选项是错误的，调试程序对这些并没有很大用处。可以通过单独运行图形显示来“测试”不同的选项。到需要把图形形式集成到表单或报表中的时候，应该对图形的内部一致性有把握。

如果图形显示有运行时的错误，可以采用和表单及报表类似的调试程序。可以设置断点或调试触发器和查看变量和参数。除非在图形显示中广泛地使用PL/SQL，否则调试程序的作用不大。

注意 如果在调试程序中修正程序错误并重新编译程序，应该在完成调试后选择Edit|Synchronize Program Units菜单项。这样就同步修改了在调试程序中的代码与在Designer或Builder中的代码。

4. Sources窗格和断点

Sources窗格以只读文本的形式显示源程序，使用源程序窗格来设置断点和查看执行过程中处于什么地方。如图13-5所示是一个在子程序体中设置有断点的调试过程。

注意窗格左边行号处的B(01)。这显示子程序中这一行上有一个断点(断点1)。可以在Sources

窗格中单击行来设置断点。箭头(指向B)显示调试程序在这一行前停止了程序执行。

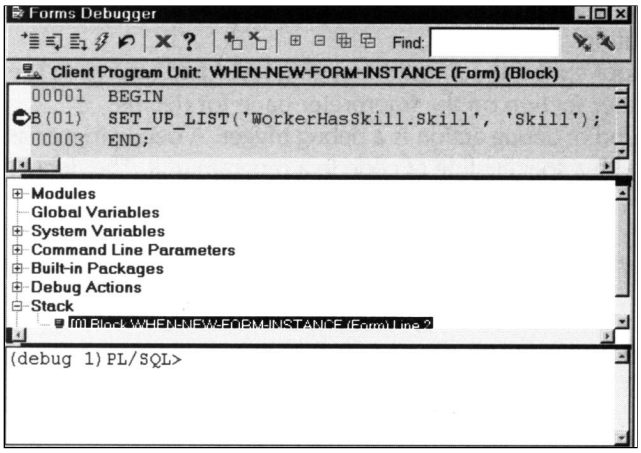

图13-5 在子程序体中设置有断点的调试过程

在最初的调试程序(比如Forms Runtime)启动时，可以在想要开始调试的地方设置一个断点。然后单击Close工具，开始执行表单。表单执行到断点处停止。当表单中断时，调试程序窗口再次显示。然后可以在Navigator窗格中检查变量(参见下一节)，或者使用Step命令开始单步执行代码。

单击Step Into工具，可以执行下一条语句。如果该语句是一个子程序调用，则进入子程序并停在子程序的第一个可执行的行上。 Step Over与Step Into类似，但Step Over执行子程序，并停在子程序调用后的下一个可执行的行上。 Step Out Of工具连续执行，直到当前子程序或块结束返回为止。然后调试程序停在下一个可执行的行上(见图13-6)。

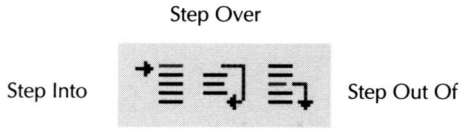

图13-6 使用Step命令

通过使用这些命令，可以在子程序中按照执行路线跟踪执行，一条语句接一条语句执行。在任何地方，都可以检查任意变量的值，查看发生了什么。

使用Go工具可以连续执行，直到遇到下一个断点或退出应用程序为止。

Reset工具退出当前的层次调试程序并返回上一个层次，参见后面关于注释器窗格的部分。

另一种调试行为是调试触发器。调试触发器把一个 PL/SQL块与程序单元代码中的一个特定行连接到一起。当执行到这行时，调试程序触发触发器。也可以设置为调试程序取得控制权或在每一个PL/SQL源程序行触发触发器。通过产生作为触发器一部分的 DEBUG.BREAK异常，可以引起中断。这样就可以设置条件断点：

```
IF DEBUG.GETN('RELDEF') = TRUE THEN
  RAISE DEBUG.BREAK;
END IF;
```

在触发器代码中，使用 GETN和SETN子程序访问在当前作用域的变量。 Break异常引起调试程序中断并显示当前源程序行 (触发调试触发器的那行)。

警告 也可以在PL/SQL代码中用Break过程建立作用持续的断点。本书不推荐调试触发器和作用持续的断点，因为至少有两个原因。第一，主要是使用中断和调试程序检验假设，而不是广泛地跟踪代码。使用跟踪设备就可以实现上述目的。第二，如果忘记了代码中有中断并把中断遗留在代码中，用户可能会意外地进入调试程序。这不利于

提供一个高质量的应用程序，相反，这实际上是在代码中加入了错误。调试触发器通过自动测试条件可以帮助加速处理过程，尤其是在循环中，但是很少会用到它。

5. Navigator窗格和变量

Navigator窗格可以检查不同类型的变量。可以在Module(模块)的标题下找到任何项并查看该项。通过在相应标题下查找，可以查看全局变量、系统变量和命令参数。Stack标题提供了PL/SQL调用堆栈的当前状态。这个堆栈显示了当前执行的 PL/SQL块的分层结构。这个堆栈标题显示调试程序正在执行一个触发器 —— ON-POPULATE-DETAILS，接下来应该调用一个子程序 —— QUERY_MASTER_DETAILS(见图13-7)。

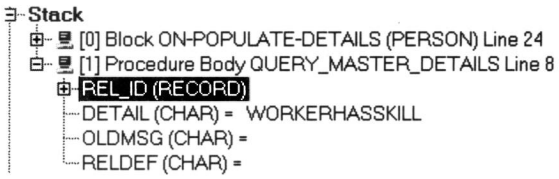

图13-7 Stack标题

在这个堆栈中，每个项代表一个正在执行的定义变量的作用域，并且可以在Stack项下找到这些变量(参数和局部变量)。如图13-8所示的例子显示在运行的 Set_Up_List子程序中的变量，包括变量ERRFLAG，被对Populate_Group的调用分配。

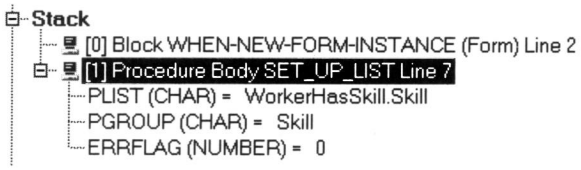

图13-8 运行Set_UP_List子程序中的变量

Navigator窗格显示所有变量的当前值，包括参数。例如， PLIST的值为WorkerHasSkill.Skill，是列表项的数据块和项名。可以通过选中并在等号后直接输入值在 Navigator中改变这些变量的值。

注意 调试程序现在不支持程序包的程序体的变量，所以没有办法查看这些变量的值。程序包的程序体变量是在程序包的程序体的说明部分说明的变量。在程序包的外部无法访问这些变量，但是在程序包中所有的子程序体和程序包初始化代码可以共享这些变量。细节参见第12章。

也可以通过断点和触发器对话框查看调试行为的细节。在 Debug Action标题下，可以选中任意的断点或调试触发器，然后双击其图标，就可以看到对话框。图 13-9是前面所述的断点的Breakpoint(断点)对话框。

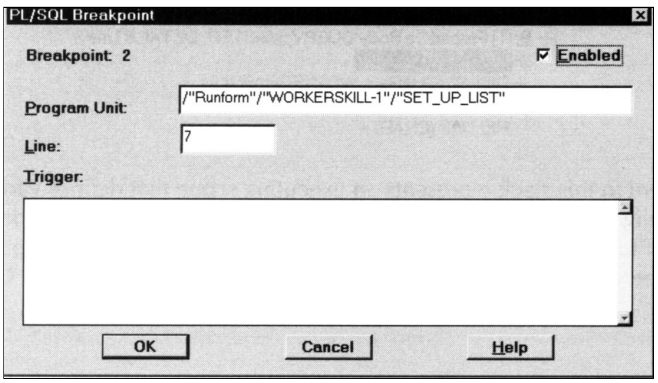

图13-9 PL/SQL Breakpoint(断点)对话框

使用这个对话框，可以通过Enabled复选框打开或关闭断点。也可以输入一些PL/SQL代码，作为中断发生时的触发器来执行。

6. 注释器窗格和命令

注释器窗格可以输入文本形式的调试程序命令，例如 Step Into或Go。也可以看到调试层次。调试层次是在程序执行中中断的嵌套层数。也就是说，中断程序单步执行越过子程序(Step Over)，在子程序中又遇到了中断。然后在注释器窗格中建立调试层 2。在注释窗格中可以看到一个加上圆括号的表达式作为命令提示符，例如“ (debug1)PL/SQL> ”。这个调试层的嵌套使你能够重置调试程序回到上一调试层，重新启动调试任务程序。

13.2.2 跟踪表单

在Oracle Developer Forms组件中有两个跟踪工具 —— 一个新的和一个旧的。首先说旧的。可以在运行应用程序时打开专门的调试信息。当一个事件引发一个触发器执行时，运行系统在信息行显示一个信息和一个警告窗口。单击警告窗口中的 OK按钮，应用程序继续执行。

调试信息显示触发器名和拥有触发器的对象 (表单、块或项)名(见图13-10)。

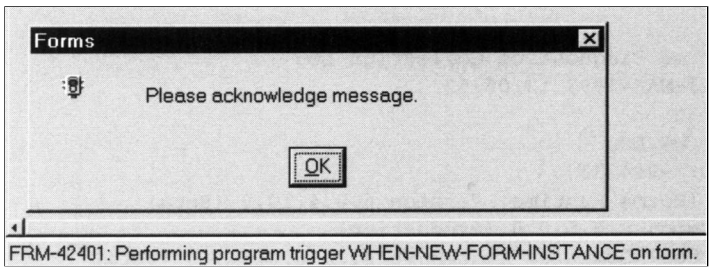

图13-10 警告窗口

要在Forms Runtime中显示信息，在Forms Runtime命令行中加上DEBUG_MESSAGES参数即可。

```
IFRUN60 workerskill talbot/george@orcl debug_messages=YES
```

要在Form Builder中运行表单时显示同样的信息，选择 Tools|Preferences菜单项并在Preferences对话框的Runtime标签上选择Debug Messages preference复选框即可。

注意 如果使用调试信息，将发现需要大量数据和很长的调试时间。可以至少尝试一次，查看触发器数量适当的表单到底发生什么。这给“ 最后一着 ”这个名词一个新的含义。

现在介绍新的跟踪工具：Forms Runtime Diagnostics(FRD，表单运行诊断)。FRD工具提供了对Oracle Developer/Forms执行的完全的跟踪，跟踪结果放在一个跟踪文件里，在你空闲时可以验证它。可能你需要大量的空闲，因为跟踪真的是很完全。

运行要跟踪的表单并在命令行中加上 RECORD和LOG参数，即可启动FRD：

```
IFRUN60 workerskill talbot/george@orcl RECORD=COLLECT LOG=WS.LOG
```

这个命令行运行表单，跟踪表单中所有的活动，并把跟踪结果写到当前工作目录下的WSP500LOG文件中。下面是这个文件前几个跟踪部分的例子：

```
Forms Runtime Diagnostics Collection Log
Created: 13-MAR-1999 12:06:52

File Name: WS.LOG
```

```
Process ID: -946733
Forms 6.0 (Forms Runtime) Version 6.0.4.10.0 (Beta)
PL/SQL Version 8.0.5.0.0 (Production)
Oracle Virtual Graphics System Version 6.0.5.3.0 (Beta)
Oracle Multimedia Version 6.0.2.0.0 (Production)
Oracle Tools Integration Version 6.0.5.1.0 (Production)
Oracle Tools Common Area Version 6.0.5.3.0
Oracle CORE Version 4.0.5.0.0 - Production


Opened file: C:\BOOKS\DEV3\CODE\workerskill.fmx

WHEN-NEW-FORM-INSTANCE Trigger Fired:
Form: WORKERSKILL

State Delta:
FORM WORKERSKILL
  CURBLOCK    PERSON
  CURFIELD    NAME
  STATUS      NEW
  BLOCK PERSON
    RECCOUNT    1
    TOPREC      1
    CURREC      1
    STATUS      NEW
    RECSTATUS   NEW
    FIELD NAME
      CANVAS      CANVAS2
      GEOMETRY    499,67:1694,240
      MANDATORY   TRUE
      ENABLED     TRUE
      NAVIGABLE   TRUE
      INSERTABLE  TRUE
      QUERYABLE   TRUE
      UPDATEABLE  TRUE
  BLOCK WORKERHASSKILL
    RECCOUNT    1
    TOPREC      1
    CURREC      1
    STATUS      NEW
    RECSTATUS   NEW
    FIELD NAME
      MANDATORY   TRUE
      INSERTABLE  TRUE
      UPDATEABLE  TRUE
    FIELD SKILL
      CANVAS      CANVAS2
      GEOMETRY    0,544:1000,240
      MANDATORY   TRUE
      ENABLED     TRUE
      NAVIGABLE   TRUE
      INSERTABLE  TRUE
      QUERYABLE   TRUE
      UPDATEABLE  TRUE
    FIELD ABILITY
      CANVAS      CANVAS2
      GEOMETRY    1000,544:1069,240
```

```
      ENABLED     TRUE
      NAVIGABLE   TRUE
      INSERTABLE TRUE
      QUERYABLE   TRUE
       UPDATEABLE TRUE

Executing FIND_GROUP Built-in:
In Argument 0 - Type: String    Value: Skill
Out Argument 0 - Type: Integer    Value: 1

Executing POPULATE_GROUP Built-in:
In Argument 0 - Type: Integer  Value: 1
Out Argument 0 - Type: Number    Value: 0

Executing FIND_ITEM Built-in:
In Argument 0 - Type: String    Value: WorkerHasSkill.Skill
Out Argument 0 - Type: Integer    Value: 131075

Executing Clear_List Built-in:
In Argument 0 - Type: Integer  Value: 131075

Executing FIND_ITEM Built-in:
In Argument 0 - Type: String    Value: WorkerHasSkill.Skill
Out Argument 0 - Type: Integer    Value: 131075

Executing FIND_GROUP Built-in:
In Argument 0 - Type: String    Value: Skill
Out Argument 0 - Type: Integer    Value: 1

Executing Populate_List Built-in:
In Argument 0 - Type: Integer  Value: 131075
In Argument 1 - Type: Integer  Value: 1

# 1 - WORKERSKILL:PERSON.NAME - WINDOW
END

# 1 - WORKERSKILL:PERSON.NAME
WINDOW WORKERSKILL WINDOW1 ACTIVATE 1

# 2 - WORKERSKILL:PERSON.NAME - MENU
END

# 2 - WORKERSKILL:PERSON.NAME
MENU DEFAULT Query eXecute

ON-CLEAR-DETAILS Trigger Fired:
Form: WORKERSKILL

State Delta:

Executing FIND_BLOCK Built-in:
In Argument 0 - Type: String    Value: PERSON
Out Argument 0 - Type: Integer    Value: 1

Executing GET_BLOCK_PROPERTY Built-in:
In Argument 0 - Type: Integer  Value: 1
In Argument 1 - Type: Number  Value: 160
```

```
Out Argument 0 - Type: String   Value: PERSON_WORKERHASSKILL

Executing FIND_RELATION Built-in:
In Argument 0 - Type: String   Value: PERSON_WORKERHASSKILL
Out Argument 0 - Type: Integer   Value: 65537

Executing GET_RELATION_PROPERTY Built-in:
In Argument 0 - Type: Integer  Value: 65537
In Argument 1 - Type: Number  Value: 152
Out Argument 0 - Type: String   Value: WORKERHASSKILL

Executing FIND_BLOCK Built-in:
In Argument 0 - Type: String   Value: WORKERHASSKILL
Out Argument 0 - Type: Integer   Value: 2

Executing GET_BLOCK_PROPERTY Built-in:
In Argument 0 - Type: Integer  Value: 2
In Argument 1 - Type: Number  Value: 100
Out Argument 0 - Type: String   Value: NEW

Executing FIND_RELATION Built-in:
In Argument 0 - Type: String   Value: PERSON_WORKERHASSKILL
Out Argument 0 - Type: Integer   Value: 65537

Executing GET_RELATION_PROPERTY Built-in:
In Argument 0 - Type: Integer  Value: 65537
In Argument 1 - Type: Number  Value: 153
Out Argument 0 - Type: String   Value: NULL

ON-POPULATE-DETAILS Trigger Fired:
Form: WORKERSKILL
Block: PERSON

State Delta:
FORM WORKERSKILL
  STATUS     QUERY
  BLOCK PERSON
    STATUS     QUERY
    RECSTATUS  QUERY
    FIELD ROWID
      VALUE      0004B006.0000.0001
    FIELD NAME
      VALUE      "Bart Sarjeant"
```

通过这个例子可以看到，跟踪给出了想要知道的详细情况，可以了解 Oracle Developer作了什么。跟踪文件给出了每个触发器的触发的和能够引起数据改变的每个事件。数据的改变伴随着一个state delta —— 显示自最近的state delta以来任何改变了的数据元素。

13.2.3 跟踪报表

用Tools|Trace菜单项打开报表跟踪，即可看到Runtime Trace Setting对话框(见图10-11)。

在这里可以复选想要跟踪日志显示的统计信息。下面是在复选了所有跟踪选项产生的Ledger Summary报表跟踪日志的第一部分：

```
LOG :
      Report: Ledger
      Logged onto server:
      --
```

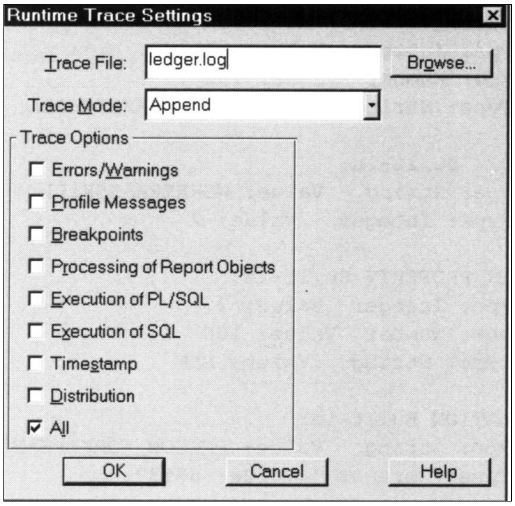

图13-11 Runtime Trace Setting对话框

```
      Username:


LOG :
      Logged onto server: orcl
      Username: talbot


15:33:48  APP  (  Header
15:33:48  APP . (  Chart                    BOUGHTDISPLAY
15:33:53  APP . )  Chart                    BOUGHTDISPLAY
15:33:53  APP  )  Header
15:33:56  APP  (  Frame
15:33:56  APP . (  Generic Graphical Object B_1
15:33:56  APP . )  Generic Graphical Object B_1
15:33:56  APP . (  Text Boilerplate         B_DATE_BLULOGO1
15:33:57  APP . )  Text Boilerplate         B_DATE_BLULOGO1
15:33:57  APP . (  Text Boilerplate         B_TITLE_BLULOGO1
15:33:57  APP . )  Text Boilerplate         B_TITLE_BLULOGO1
15:33:57  APP . (  Text Boilerplate         OR$_BPAGENUM_BLULOGO1
15:33:57  APP . )  Text Boilerplate         OR$_BPAGENUM_BLULOGO1
15:33:57  APP . (  Text Field               F_DATE1
15:33:57  APP .. (  Database Column          Name unknown
15:33:57  APP .. )  Database Column          Name unknown
15:33:57  APP . )  Text Field               F_DATE1
15:33:57  APP  )  Frame
15:33:57  APP  (  Frame
15:33:57  APP . (  Frame                    M_G_PERSON_GRPFR
15:33:57  APP .. (  Repeating Frame          R_G_PERSON
15:33:57  APP ... (  Group                    G_Person  Local Break:  0  Global Break:  0
15:33:57  APP .... (  Query                    LedgerQuery
15:33:57  SQL         EXECUTE QUERY : select Person , ActionDate , Item , Quantity , Rate
                      from Ledger  ORDER BY 1 ASC , 1 , 2
15:33:57  APP .... )  Query                    LedgerQuery
15:33:57  PLS .... (  Function:       amountformula
15:33:57  PLS .... )  Function:       amountformula
15:33:57  APP ... )  Group                   G_Person
15:33:57  APP ... (  Text Field              F_PERSON
15:33:57  APP .... (  Database Column         Person
15:33:57  APP .... )  Database Column         Person
15:33:57  APP ... )  Text Field              F_PERSON
15:33:57  APP ... (  Frame                   M_G_ACTIONDATE_GRPFR
15:33:57  APP .... (  Repeating Frame         R_G_ACTIONDATE
15:33:57  APP ..... (  Group                    G_ActionDate  Local Break:  0  Global Break:  0
15:33:57  APP ..... )  Group                    G_ActionDate
15:33:57  APP ..... (  Text Field               F_RATE
15:33:57  APP ..... (  Database Column           Rate
15:33:57  APP ..... )  Database Column           Rate
15:33:57  APP ..... )  Text Field               F_RATE
15:33:57  APP ..... (  Text Field               F_QUANTITY
15:33:57  APP ..... (  Database Column           Quantity
15:33:57  APP ..... )  Database Column           Quantity
15:33:57  APP ..... )  Text Field               F_QUANTITY
```

```
15:33:57  APP ..... (  Text Field              F_ITEM
15:33:57  APP ..... (  Database Column          Item
15:33:57  APP ..... )  Database Column          Item
15:33:57  APP ..... )  Text Field              F_ITEM
15:33:57  APP ..... (  Text Field              F_ACTIONDATE
15:33:57  APP ..... (  Database Column          ActionDate
15:33:57  APP ..... )  Database Column          ActionDate
15:33:57  APP ..... )  Text Field              F_ACTIONDATE
15:33:57  APP ..... (  Group                   G_ActionDate  Local Break:  1  Global Break:  1
15:33:57  PLS ..... (  Function:       amountformula
15:33:57  PLS ..... )  Function:       amountformula
15:34:05  APP ... )  Frame                     M_G_ACTIONDATE_GRPFR
15:34:05  APP ... (  Group                     G_ActionDate  Local Break:  1  Global Break:  225
15:34:05  APP ... )  Group                     G_ActionDate
15:34:05  APP ... (  Group                     G_Person  Local Break:  52  Global Break:  52
15:34:05  APP ... )  Group                     G_Person
15:34:05  APP .. )  Repeating Frame           R_G_PERSON
15:34:05  APP .. (  Frame                     M_G_PERSON_HDR
15:34:05  APP ... (  Text Boilerplate          B_RATE
15:34:05  APP ... )  Text Boilerplate          B_RATE
15:34:05  APP ... (  Text Boilerplate          B_QUANTITY
15:34:05  APP ... )  Text Boilerplate          B_QUANTITY
15:34:05  APP ... (  Text Boilerplate          B_ITEM
15:34:05  APP ... )  Text Boilerplate          B_ITEM
15:34:05  APP ... (  Text Boilerplate          B_ACTIONDATE
15:34:05  APP ... )  Text Boilerplate          B_ACTIONDATE
15:34:05  APP ... (  Text Boilerplate          B_PERSON
15:34:05  APP ... )  Text Boilerplate          B_PERSON
15:34:05  APP .. )  Frame                     M_G_PERSON_HDR
15:34:05  APP . )  Frame                     M_G_PERSON_GRPFR
15:34:05  APP  )  Frame

                    +-------------------------------------+
                    | Report Builder Profiler statistics  |
                    +-------------------------------------+

          TOTAL ELAPSED Time:      60.00 seconds

         Reports Time:      60.00 seconds (100.00% of TOTAL)

                 ORACLE Time:       0.00 seconds ( 0.00% of TOTAL)

                        UPI:        0.00 seconds
                        SQL:        0.00 seconds

TOTAL CPU Time used by process: N/A
```

与其跟踪整个报表，不如使用 SRW包通过在报表触发器代码中调用 SRW.TRACE_START()和SRW.TRACE_END()来打开对报表处理中特定处的跟踪。可以通过 SRW.TRACE_ADD_OPTIONS()和SRW.TRACE_REM_OPTIONS()来设置跟踪选项。

13.2.4 跟踪SQL和客户机/服务器事件

在某些情况下有两种跟踪可能会有用处，尽管严格地说它们并不是 Oracle Developer的一部分。通常有这种情况经常发生，就是你不知道在发生什么并且不得不从大量的几乎没用的资料去寻找发生的问题，所以客户机和服务器级的跟踪仍然是对应用程序开发者有价值的工具。再次给“最后一着”以新的含义。

1. 在客户机端的SQL*NET跟踪

如果使用 Oracle 7数据库服务器和 SQL*Net版本2从客户机连接到服务器，可以使用SQL*Net跟踪。它提供了 SQL*Net知道的所有事件的十六进制转储 (dump)。可以直接看到哪个字符串从客户机端发送到服务器，这也许足够用来确定错误。这种跟踪对于调试第三代程序很有帮助，在其中可能通过改写存储器位置或偶尔在字符串末尾连接上多余的字符串造成奇怪的情况。这在Oracle Developer中很难作到。

打开SQL*Net跟踪需要在Oracle_Home/Network/Admin目录下的SQLNET.ORA文件中设置两个参数：TRACE_LEVEL_CLIENT(值USER或ADMIN为网络信息摘要或详细数据， OFF

为关闭跟踪)和TRACE_DIRECTORY_CLIENT(值为要建立跟踪文件的目录)。跟踪结果写在指定目录下的SQLNET.TRC文件中。

注意 如果上面的说明不充分，向Oracle Corporation Technical Support(Oracle公司技术支持)查询使用这种技术的更多信息。这些细节相当复杂，超出了本书的范围。由于跟踪文件增长得非常快，要确定有足够的磁盘空间，并且在取得了需要得跟踪信息后及时关闭跟踪。不要在网络基础上打开跟踪，应该仅仅是在本地 (局部变量设置或SQLNET.ORA本地副本)，否则网络系统管理员会发现系统中的可用磁盘空间的急剧减少。

2. 在客户机端的Net8跟踪

如果使用Oracle 8数据库服务器，可以使用Net8从客户机连接到服务器，从而可以使用Net 8跟踪查看发送到服务器的SQL(也包括所有其他网络相关事件)。Net8提供了所知道的所有事件的十六进制转储。可以直接看到哪个字符串从客户机端发送到服务器，这也许足够用来确定错误。这种跟踪对于调试自定义的Pro*C或Oracle Call Interface(Oracle 调用接口)程序很有帮助，在其中可能通过改写存储器位置或偶尔在字符串末尾连接上多余的字符串造成奇怪的情况。这在Oracle Developer中很难作到。

打开Net8跟踪很容易。在客户机端Net8 Admin目录下找到SQLNET.ORA文件，通常是在Oracle_Home\Net80\Admin这样一些地方。通常在文件中会发现一个与跟踪相关的行，TRACE_LEVEL_CLIENT=OFF。把OFF改为SUPPORT并加上两行来给跟踪文件起名字：

```
TRACE_LEVEL_CLIENT = SUPPORT
TRACE_FILE_CLIENT = Net8_Trace
TRACE_DIRECTORY_CLIENT = C:\Orawin95\Net80\Trace
```

这个例子打开了SUPPORT-level跟踪并设置跟踪文件为C:\Orawin95\Net80\Trace\ Net8_Trace.trc。可以随便给文件起名字，但不要加扩展名。可以随便设置目录，但要确保在运行一个客户机Oracle程序之前这个目录必须存在。 LEVEL有另外的级别值，包括USER和ADMIN，但它们对输出有限制。 SUPPORT可能是最好的选择，因为它给出了所有存在的信息并且有一个跟踪文件格式化工具可用，如trcasst.

注意 关于使用Net8进行跟踪和跟踪辅助工具的更多信息，查阅 Oracle Network Administration(网络管理员)文档。例如，如果运行第 4章的Ledger应用程序并按下Execute Query按钮来查询所有记录，将得到下面的Net8跟踪文件：

```
(3ff2) nspsend: 00 FD 00 00 06 00 00 00  |........|
(3ff2) nspsend: 00 00 03 5E 0D 61 80 00  |...^.a..|
(3ff2) nspsend: 00 00 00 00 00 98 F2 00  |........|
(3ff2) nspsend: 01 84 00 00 00 C8 9F 00  |........|
(3ff2) nspsend: 01 09 00 00 00 00 00 00  |........|
(3ff2) nspsend: 00 EC 9F 00 01 00 00 00  |........|
(3ff2) nspsend: 00 0A 00 00 00 1E 00 00  |........|
(3ff2) nspsend: 00 00 00 00 00 00 00 00  |........|
(3ff2) nspsend: 00 00 00 00 00 00 00 00  |........|
(3ff2) nspsend: 00 00 00 00 00 00 00 00  |........|
(3ff2) nspsend: 00 EE 9F 00 01 53 45 4C  |.....SEL|
(3ff2) nspsend: 45 43 54 20 52 4F 57 49  |ECT ROWI|
(3ff2) nspsend: 44 2C 4C 2E 50 45 52 53  |D,L.PERS|
(3ff2) nspsend: 4F 4E 2C 4C 2E 4C 45 44  |ON,L.LED|
```

```
(3ff2) nspsend: 47 45 52 49 44 2C 4C 2E  |GERID,L.|
(3ff2) nspsend: 41 43 54 49 4F 4E 44 41  |ACTIONDA|
(3ff2) nspsend: 54 45 2C 4C 2E 41 43 54  |TE,L.ACT|
(3ff2) nspsend: 49 4F 4E 2C 4C 2E 49 54  |ION,L.IT|
(3ff2) nspsend: 45 4D 2C 4C 2E 51 55 41  |EM,L.QUA|
(3ff2) nspsend: 4E 54 49 54 59 2C 4C 2E  |NTITY,L.|
(3ff2) nspsend: 51 55 41 4E 54 49 54 59  |QUANTITY|
(3ff2) nspsend: 54 59 50 45 2C 4C 2E 52  |TYPE,L.R|
(3ff2) nspsend: 41 54 45 2C 4C 2E 41 4D  |ATE,L.AM|
(3ff2) nspsend: 4F 55 4E 54 2C 4C 2E 50  |OUNT,L.P|
(3ff2) nspsend: 45 52 53 4F 4E 2E 4E 41  |ERSON.NA|
(3ff2) nspsend: 4D 45 20 46 52 4F 4D 20  |ME FROM |
(3ff2) nspsend: 4C 45 44 47 45 52 20 4C  |LEDGER L|
(3ff2) nspsend: 20 01 00 00 00 01 00 00  | .......|
(3ff2) nspsend: 00 00 00 00 00 00 00 00  |........|
(3ff2) nspsend: 00 00 00 00 00 00 00 00  |........|
(3ff2) nspsend: 00 00 00 00 00 01 00 00  |........|
(3ff2) nspsend: 00 00 00 00 00 00 00 00  |........|
```

3. 在服务器端的SQL跟踪

在服务器端，使用SQL跟踪可以跟踪SQL语句及其性能指数。

在Oracle Developer/Forms中，通过在命令行中的Statistics选项打开SQL跟踪。

```
IFRUN60 workerskill talbot/george@orcl statistics=YES
```

注意 我未能在Oracle Developer组件(6.0.5.0.2)中通过IFRUN60命令行参数启动statistics(统计信息)跟踪。Oracle公司证明它支持这个功能，可以试一试。

在Reports中，在Before-Parameter-Form触发器中通过SRW过程调用来打开跟踪：

```
SRW.Do_SQL('ALTER SESSION SET SQL_TRACE TRUE');
```

在Graphics中，在 Open触发器中使用类似的调用：

```
Do_SQL('ALTER SESSION SET SQL_TRACE TRUE');
```

注意 也可以通过设置Oracle实例的SQL_TRACE参数TRUE,为所有会话打开跟踪。通常会想这样做，但其影响范围过大。因为这样会在每个人使用服务器时跟踪每个细小的SQL处理过程。

当为应用程序打开跟踪时，服务器产生一个放在转储目录中的跟踪文件。可以通过运行sysmgr和下面的命令来确定这个目录：

```
SHOW PARAMETER USER_DUMP_DEST
```

也可以从Initialization Parameters(初始化参数)窗口中的Instance Manager(实例管理器)中取得该信息。

在Windows NT操作系统中，跟踪文件目录可能是Oracle_Home\RDBMS80\Trace或类似的目录。在目录中找到扩展名为 .TRC并且主文件名是唯一值 (进程ID或类似的数字)的文件，例如ORA00075.TRC。例如，运行Ledger应用程序，产生如下跟踪文件：

```
Dump file D:\orant\rdbms80\trace\ORA00075.TRC
Mon Mar 29 09:50:48 1999
ORACLE V8.0.5.0.0 - Production vsnsta=0
vsnsql=c vsnxtr=3
Windows NT V4.0, OS V5.101, CPU type 586
Oracle8 Enterprise Edition Release 8.0.5.0.0 - Production
With the Partitioning and Objects options
PL/SQL Release 8.0.5.0.0 - Production
```

```
Windows NT V4.0, OS V5.101, CPU type 586
Instance name: orc8

Redo thread mounted by this instance: 1

Oracle process number: 10

pid: 4b


*** SESSION ID:(8.37) 1999.03.29.09.50.48.306
=====================
PARSING IN CURSOR #1 len=32 dep=0 uid=25 oct=42 lid=25 tim=22639233 hv=2162858323 ad='29e4478'
ALTER SESSION SET SQL_TRACE TRUE
END OF STMT
EXEC #1:c=3,e=3,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=4,tim=22639234
=====================
PARSING IN CURSOR #1 len=32 dep=0 uid=25 oct=42 lid=25 tim=22639503 hv=2162858323 ad='29e4478'
ALTER SESSION SET SQL_TRACE TRUE
END OF STMT
PARSE #1:c=1,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22639503
EXEC #1:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22639503
=====================
PARSING IN CURSOR #2 len=132 dep=0 uid=25 oct=3 lid=25 tim=22640182 hv=1199613752 ad='2a54244'
SELECT
ROWID,L.PERSON,L.LEDGERID,L.ACTIONDATE,L.ACTION,L.ITEM,L.QUANTITY,L.QUANTITYTYPE,L.RATE,L.AMOUNT,L.P
ERSON.NAME FROM LEDGER L
END OF STMT
PARSE #2:c=1,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22640183
EXEC #2:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22640183
=====================
PARSING IN CURSOR #4 len=47 dep=1 uid=0 oct=3 lid=0 tim=22640184 hv=1158616671 ad='2b6c630'
select metadata from kopm$  where name='DB_FDO'
END OF STMT
PARSE #4:c=1,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,tim=22640184
EXEC #4:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=4,tim=22640184
FETCH #4:c=0,e=0,p=0,cr=2,cu=0,mis=0,r=1,dep=1,og=4,tim=22640184
STAT #4 id=1 cnt=1 pid=0 pos=0 obj=240 op='TABLE ACCESS BY INDEX ROWID KOPM$ '
STAT #4 id=2 cnt=1 pid=1 pos=1 obj=241 op='INDEX UNIQUE SCAN '
FETCH #2:c=2,e=2,p=0,cr=21,cu=3,mis=0,r=11,dep=0,og=4,tim=22640185
=====================
PARSING IN CURSOR #4 len=156 dep=0 uid=25 oct=3 lid=25 tim=22640616 hv=3424553781 ad='2a121cc'
SELECT L.PERSON,L.LEDGERID,L.ACTIONDATE,L.ACTION,L.ITEM,L.QUANTITY,L.QUANTITYTYPE,L.RATE,L.AMOUNT
FROM LEDGER L WHERE ROWID=:1 FOR UPDATE OF L.PERSON NOWAIT
END OF STMT
PARSE #4:c=1,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22640616
EXEC #4:c=0,e=0,p=0,cr=1,cu=3,mis=0,r=0,dep=0,og=4,tim=22640616
FETCH #4:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=1,dep=0,og=4,tim=22640617
=====================
PARSING IN CURSOR #1 len=14 dep=0 uid=25 oct=46 lid=25 tim=22640800 hv=278968606 ad='2a112a8'
SAVEPOINT FM_1
END OF STMT
PARSE #1:c=1,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22640800
EXEC #1:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22640800
=====================
PARSING IN CURSOR #5 len=155 dep=0 uid=25 oct=6 lid=25 tim=22640801 hv=1725894326 ad='2a0eef4'
UPDATE LEDGER L SET
L.PERSON=:1,L.LEDGERID=:2,L.ACTIONDATE=:3,L.ACTION=:4,L.ITEM=:5,L.QUANTITY=:6,L.QUANTITYTYPE=:7,L.RA
TE=:8,L.AMOUNT=:9 WHERE ROWID=:10
END OF STMT
PARSE #5:c=1,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22640802
EXEC #5:c=0,e=0,p=0,cr=0,cu=1,mis=0,r=1,dep=0,og=4,tim=22640802
XCTEND rlbk=0, rd_only=0
EXEC #4:c=1,e=1,p=0,cr=1,cu=3,mis=0,r=0,dep=0,og=4,tim=22640937
FETCH #4:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=1,dep=0,og=4,tim=22640937
=====================
PARSING IN CURSOR #1 len=14 dep=0 uid=25 oct=46 lid=25 tim=22641088 hv=278968606 ad='2a112a8'
SAVEPOINT FM_1
END OF STMT
PARSE #1:c=1,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22641088
EXEC #1:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22641088
EXEC #5:c=0,e=0,p=0,cr=0,cu=1,mis=0,r=1,dep=0,og=4,tim=22641089
XCTEND rlbk=0, rd_only=0
=====================
```

```
PARSING IN CURSOR #1 len=8 dep=0 uid=25 oct=45 lid=25 tim=22641236 hv=1470906206 ad='2a50020'
ROLLBACK
END OF STMT
PARSE #1:c=1,e=1,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22641237
XCTEND rlbk=1, rd_only=1
EXEC #1:c=0,e=0,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=4,tim=22641237
XCTEND rlbk=0, rd_only=1
STAT #2 id=1 cnt=11 pid=0 pos=0 obj=2488 op='TABLE ACCESS FULL LEDGER '
STAT #4 id=1 cnt=2 pid=0 pos=0 obj=0 op='FOR UPDATE '
STAT #4 id=2 cnt=2 pid=1 pos=1 obj=2488 op='TABLE ACCESS BY USER ROWID LEDGER '
STAT #5 id=1 cnt=0 pid=0 pos=0 obj=0 op='UPDATE LEDGER '
STAT #5 id=2 cnt=2 pid=1 pos=1 obj=2488 op='TABLE ACCESS BY USER ROWID LEDGER '
```

这个文件有些难读。可以使用 Oracle Toolkit profiler program(TKPROF,)来把这个跟踪文件格式化，使其容易使用。在 TKPROF命令行提供输入 TRC文件名和输出文件名。还要输入用户名和口令：

```
TKPROF80 c:\orant\rdbms80\trace\ora00075.trc c:\temp\output.trc
```

格式化LEDGER应用程序的跟踪文件产生下面格式的文件：

```
TKPROF: Release 8.0.5.0.0 - Production on Mon Mar 29 10:4:24 1999

(c) Copyright 1998 Oracle Corporation.  All rights reserved.

Trace file: ora00075.trc
Sort options: default

********************************************************************************
count    = number of times OCI procedure was executed
cpu      = cpu time in seconds executing
elapsed  = elapsed time in seconds executing
disk     = number of physical reads of buffers from disk
query    = number of buffers gotten for consistent read
current  = number of buffers gotten in current mode (usually for update)
rows     = number of rows processed by the fetch or execute call
********************************************************************************

ALTER SESSION SET SQL_TRACE TRUE


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0          0          0           0
Execute      2      0.03       0.03          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      0.04       0.04          0          0          0           0

Misses in library cache during parse: 0
Misses in library cache during execute: 1
Optimizer goal: CHOOSE
Parsing user id: 25
********************************************************************************

SELECT ROWID,L.PERSON,L.LEDGERID,L.ACTIONDATE,L.ACTION,L.ITEM,L.QUANTITY,
  L.QUANTITYTYPE,L.RATE,L.AMOUNT,L.PERSON.NAME
FROM
 LEDGER L


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1      0.02       0.02          0         19          3          11
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      0.03       0.03          0         19          3          11
```

```
Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: 25
********************************************************************************

select metadata
from
 kopm$  where name='DB_FDO'


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1      0.00       0.00          0          2          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      0.01       0.01          0          2          0           1

Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: SYS   (recursive depth: 1)
********************************************************************************

SELECT L.PERSON,L.LEDGERID,L.ACTIONDATE,L.ACTION,L.ITEM,L.QUANTITY,
  L.QUANTITYTYPE,L.RATE,L.AMOUNT
FROM
 LEDGER L WHERE ROWID=:1 FOR UPDATE OF L.PERSON NOWAIT


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0          0          0           0
Execute      2      0.01       0.01          0          2          6           0
Fetch        2      0.00       0.00          0          0          0           2
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        5      0.02       0.02          0          2          6           2

Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: 25
********************************************************************************

SAVEPOINT FM_1


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        2      0.02       0.02          0          0          0           0
Execute      2      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.02       0.02          0          0          0           0

Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: 25
********************************************************************************

UPDATE LEDGER L SET L.PERSON=:1,L.LEDGERID=:2,L.ACTIONDATE=:3,L.ACTION=:4,
  L.ITEM=:5,L.QUANTITY=:6,L.QUANTITYTYPE=:7,L.RATE=:8,L.AMOUNT=:9
WHERE
 ROWID=:10


call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0          0          0           0
Execute      2      0.00       0.00          0          0          2           2
Fetch        0      0.00       0.00          0          0          0           0
```

```
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      0.01       0.01          0          0          2           2

Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: 25
********************************************************************************

ROLLBACK
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.01       0.01          0          0          0           0

Misses in library cache during parse: 0
Optimizer goal: CHOOSE
Parsing user id: 25


********************************************************************************

OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        7      0.07       0.07          0          0          0           0
Execute     10      0.04       0.04          0          2          8           2
Fetch        3      0.02       0.02          0         19          3          13
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       20      0.13       0.13          0         21         11          15

Misses in library cache during parse: 0
Misses in library cache during execute: 1


OVERALL TOTALS FOR ALL RECURSIVE STATEMENTS

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        1      0.00       0.00          0          2          0           1
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        3      0.01       0.01          0          2          0           1

Misses in library cache during parse: 0

    8  user  SQL statements in session.
    1  internal SQL statements in session.
    9  SQL statements in session.
********************************************************************************
Trace file: ora00075.trc
Trace file compatibility: 7.03.02
Sort options: default

       1  session in tracefile.
       8  user  SQL statements in trace file.
       1  internal SQL statements in trace file.
       9  SQL statements in trace file.
       7  unique SQL statements in trace file.
      89  lines in trace file.
```

注意 可以通过设置Oracle实例的TIMED_STATISTICS参数为TRUE，来打开时间统计信息(timing statistics)。如果这个参数为FALSE，在统计信息表中只能看到0。应该清楚，

打开这个参数会明显地降低系统的速度，所以，不要在产品系统中打开这个参数，除非你渴望让所有的用户都受到影响。另外，应该查看Enterprise Manager(企业管理器)中新的Oracle Trace选项 —— Performance Pack(性能压缩)选项的一部分， Enterprise Manager中提供了更多跟踪Oracle应用程序的工具。

第三部分介绍了如何使用 Oracle Developer的高级工具编写代码和调试应用程序。系统地使用这些工具，可以建立一个高质量的系统。通过系统中的可重用的代码，可以建立许多应用程序。下一部分将介绍如何把应用程序从开发环境中移到现实的环境中，交付客户使用并且进行有效维护。另外也将介绍现实的环境中如何把应用程序与其他软件结合在一起，如何与其他人的创造性成果和你自己的有效方法相结合。