第二部分　Oracle Developer教程

第3章　数据库、表单和报表共同工作

本章为下面几章的教程内容作准备。它建立教程使用的数据库，然后介绍在 Oracle Developer中建立表单、报表和图形的一些规则。

3.1　建立Talbot数据库及其应用程序

在我们创建数据库以前，让我们回顾一下历史， Talbot农场是进入新世纪时在新罕布什尔州的一个工作农场。这个农场位于靠近Keene镇和Edmeston镇的新罕布什尔州西南的乡村地区。以前的一本书(George Koch和Kevin Loney著《Oracle 8完全参考手册》，曾经使用Talbot农场作为它的例子的一部分。该例子非常适合 Oracle Developer教程的需要。

这个例子出自被George Koch发现的一个旧的总帐帐簿，上面有从 1896年开始直到1905年的记录(图3-1)，这个帐簿是Dora Talbot农场的普通帐簿，可能是由他的儿子 George B.Talbot保存。

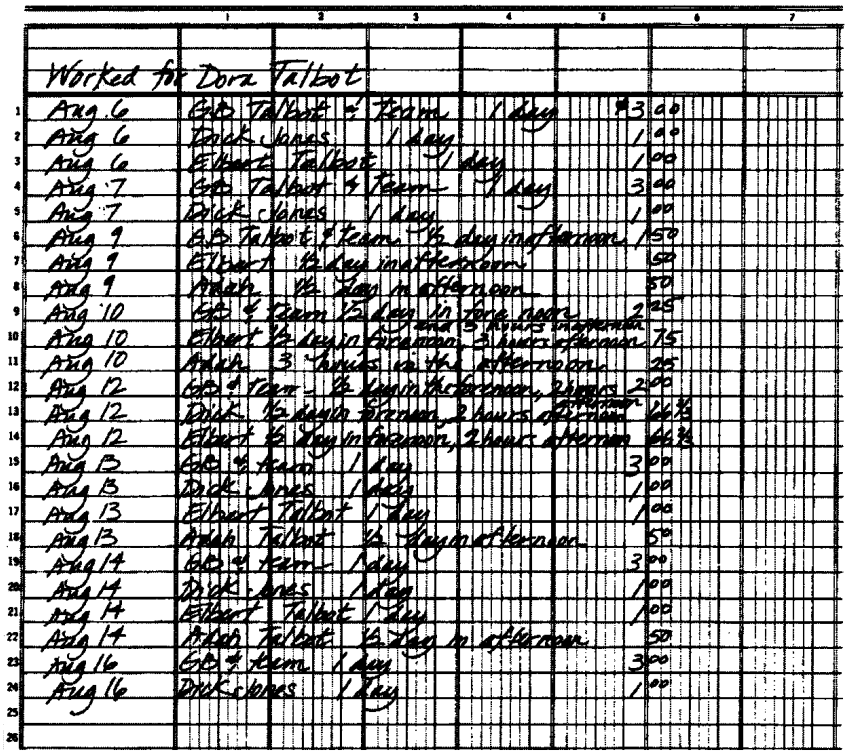

图3-1　Talbot帐簿的样本页

这个Ledger簿包括销售、购入、劳动者的薪金支付和材料的采购：所有的按照顺序的工作业务的事务。它还包含有另外一些有用的信息，如在农场工作的临时工的地址清单。

这个帐簿启发George Koch把它作为在关系数据库之前人们是如何记帐的一个好例子——而且作为在旧式的帐簿记录和现在的关系数据库的记录之间的相似性的有力说明。他使用这个帐簿作为一个小的但是极重要的关系数据库的例子。这个帐簿可以很容易变成一个关系表。最初的数据库为如何写 SQL提供了一个相关的例子。这本书进一步发展了这个想法，并将Talbot帐簿用于Oracle Developer开发应用程序。

《Oracle 8完全参考手册》实际上并不与开发应用程序的任务有关，而只是和 SQL和Oracle的教学有关。那本书的例子虽然正确有用，但未坚持那本书提倡的过程和设计建议。在这里，你将看到一个更完全的、带有全部完整的约束的 Talbot数据库版本，它将作为在这本书中采用的综合例子。

这本书还自由地扩展了过时的事物，使某些 Talbot农场范围之外的东西进入农场的范围，为了现代数据处理的需要而扩展，将其变成扩展了的现代商业。 Talbot的操作可能是使用牛和雇员去挖坟墓或打马掌，但是这本书却要面向下个世纪——无论是什么样的世纪，要把它作为是现代共同农场的一个例子。 Talbot农场将增长，而且它的顾客和供应品遍及全世界。对全球的问题要求全球的方法，故将 Talbot扩展到访问因特网。因此，请暂停对历史的怀疑，这不过是用Talbot的例子说明如何在自己的业务中有效地使用 Oracle Developer而已。如果有运气，它将继续存在(只是至少要保藏一个Oracle Developer应用程序，为了在2095年被发现用来作为器官数据库或量子数据库的例子，或者作为未来才有的无论什么技术的例子)。

3.1.1 Talbot的需求

任何数据库应用程序的工程均从应用程序系统的用户要求开始。 Talbot的要求集中在两类查询和维护功能上：总帐以及在 Talbot工作的人员的名单，其中名单包括他们的地址和技能。

> **注意** 在现实世界中，有许多需求，但可能有的需求超出了范围，需求的内部定义和外部定义超出了本书中教程和参考资料的范围，本章的材料提供了在现实系统中可以看到的这种要求和数据库的一个样本，但是只够在下面几章中提供不同的指导。企业广泛的发展过程将使用比较复杂的要求定义技术，如在 Oracle Designer的FHD图，有关Oracle Designer和它与Oracle Developer的使用的基本描述，请参看第15章。

第一步是评估工程的可行性。使用有限的一组要求，而且把要求的焦点放在重要的数据库上。有各种理由相信，一个简单的 Developer数据库应用程序系统完全可以满足 Talbot的要求，因为它没有什么罕见的技术要求，所以 Oracle Developer(可以建立非常大的系统的企业工具)可以容易地处理这些要求。

第二步是要得到对问题的陈述。对这个要求的简略回答是图 3-1中所说明的帐簿，它包含有Talbot帐簿所保存的在几年时间内生成的数据和业务逻辑。本书用不同技术将 Talbot的信息处理为对它当前的要求。为什么使工作内容发生了变化？因为 Talbot农场不是静止的，它们在迅速地扩展农业和工业，将农场和农业加工业扩大到全球范围。人工的 Ledger系统是为单个农场工作，但是不能为全球的农场业工作。因此，问题是使人工 Ledger系统自动化，用比当前系统大得多的能力支持企业资源计划的解决。而新系统特殊的地方又是什么呢？

Talbot需要维护和使用包含现金交易的数据库，包括货物买卖的金额、工时费的支付金额

以及收入的金额。Talbot还需要维护和使用包含有Talbot的工作人员和他们的地址及技能的数据库，如何转换这个功能呢？本部分将介绍一些需求的思路，比如 Talbot那样的公司的需求，以及你所需要得到的那些信息。下面的过程并不像真实的需要分析那样进行详尽无遗的讨论。

最初的推测可能是：帐簿事务是数据库的中心。在某种程度上的确如此， Talbot非常注意保持跟踪所花费和所接收的现金数量。然而，如果要更深地探究，那么就会开始认识到在事务中和人员应用中所涉及到的人的重要性。人员将系统的财务和人员的资源方面结合在一起。一个不成熟的决定是，重点表示人，将劳动者随同进入到 Talbot事务中的人员一起放入单个表中。

首先让我们列出需要强调的软件应用程序的功能。

- 功能：表示所有的人员和机构，这个机构把 Talbot农场约束在一个主要的列表上，这个表格可用来查询和维护帐目及人员。

对于这个功能来说，最根本的是可利用这个列表来管理帐目和人员。还有一点是，与Talbot有货币业务关系的人员要在这个列表上。而且希望应用程序能够容易地修改列表中的错误。

- 功能：跟踪Talbot涉及的货币业务，包括事务的种类(买、卖、支付、收入)、项目说明、单位的数量、每个单位的估价以及事务的数目。这些信息必须可以查询，而且必须可以被审查。

为了能够进行审查，重要的是所有事务都应存在数据库里面，而且是对独立的事务进行修正。极其重要的是，在任何时候，不论任何理由，事务都不能改变或消失。希望应用程序根据质量和价格自动计算事务的数值。用户则希望能够通过数据元素将查询的事务排序。

- 功能：为了联系和支付工资，要保持劳动者的地址清单。

重要的是，地址应该是当前的地址，而且必须正确，希望能够为所有的劳动者产生记入标记。

- 功能：要保持劳动者技能和收入的清单，以便分配工作。

希望所有的劳动者都有技能登记项，而且各个登记项可以评估他们在某种技能上的能力。

还有一个配置的要求：应用程序必须可以为 Talbot农场的整个网络和子公司使用。 Talbot的IT部门相信，通过公司的内部和全球因特网来配置应用程序将满足广域网的需要。因此，所有开发的应用程序必须适合在因特网上使用。给出了这些要求后，就可以进行数据库设计了。

3.1.2 示例数据库

不先进行数据库设计，要使表单、报表和图形形象化是困难的。因为它们都反映了 Oracle Developer的基本目的：允许终端用户操作数据信息。数据库设计要将问题叙述清楚。格式语句对定义表单、报表和图形是很重要的。有一个示例数据库也可以使一个原型的开发工作比较容易。

附录A有一个完整的、带有数据定义语句和全部的数据的数据库。本部分内容解释数据库设计，并且讲述在以后的内容中解释 SQL及数据所需要的知识。

图3-2展示了定义Talbot数据库的统一模型语言 (Unified Modeling Language, UML)图表，如果你以前没有见过 UML分类图，以下是有关的图形约定。

- 种类(方框)：表示数据库内的表，用表的行表示种类的对象。种类的名字在顶部，种类的属性在底部。
- 关联(直线)：表示种类之间的联系，某些关联也表示数据库的表（例如，多对多联系以及和它们自己特性的联系）。从地址到住所的关联上的小黑菱形表示拥有权（正式的UML术语是composite aggregation(合成归并)），住所行拥有地址行，所以如果删除住所行，则必须删除地址行。
- 属性(方框内的文本)：表示对象的数据特性。
- 角色(名字旁边有关联线)：表示这个种类在关联中扮演的角色。
- 角色的多重性(1和*标记)：表示在关系中和另外的实体关联的对象的数目，一个或多个。

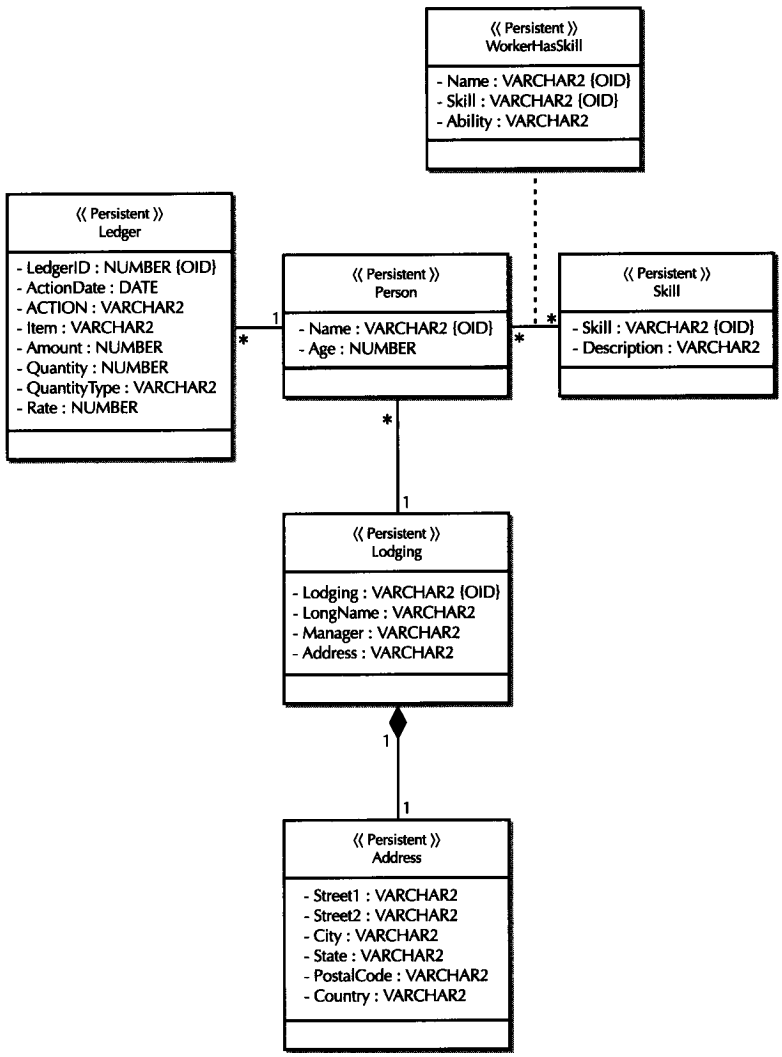

图3-2 Talbot数据库的UML种类图

有五种对象：

- Ledger：Ledger里的登记项。每个对象涉及的某种单个事务，用唯一号码标识(Ledger ID)。

- Person：与Talbot打交道的人，用名字标识。
- Lodging：某人居住的地方，由住所名标识。
- Addres：住所的邮政信息。
- Skill：人的技能，用技能名标识。

模型中有四个关联：

- refers to(涉及的)：把Ledger登记项和人员联系起来， Ledger登记项涉及到人员作为卖方或收款人或付款人。
- live at(居住在)：把人和住所联系起来，人居住在住所。
- has address(有地址)：把住处和邮政地址联系起来。
- worker has skill(工人技能)：把某个人和一种或多种技联系起来，或者将一种技能和一个人或多个人联系起来。能力是个单一属性，即表示工人在技能方面的能力 Ability，这是一个关系属性。

头两个关联是可以选择的一对多的关系。 Ledger登记项准确地指向某个人，而一个人可能涉及到Ledger里的多个登记项。一个人居住在某个住所内，而一个住所可能有多个人居住。在关系数据库里，表示人员的数据行包含有和住所有关的外关键字列。而表示住所的数据行则包含有与人有关系的外关键字列。在对象关系数据库中可以有外关键字，或者有连接这些数据行的参考(REF列)。有关这两种方法的完整方案信息请参看附录 A。

第三个关联需要一对一的关系。一个住所有一个确切的地址，而每一个地址则精确地对应一个住所。在关系数据库中，很可能要把地址列移到住所表中。在对象关系数据库中，可能在住所类型中创建一个地址类型和该类型的列，有关这两种方法的完整的方案信息参看附录 A。

第四个关联是可选择的多对多关系。一个人可能有几种技能，而一种技能则可能对应到多个人(多个人可能具有同一种技能)。功能只对应到工人身上 —— 这个特殊的数据库并不表示这个事实，因为它有损于这个例子的清晰度。

将这些转换成关系数据库或对象关系数据库，要采取以下步骤：

1) 为每个实体(Ledger、Person、Lodging、Address和Skill表)创建一个表或对象类型，根据属性创建列，并且给出它们的主关键字列 (分别是LedgerID、Name、Lodging，Address和Skill)。

2) 为每一个多对多关系(Worker Has Skill)创建一个表或对象类型。创建任一列(Ability)为表的关键字，并且是参与这个关系的实体的两个关键字的主关键字 (根据Person和Skill分别是Name和Skill)。

3) 对于每一个一对多关系，在多边关系的表增加一列，这与单边关系的表的主关键字有关(在Ledger表上的Person涉及Person表，而在Person表上的Lodging涉及Lodging表)。对于关系数据库用标准的外关键字，而对象关系数据库则用参考 (REF)列。

4) 对表之间的一对一关系，变换的理解有点困难。首先，要决定这两个表中哪个是关系的拥有者。例如，在Talbot数据库中，住所拥有地址(那就是图3-2上的小的黑菱形表示的意思)。创建Lodging的关联表，要从Address中移动一列到Lodging表上，而且无单独的Address表。要创建这种结构形式的对象关系，先创建 Lodging和Address对象类型，然后，在类型为 Address的Lodging表上创建Lodging Address列。

在这个示例数据库中，要创建关系数据库或对象关系数据库，使用这些规则就足够了。

在比较复杂的数据库中，还有另外一些事情使人烦脑。比如，两个以上的实体的关系以及带有属性的一对多关系。使人烦脑的是还有更多的约束，而且还要考虑数据类型，有关这些问题的详细讨论请参看我写的书《Database Design for Smarties》(Morgan Kaufmann, 1999)。

如果查看附录A中的SQL，将会看到它产生一个附加的对象：序列。 Oracle提供宣布序列的能力，这是一种工具，连续地产生唯一的整数。 Talbot数据库利用这个序列在INSERT语句中为Ledger表产生Ledger标识符ledger ID。如果要求Talbot数据库可以移植到其他的数据库管理程序中，这个数据库将不包含这个序列，因为它是非标准的。不能依靠有 CREATESEQUENCE语句或NEXTVAL伪列(pseudo-column)来返回下一个序列号。 Talbot数据库有一个重要的约定，只使用Oracle。所以它可以充分利用Oracle数据库管理系统的许多非标准的特性。

注意 如果你对使用UML来设计数据库感兴趣，请参考我的书《Designing Databases for Smarties: Using UML to Design Database》(Morgan Kaufmann, 1999)或Paul Dorsey的书《Oracle 8 UML对象建模设计》。Oracle Designer工具现在对模式定义支持UML，所以还应该参考Dorsey和Koletzke著《Oracle Designer /2000 Handbook》。

3.1.3 Oracle Developer工程管理

在作任何别的事情以前，应该利用 Oracle Developer的Project Builder建立一个工程。

1. 用Project Builder产生一个工程

Project Builder是Oracle Developer工具集的一个组件，它让你把文件组织到工程里面，并且提供管理它们所需要的工具。

对于Talbot系统，我们将建立两个工程，一个对 Ledger系统，而另一个对 Worker Skills系统。要根据许多因素来决定工程的边界，包括应用程序的时间确定、你的机构如何给人们分配任务、在工程之间有多少新软件开发要共享，等等。我们将假定这两个应用程序是独立的，一个由Financial Services的信息服务(IS)小组开发，另一个在Talbot农场的人力资源内部开发。

1) 要创建一个工程最好的“赌注”是 Project Wizard。在Project Builder图标上双击，将出现如图3-3所示的对话框。

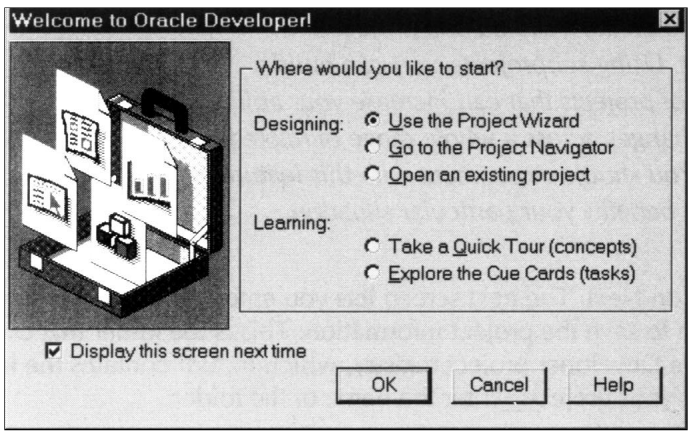

图3-3 点选Project Builder图标产生对话框

2) 在OK 按钮上单击，启动Project Wizard。在欢迎屏幕以后，出现Project Wizard对话框，

提示输入工程注册文件名，这个文件包含有关工程文件的所有信息和它们之间的相关性。输入文件名(见图3-4)。

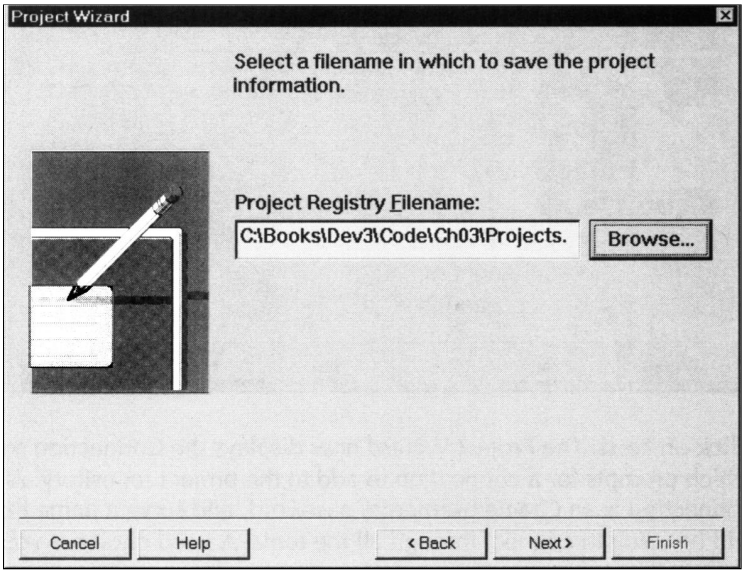

图3-4 Project Wizard对话框

注意 当有某些被定义的工程时，Proiect Wizard将插入一个屏幕，提问你是要建立一个独立应用的工程还是建立一个子工程。如果使用子工程，则可以建立工程的层次结构，可以增加管理整个关联工程范围内的变化的能力。你应该利用这个特点做试验，看其对你的特殊情况有什么好处。

3) 单击Next按钮。出现的屏幕提示输入工程的题目和保存工程信息的文件夹。该文件夹保存Oracle Developer工程注册，同样也包含所有的工程的信息。输入文件夹的名字 (见图3-5)。

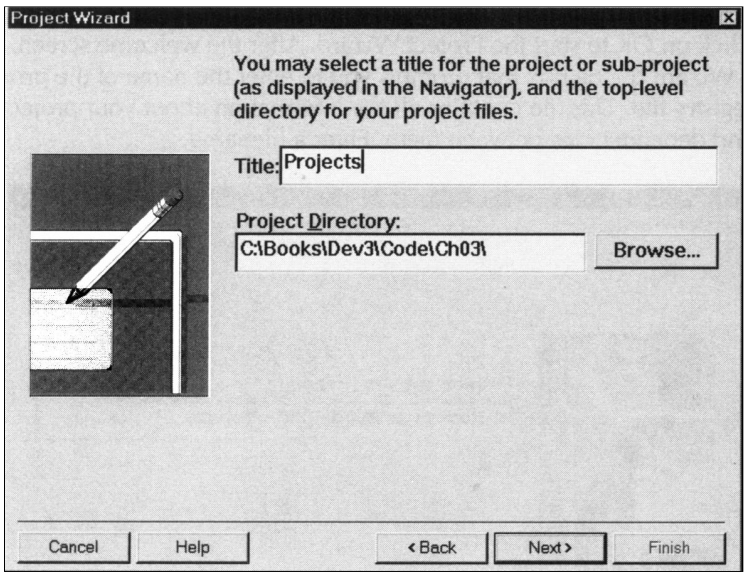

图3-5 输入文件夹的名字

4) 单击Next按钮，则Project Wizard显示Connection屏幕，提示在工程资料档案库中添加连接。连接内容有Oracle用户名、口令和服务名，你将通过全部工具使用连接。一个好的选择就是开发的数据库。在Talbot例子的情况下，连接就用talbot作用户名，如图3-6所示。

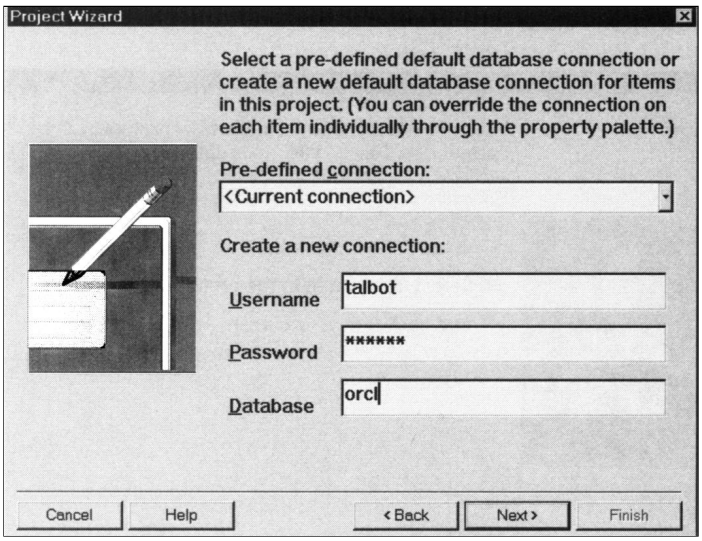

图3-6 在Project Wizard对话框中以talbot为用户名

5) 单击Next按钮，进到下一屏幕。提示输入本人的名字 (你的名字、工程管理员的名字，或者任何一个名字)和对工程的缺省连接 (如果有的话)。还可以在这里输入一段注释来说明这个工程(见图3-7)。

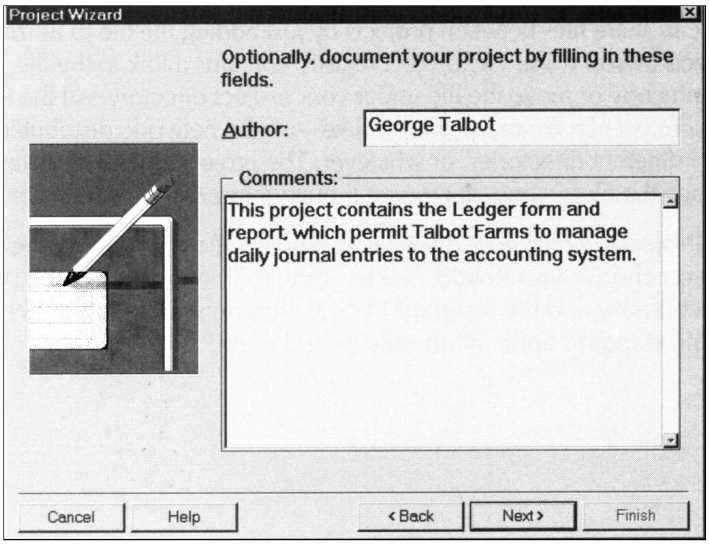

图3-7 在Project Wizard中输入注释

6) 单击Next按钮，进入最后一个Project Wizard屏幕。这个屏幕提供向工程中放置内容的三个选择：什么也不放，从 Open对话框中选择的一组文件或先前在工程文件夹中选择的所有文件(见图3-8)。

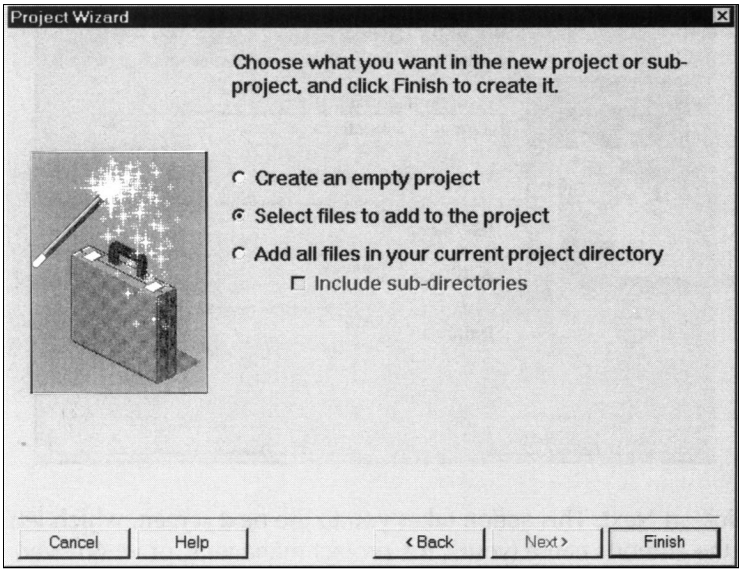

图3-8 最后一个Project Wizard屏幕

7) 单击Finish按钮，创建工程。

将文件加到你想将它们加入进去的那些工程中，就可以在工程之间共享文件。工程注册包含有对这个文件的连接，它并不在工程目录下拷贝或移动这个文件。工程中的所有文件可以出现在你喜欢的任何地方 —— 在网络上，或分散在许多不同的目录里，或无论什么地方。工程给你一种方法，即只在一个地方管理文件，即使这些文件并不在同一个地方。

Project Wizard接下来显示一个标准的打开文件对话框，提示选择要加入的文件。找到和选择标准对象库和标准的PL/SQL库，它们是Talbot IS创建用来保存标准应用程序对象和代码的(见图3-9)。

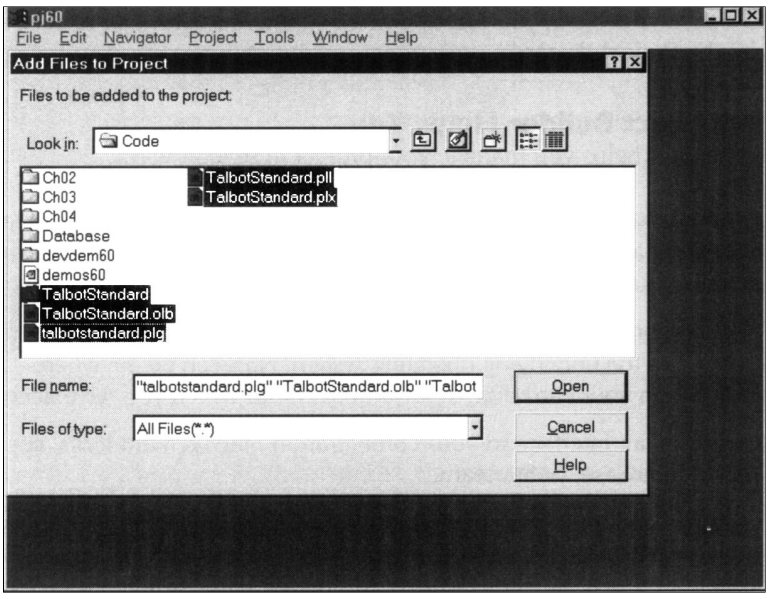

图3-9 标准的打开文件对话框

8) 单击Open按钮，Project Builder将被选中的文件加到工程上，这时工程已经准备好，并等待新的应用程序(见图3-10)。

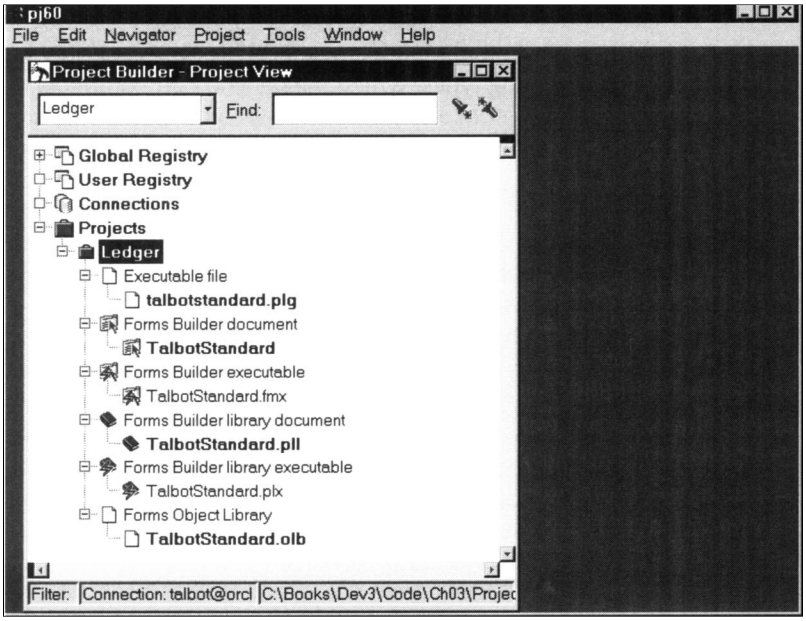

图3-10 Project Builder将选中的文件加到工程上

要添加第二个对象，只要双击Project结点，用Project Wizard启动另一个会话过程。

2. Project Builder如何帮助你

Project Builder以几种方法帮助管理工程：

- 启动各种各样的建立器。在 Project Builder窗口旁边的工具条上的建立器工具上单击，或者通过双击先前创建并加到工程中的文件。
- 可以不受基本操作系统目录结构的约束去组织文件。文件可以在任何地方——在网络上、在硬盘上，或者在所访问的任何其他地方。
- 为配置管理工具提供界面，比如PVCS、ClearCase或Starteam
- 提供一种方法，指定某个文件依赖于什么其他文件，从而自动创建多个文件。你可以改变PL/SQL程序库，例如，用一条命令可以重新建立依赖于那个程序代码的所有的表单、报表和显示。
- 通过Deliver(传递)行动使部署过程自动化，而你可以编程做适当的事情，比如把将可执行的文件压缩为一个可交付的文件。
- 因为是完全可编程的，故可以为新类型和已经存在的类型增加文件类型和各种操作。这就意味着你能够完满地定制环境以实现所希望的自动化。本书不涉及定制，有关细节请参看Project Builder的文档资料。

3.1.4 开发表单

这部分内容讲述如何根据要求着手设计各种表单和报表的入门知识。

第一个表单是Ledger表单，它着眼与Ledger有关的功能。这个表单作为会计部门与 Ledger

表的主要接口。这个表单必须完成以下任务：

- 表单必须允许记帐员进入会计事务。每个事务必须有日期、项目说明、活动、数量和单位(包括单位的类型、价格和总额)。
- 表单必须产生Ledger标识ledger ID以标识每一个事务。
- 事务必须通过人的名字与已经在数据库内存在的人相连接。
- 表单必须用数量乘以价格计算总额值。
- 表单必须防止对数据库中的事务进行任何更新或删除。
- 表单必须允许会计利用数据元素对事务进行特定的排序。
- 表单必须遵循适合于因特网使用的准则。

表单的域以一种可使用的格式表示 Ledger表的结构。首先要做的选择是否安排表单每次处理一个事务或多个事务。维护需求表示一个事务，而查询需求表示多个事务。单事务屏幕将域分布在表单窗口的各处，多事务屏幕则具有像电子表格式的包含字段域和数据行的网格。在本例中，选择用多事务屏幕。这种表单可以适应查询和维护。因为可以使部署图表的大小和形状适合所要求的窗口的大小。如果有更多的域，或者所需要的域比较大，那么可以建立两个表单，一个用于查询，一个用于维护。

在输入表的Person域中创建值列表(List of Values, LOV)对话框或下拉列表，可以用来满足与人员连接的要求。

第二个表单是Enter Worker Skills表单。这个表单表明第二种主要的表单是主 -从式表单。这个表单的功能是维护劳动者技能数据库，并且给管理员提供一种查询各个工人的技能的方法。引用报表也可以完成后一个任务。随后的部分给出 Ledger Summary报表的示例。

因为需求的多样性和多对多关系，最好的解决办法是用独立的表单来维护各个表 (Person、WorkerHasSkill和Skill)。Person表单包含有Person数据列， Skill表单有Skill数据列，而Worker Skill表单中有关系列(来自Person列、Skill列和Ability列的关键字)。在Worker Skill表单中，不需要包括人员的年龄。查询 Skill列和Description列的值，主关键字是叙述性的，足以作为Worker Skill表单的简短说明。可能并不需要显示 Skill Description。这样，可在表单上显示Person Name、Ability和Skill这三个域。

因为这个表单表示了一种关系，可以使用这种主 -从关系来建立登记项的自然层次结构。主记录是只读的，查询它可以找出有指定技能的人。从记录把多种技能与某个人连接起来。可以通过在这些域上的对话框List of Values或下拉列表来连接，那就意味着多次显示同一个人，以及允许多个人在同一时间得到指定的技能，那样似乎有些混乱。用层次安排则比较自然，较少混乱，实现也不复杂。对用户看起来也比较简单。因为 Person域是单独的，与技能信息的有关的两列的表不如人员和技能信息的部署图表类型的表复杂。

其他表单也许包括与Lodging表有主-从关系的Person表单，以及带有可用来输入技能的简单表的Skill表单。在表单设计过程中，可能将这些表单组合成一个更通用的人力资源表单，让Talbot的管理人员用与 Person有关的多个主 -从关系去操作所有与人员有关的信息。通过使用tab canvases(标签页画布)，这个表单可以简化它的界面。 tab canvases是一种用户界面技术，可让你把几个元素放入单个视图，并通过在文件夹标签页上单击，在它们之间移动。

在创建这些表单的过程中，开始认识到要求用一些标准来确保所有这些表单给用户提供一个一致的接口。这里有一些例子：

- 每个表单表示一个基本的功能，使用表单的标题来概括这个功能并把这个标题放在窗口的标题中。
- 使数据块的滚动条放在这个块的右边，使它和滚动窗口的通用格式一致。
- 使用LOV(值列表)而不使用下拉列表来表示多行表中的选择项。
- 对字型、提示的放置和其他可视化特性使用用户界面标准。为了表单在因特网上的使用创造最好的效果。

3.1.5 开发报表

虽然Ledger表单对事务数据和特定查询是足够了，但是， Ledger信息的各个用户需要能够更容易理解会计汇总信息。这就需要报表而不是表单。虽然它们的区别很细微。通常，当想用一种标准的格式查询数据库的信息，但不需要操作数据时 (例如改变数据或增加新数据)则使用报表。利用Oracle Developer Server可以通过因特网使用报表，就像使用表单那样容易。所以当用户需要通过查询得到最新的信息时，应该考虑使用报表。而当他们需要增加或改变信息时，考虑使用表单。

对于那些由Talbot之外的财务人员制作的现时的会计报表， Talbot的设计者可以容易地自动产生类似的报表，而不需要在会计年审上核实这些数字而花费更多的钱。 Talbot的会计将不再需要制作报表(和承担产生报表的任务)，他们只需要做他们自己的工作。

这些报表按照事务对人的影响来汇总这个 Ledger。审查会计通过按年月顺序列出的人和对各个人的总计来查询个别事务。

付款的会计部门要按照家庭地址给工人发薪金，这就需要报表。因为 Persennel也希望能够把公司的时事通信和其他信息记入给工人的家庭地址。要做这个工作， Talbot需要根据Person和Lodging表产生记入标签。

唯一需要对记入标签报表做的主要决定是标签的大小。记入标签现在流行使用激光打印机，它包含一个相同的标记的设计图，通常每页两至三列。 Talbot Lodging Mailing Label报表需要一个设计图，说明希望 Personnel的名字和住所地址如何出现在标签页中。还可能希望能够把标签的大小当作报表的下一个参数，以及可以使用带有不同大小标签的报表。

3.2 建立表单和报表的一些规则

在商业上，表单和报表和人的关系就像图书和读者的关系那样自然。任何尝试过写书的人都会告诉你写书并不像读书那样简单。表单和报表的设计是一种艺术，它要求有丰富的经验，要了解人们对如何使用这些东西的看法。本部分内容概括了在设计 Oracle Developer的表单和报表中的一些主要原则性问题，特别是如果你打算在因特网或在内部网上使用它们的话，更要注意这些问题。

下面是五个与表单和报表设计有关的基本原则：

- 表单或报表必须用尽可能有效的方式将信息传递给用户。
- 表单或报表必须有多方面的适应性，以适应预期用户的经验水平、知识水平和不同的需要。
- 表单或报表必须尽可能地简单，而不仅仅是稍微比较简单。
- 表单或报表必须和人的响应速度和流程相适应。

■ 表单或报表必须通过适当层次上的反馈为它的用户提供辅助功能，当需要时，应提供帮助和提供消除错误的能力。

这些原则大多数适用于纸张表单和报表，同样也适用于联机表单和报表。适用于税款的表单(除了装有支票的税款表单以外的任何税款表单)将证明艺术在任何地方都不能接近它的顶点，以下对设计的建议可能会帮助你领会到如何使得表单更好。学习这类设计最好的方法是先看一些优秀的设计书籍，然后动手去做，而且观测用户使用这些设计的容易程度。

概括起来表单和报表设计有三要素：理解、学习和结合。理解表单或报表加速信息传递的能力，理解来自表单或报表的内在的简易性。学习如何根据表单和报表的多方面的适应性和简易性以及给予帮助的方法来使用表单和报表。如何将报表或表单安装在一起加速信息的流动以提高其性能，特别是可以让用户迅速可靠地输入信息。这些要素涉及设计的各个方面，每一个要素在用户交互系统中都起着它的作用。当然，表单和报表在某些方面是不相同的，但是它们的布局和涉及逻辑完全类似。它们的不同之处在于表单是交互式的，而报表有重复的页结构，除此之外，它们是相同的。这里讲述的设计要素，与数据登录项有关的只适用于表单，其他的要素，对报表和表单都适用。

3.2.1 通信

为了有效地通信，表单或报表只需要提供所需要的信息。要做到这一点，首先必须精确地确定有那些信息？用户将如何使用它们？有时候你会发现，将信息分成两部分，在通信上没有损失什么，却得到了简易明了的信息。这样，使用户比较容易地了解要传递的信息，从而改善了通信的性能。

作到有效通信有若干技术：尽可能少用术语。如果可能，尽量使用图形。使用标题说明或标签，避免多余信息，而且信息要一致。

1) 尽量少用术语　避免代码词，缩写词和错别字 (误用的字)。在表单或报表以及数据库的数据中使用明白易懂的英语 (或本国的语言)。如果允许向数据库输入数据，那么，可能需要提供某种编辑功能，将输入的表达为便于通信的形式。

2) 使用图形　如果图片和图形传递的信息正确，显示的层次正确，则使用图片和图形而不采用文本和表。第8章的“ 有效使用图形 ”部分比较详细地讨论有关的问题。

3) 使用标签　使用标题说明或标签能够使表单或报表的数据含义清楚。一些数据或图形可以独立，而且便于通信，但是，这种情况极少。经常可以发现标题上的几个字会立即把图形或数据的用途和性质传递给用户。

4) 尽量使冗余度最小　检查表单是否有冗余的信息。尽量减少用户输入 /输出表单或报表数据必经路径的数量。在保证整个图形的接收时，冗余技术将改善通信的性能。在吸收信息困难时，使得通信难以理解。移动数据到数据库或理解报表的内容所必须走的路径长度和复杂程度将确定通信的水平。

5) 有节制地增加多样性　有节制地利用差别来强调重要的数据或重要的指令。利用不同的字形(例如，黑体字，不同的磅值或不同的字体)可以区别标题或有关的数据，就像图形的布局那样(例如，斜体的三维区域)。但是差别太大会导致复杂性的增加，使接受信息或记忆关系的能力减小。在屏幕周围到处都有许多不同的字体、不同的颜色、有趣的符号和图标，这可能使用户感到很有兴趣(短暂地)，但是对通信并不一定有效。还有，每个平台都有风格样式的

要求。在不同的对象上有不同的限制范围，而且差别很大，通常会使应用程序面对互相矛盾的平台位置。

3.2.2 适应性

适应性，是指对于具有不同需要的用户，应用程序适应于他们所需要的程度。需求并不限制一个真正灵活的系统。就是说，不管计划中用户将做什么，他们可以做他们想做的任何事情。

在真实的数据库应用程序中，情况太复杂，故难以有真正的适应性。只能用编程语言或非大多数用户选择的其他类似的复杂性水平上才能得到真正的适应性。如果愿意使用 C语言或COBOL语言，那么，可以在一定的限度内做想做的任何事。但是，一个最终用户做不到，因为这些技能超出了他们的知识和能力范围。

表单或报表为最终用户提供了细心构造的、适应性较低的界面。然而，在这个细心构造的范围内，应该使用户有灵活使用应用程序的能力。

要考虑的主要的适应性问题是经验。最终用户有什么样的知识水平？当用户对应用程序的使用学得比较多时，应用程序在可用性方面可以调节吗？通常，简单的和带帮助的层次适合于经验少或知识少的用户水平，使其成为有经验的用户。应该提供一些对有经验用户有帮助的方法，比如使用关闭特性，与通过访问列表得到输入项相反而使用直接输入项、特殊按键等等。如果不能通过一个应用程序做这所有的事情，那么为不同知识水平建立多个应用程序，并且使这些应用程序相容，使得一个新用户变得比较有经验时，可以迅速地适应那些供经验用户使用的应用程序。

另外一个要考虑的主要的适应性问题是指通过可能性的网络的不同路径的潜力。人们看事物的观点不同，不应该在不需要的地方限制他们。如果输入信息的次序不重要，则应该允许可以采用任何次序输入数据。如果可以有几种方法输入一个值 (例如，以各种各样的格式输入日期)就不要限制用户只用一种格式，因为很容易检查是否正确。提供一些选项便于不同的用户用他们自己想用的方法，而不是你想他们用的方法去做事情。

3.2.3 简单性

减少复杂性就增加了简单性，反之亦然。有许多不同的原则可以用来减少复杂性。

- 可以对窗口和对话框层上的用户隐藏复杂的程度。
- 可以使得用户需要作的公用的操作简单而且容易，即使是实现这个要求比较困难。
- 可以利用人的认识能力的限度作为指导原则来限制界面中成分的数目 (在七个成分的基础上加/减两个)。
- 可以使界面成分尽可能一致，以改善用户领会这些成分的能力。
- 通过一致性原理把成分编组，使界面成分有聚合力，并且将用户需要的所有数据保存在一个地方。
- 可以使冗余度和路径的复杂性尽可能小，这也可以改善通信的性能。
- 可以使用设计布局和格式化标准，引导用户到正确的地方，使得好像是一个自然的过程。
- 通过确认和结构化输入技术 (例如值列表、缺省和智能触发器)使数据输入自动化，改善

可靠性。

(1) 布局标准

表单和报表应该按照用户的语言和文化情况而自然布局。对于英语文化背景，就意味着把域的起点放在页面或屏幕的左上方而且从左到右，从上到下延续。

表单应该遵循平台所使用的布局标准。例如，大多数图形用户界面把表单的标题放在窗口顶部的中央，大多数平台的滚动条和状态条也有标准的位置。

为了强调，应该选定几条重要的约定，在布局中少量使用它们。记住，强调的地方越多，则被强调的地方好象是越不重要。可以使用不同的布局选项来区别表单或报表的不同元素。使标题和标签与数据不同，而且从它们概括的数据中区别汇总的数据和总计。

在文本中使用混合的字体(正常的是使用大写字母，就像英语的句子那样)比较容易读。也可以在标题或标签中使用大字字母，但是这本书建议不要这样做。在文本的标题和题目行内，避免缩写词。

(2) 结合和一致

通过把元素编组并通过在屏幕或页面上均衡分配它们，使表单和报表的布局有聚合性。

将元素分组，按照某些原则把元素结合在一起，仔细地选择原则，然后始终坚持，让布局容易被理解：

- 频率：根据用户引用的频率或进入的频率给项目分组。把最经常使用的项目放在“首要”位置，比如在英语环境的布局的左上方。
- 顺序：如果项目形成某种自然的顺序，那么按照文化背景 (例如，在英语中从左到右)的自然顺序布置它们。
- 重要性：把比较重要的项目放在一起。
- 功能：把和特定功能有关的项目放在一起。

均衡元素是一种图形设计策略，为的是让布局使用起来感到舒服。为了整齐，将域和域对齐，其他项和其他项对齐，即使是外部轮廓也一样。如果有许多不同大小的域，就有可能使用户的眼睛移来移去，这样容易分散用户的注意力。可以例外的是，在多行的文本域中使用不规则的文本是合理的。如果有这些情况的话，在大多数计算机文本布局中，右边有参差不齐的空白，其可读性比较好。而使用比例的字型的缺点可能使对齐的文本看起来很愚蠢，而且右边参差不齐的外观给用户留下更多的白色空间。最后使表单或页面的中心处于域或样板文本的重心处，这使得布局看起来比较舒服。

(3) 表单的自动化数据登录

用Oracle Developer几乎可以使表单的数据登录自动循环，但是有一些特殊的技术应该考虑。

首先，尝试重新安排表单中的数据登录域，增进其可靠性。可以使用分组 (请参看前面部分)使得登录是一个比较自然的过程。还可以利用确认来提供输入数据的立即反馈。

第二，如果可能，提供缺省值。让用户按 ENTER键接收一个值比任何其他数据的输入更可靠、更快。如果聪明地选择了缺省值，那么用户可能只需要操作不多的几项，因而节省了输入。

第三，对没有经验的用户使用值列表来提供可接收的输入。 Oracle Developer可以很容易将列表对话框和域联系起来，也容易为输入数值提供下拉列表或组合框。

第四，采用数据项触发器可使域变得更聪明。一种技术是依据上一个域的输入来填写另外的域，例如，采用域分组策略就能很好地工作。另外一种技术是通过智能解释输入，为有经验的用户提供输入数值的捷径。例如，可以用 PL/SQL合理地写类属代码，只用几个字符在表或记录组中查找一个值。这种“迅速—选择”方法在用户界面中越来越多地出现，而且用基本SQL的WHERE子句并不难做，可以在许多有利创造的方法中扩展这种思想，改进数据输入的速度和可靠性。

3.2.4 性能

表单或报表的性能的基本标准是操作速度和流程与操作者响应的配合程度。

表单的数据输入是关键的性能组件。如果在触发器中加入许多 PL/SQL代码，触发器在导航或确认事件过程中启动，那么要保证其性能和人的响应能够匹配。如果用户已经操作到域的外面，那么他们应该以相当迅速的次序看到确认的结果。如果这种触发器有性能问题，那么试着将这些引起问题的代码移到提交处理，或者移到在明确要求下启动而不是自动启动的触发器中。大多数用户对提交阶段的性能期望比较低，而且可以通过反馈信息，对所要求行为适当地设置期望值。把应用程序作为一个整体考虑，通常应该使用户可以预测延迟。一致的轻微延迟比有许多变量的用户不能预测的延迟要好。

在表单中查询实在不是问题，因为可以设置用户的期望值。但是如果可能的话应该避免这些情况：

- 当用户执行Execute_Query功能去查询所有的数据，而不是通过例子输入查询时，不应该长时间离开应用程序。为表单构造缺省查询，如果可能的话，它将接收一小组数据，也可以利用先于查询的触发器，如果无查询项，可加一查询项到这个查询中。

注意 许多平台提供中断查询的能力，比如在Windows或Windows 95中通过使用CTRL-C可以中断查询。可能要设置一个环境变量来做这个工作，有关细节请参看安装文档资料。

- 防止系统给用户一个不合理的或不清楚的响应。确保在大多数情况下，特别查询能力提供合理的响应。要考虑如果用户给查询加一个特殊的域将会发生什么情况。利用先于查询的触发器来预告用户对查询的响应是否会慢。

报表有三种流行形式：交互式的、特定的、批处理的。交互式报表使用按钮来提供下拉特性，制作交互的应用程序报表输出。下拉报表的导航性能和显示性能应该和响应的期望相匹配。特定的报表是当需要它们时产生的报表。这些报表的性能也应该和期望相匹配。通常要求相当迅速的响应。批处理报表可以在你晚上回家以前启动，或者通过系统中的计时器机构来关闭。对批处理报表唯一的性能要求是它们用的时间不要比转发这些报表的时间长。如果有一个日常批处理报表，那么这个报表应有比较好的格式，而且在一天之内打印。要确信你了解了所有的要求。

3.2.5 帮助

用户几乎总是喜欢在应用程序中得到帮助。有几种方法可以帮助用户。在前面部分所述的大多数布局技术，通过使表单或报表比较容易或比较自然来帮助用户工作。但是还有一些

更特殊的事情可以做。在交互式表单中可以做的最重要的事情之一，是立即为用户反馈他 (或她)操作的结果。这就意味着在用户的操作和信息之间，或操作和响应之间的时间应尽可能少。Oracle Developer通常提供基本反馈，提示告诉什么时间它正在工作或正在用数据库做某件事情。可以作一些工作，大大超过这个缺省特性。有两个方面要特别考虑，即确认和性能反馈。

如果用户的某个行为导致一个错误的条件，则应立即把这个错误告诉用户，有时这是比较难做的。例如，如果由于用户没有特权而改变了数据，违反了安全性，但 Oracle Developer并不能够自动地捕捉这个错误，而要等到你提交和发送数据到服务器。可以使用各种选项和技术来将反馈闭路器移动到行为上。第 9章详细讲述了这方面的内容。要确定应用程序始终在确认用户的行为的结果，或使这个结果无效。可以依靠 Oracle Developer做大部分工作，但某些情况下，可能需要做进一步的工作。

应该始终给联机用户提供足够的信息，使他 /她知道所发生的事情。 Oracle Developer对几乎每一个可以想到的对象都提供广泛的帮助能力。比如项目上的工具提示。如果可能，利用它们可以为用户通俗地解释错误，而不只是给出原始错误信息。使用任何通常可以得到的帮助或互交式文档资料，为应用程序提供联机的文档资料。

最后，需要清楚地了解用户如何能消除错误。对待错误的最好技术是在第一地方 (在它们首次出现的地方)防止它们，这是技术和可靠性技术的目的。但是，极少数应用程序做到了极点，决不允许用户出错，如果是这样，这些应用程序可能不能用。应该确信理解了用户怎样能够纠正错误，然后使应用程序对用户清楚地表示出来。

Oracle Developer应用程序使用事务管理来结束数据库的改变。直到提交数据前，都可以改正任何问题。提交以后，则需要到数据库中改变数据。如果在一致性和完整性策略中有任何必然的漏洞的话，则应该为用户提供标识和纠正数据库问题的工具。

本章讲述了建立应用程序的初始步骤，而且给定了应该怎么做和不应该怎么做的基本准则，还介绍了 Talbot农场的例子并用它来说明建立应用程序的第一步。第 4章将开始进入第一个教程。