第2章　Oracle Developer对象综述

Oracle Developer有哪些专门的工具？本章综述各种可以用来生成应用程序的工具和对象。这将为以后各章的学习打下基础。图 2-1展示了Oracle Developer系统的总体结构。

Project Builder是使用Oracle Developer模块工作的主要接口，一般不需要使用它。但是，如果要用两个以上的表单工作时，将模块组织到工程之中将会使你轻松一些。可以为每个工程的Oracle Database Server建立一个缺省的登录，以免出错或因为访问错误的开发数据库而浪费时间。可以用单击按钮来生成或重新生成模块，而不必令人厌烦地去逐个通过每个模块。可以设置工程工具条以访问所有的工具，而且可以书写宏，自动生成各种过程。

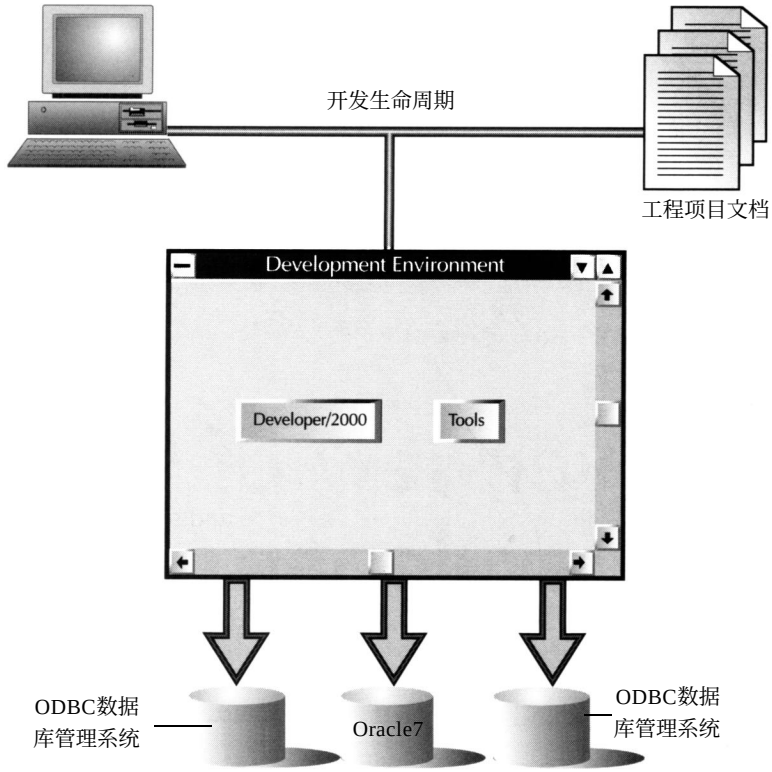

图2-1　Oracle Developer结构

Project Builder能够组织程序和自动生成程序。然而，创建 Oracle Developer应用程序的实际工作是在单个生成器 —— 表单、报表和图形生成器。这些工具可以创建各种应用程序模块：表单集、报表集、图形集。这三个工具中的每一个都有自己相应的编译器。这些编译器在编译模块时不用整个生成器环境的开销。还有一些有所帮助的工具： Procedure Builder、Query Builder、Schema Builder和Translation Builder。Procedure Builder帮助创建存储在Oracle数据库服务器上的程序，或者创建用户的 PL/SQL代码。 Query Builder帮助创建使用图形界面的SQL SELECT语句。Schema Builder帮助创建关系数据库的模式。转换生成器定位表单、报表

和图形，并且转换所使用的各种字串。

这些组件构成了建立应用程序的 Oracle Developer。在准备部署应用程序时，安装 Oracle Developer Server，它由Forms Server、报表服务器和图形服务器组成。这三个服务器和 Oracle Application Server一起工作，使得可以将应用程序部署到因特网 (或本地内部网)上。

注意 Oracle Developer Server来自它自己的CD-ROM，简化了服务器的安装和使用。在比较Report Builder和Report Server时，还应该了解Oracle单独出售Oracle Reports产品。

下面介绍使用这些工具创建的各个独立模块的范围和结构。

2.1 Forms

Oracle Developer的Forms组件是开发环境的一部分，可以在其中开发表单模块。它也为开发菜单和PL/SQL客户库模块提供软件框架。框架是一个可重复利用的对象系统，这些对象一起工作，用于定义那些基本的、由特定的应用领域所需求的抽象概念 —— 此处是菜单和表单。用Form Builder建立表单和菜单。

2.1.1 表单模块

表单模块是数据库应用程序的主要组件。在内部结构中，表单是最丰富的模块，它包含许多不同类型的对象。图 2-2显示了组成表单模块的对象的层次结构。其中的编程对象包括了一组对象。在本章的“ 可重用对象 ”一节，将会讲述这些对象是什么。

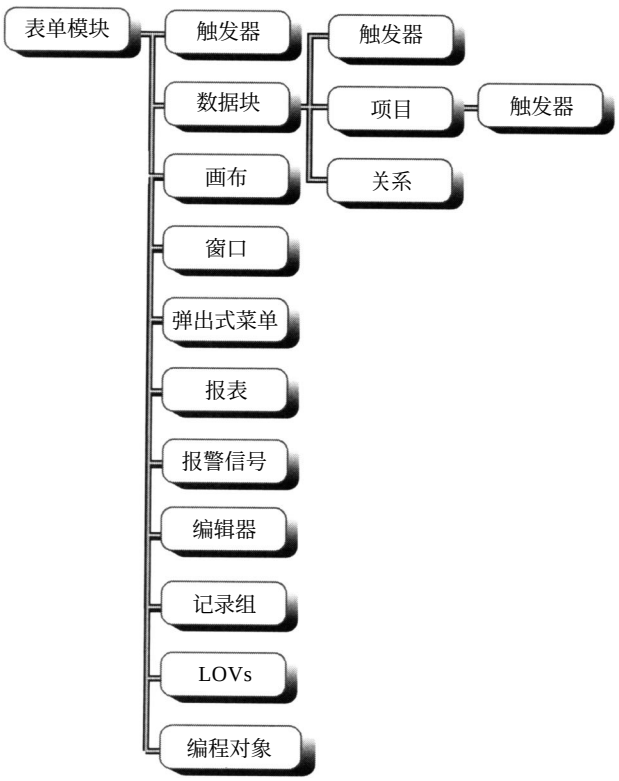

图2-2 表单模块的对象层次结构

1. 触发器

触发器是一个PL/SQL代码块，可以连接到其他对象的表单、数据块或项目。这些代码表示连接到对象上的业务逻辑。还可以将触发器附到属性类上 (在“ 可重用对象 ”一节有更详细的介绍)。当某些事件发生时，触发器“ 点燃 ”，执行触发器代码。

在表单应用程序中的触发器包含了加到应用程序中的工作代码，但不包括那些放到独立的程序单元和程序库中的代码。创建触发器及其业务逻辑吸收了表单和代码设计的大部分结果。第6章有详细的活动图，表明不同的表单的结构和触发器是如何触发的。触发器的示例和与其有关的PL/SQL代码将在以后的大部分章节中出现。

所使用的大部分触发器是内部触发器。它是由 Oracle Developer的框架提供的。触发器有若干种，在第5章中有详细介绍。每个触发器有一个专用的名字，如： When-Button-Pressed (当按钮按下时)、Post-Query(登记查询)、On-Delete(删除时)、Key-Help(关键字帮助)、Pre-Update(更新之前)，可以从这些预先定义的名字中选择。如果在不同层次上 (如数据块层或项目层)定义同样的名字，则Oracle Developer执行最低层的那个。例如为数据块中的 20个项目中的2 定义触发器，而同时可以在数据块级定义同名的触发器。对于 2的触发器项目的事件，则执行项目触发器，否则，执行数据块触发器。可以利用改变 execution style(执行方式)属性来改变执行方式。可以在高一级触发器触发低一级的触发器的前后改变执行方式，以取代缺省的特性。可以组合几个触发器的方式来取代执行方式。

可以添加用户自己定义的触发器并直接使用它。这些特点在从前的表单版本中是有用的，现在由于PL/SQL的适应性，就不那么重要了。 Oracle计划在不久的将来取消它。

一个运行中的表单有两个主要的模式：正常模式和“ 进入查询 ”模式。在正常模式下可以插入、更新和删除数据库的信息。“ 进入查询 ”模式可以创建一个实例记录，用该实例去指定一个对实例数据库数据的查询。 Oracle Developer的这种通过实例查询的方法提供了许多特色，使得在应用中不用代码或者少用代码。

注意 某些触发器不能在“ 进入查询 ”模式中使用(参看“ 数据块 ”部分)，因为它们的活动和数据操作有关，因此在这种模式下无意义。

某种触发器，如按键触发器，是在用户界面的组合按钮按下时启动。添加按键触发器允许对键盘重新编程，使按键的动作与标准键盘的定义不相同。因为这些触发器是非标准的，在标准菜单的帮助栏中，触发器定义的内容可显示出对那些被列出的可用键的描述。

2. 数据块

数据块是表单的中间建造单元。可以用两种方法考虑数据块：作为数据项目的集合或者作为记录的集合。每个记录的结构是相同的。限定每一次显示的记录数目，而且指定用水平或垂直方式显示记录。

有两种数据块。“ 基于表的数据块 ”对应于数据库中的一个关系表、一个对象表或一个视图，管理某些数目的记录，这些记录对应于表或视图中的相应的行。基于表的数据块还可以表示一个存储的过程。另一种数据块是“ 控制块 ”，它不与表、视图和存储过程相对应。通常，控制块表示一些单值项目目的集合，实际上只有一个记录。例如，需要跟踪一组记录的合计值、总计值或平均值，那么可在控制块中创建一项目来表示那个值。虽然对于控制块来说，那个项目只有一个值，但是，却相当于基于表的数据块的一组记录。

数据块的显示有两个特殊的特性与作为记录的容器有关。数据块可以有一个滚动条，让

用户管理那些显示内容大于显示屏幕区的记录组。它还有一组专门的可视化属性，在本章的“可重用对象”中讲述。这些属性为当前记录定义不同的显示，当前记录是指光标当前停留的记录。

通常，在数据块内的“导航”按照记录定义数据项目的次序进行。运行时，系统具有从记录移至记录、从项目移至项目、从数据块移至数据块的功能。当从记录的最后一项目出来时，通常返回到同一记录的第一项目。Form Builder允许重新定义这个缺省的属性。可以允许跳到下一个记录(如果从第一项后退的话，即到前一个记录)，也可以允许从最后一项或第一项移出时，跳到下一个或前一个数据块。可以设置所指定的数据块为下一块或前一块，创建一个数据块连接表。还可以告诉表单在数据块菜单中输入数据块的名字，这样，可以通过在连接表中选择数据块的名字的方式跳到被选择的数据块中。

基于表的数据块的主要功能是为数据库的关系表或对象表或存储过程提供一个接口。Oracle Developer提供一个Data Block Wizard，帮助从数据库模式(schema)中建立这些数据块。向导还帮助从数据库模式建立主-从(master-detail)数据块。这些数据块按照父子关系建立关系，比如发货单及其所属项目。

Oracle Developer通过构造基于数据块及其结构的SQL语句来自动地管理数据。因此，数据块具有若干管理SQL语句结构的属性：

- Enforce Primary Key and Key Mode(强制执行主关键字和关键字模式)：Enforce Primary Key特性缺省地，Forms使用Oracle唯一的内部标识符——行标识符rowid——来标识在SQL语句中产生的行。Enforce Primary Key特性告诉Forms，改用项目及其值来标识行，该行与数据库表中组成主关键字的列所相对应。Key Mode属性还告诉Forms UPDATA语句中是否包含主关键字值。有些数据库管理器不允许更新主关键字值。
- <operation>Allowed<操作>允许：可以对数据块禁止使用指定的SQL语句——SELECT、INSERT、UPDATE、DELETE。例如，为了防止用户删除行，可以设置Delete Allowed为No。
- WHERE子句：这是SQL的一部分。Forms将其加到为查询而产生的SELECT语句上。有关细节请参看第5章的有关内容。
- ORDER BY子句：这也是SQL的一部分。Forms将其作为查询而产生的SELECT语句的ORDER BY子句。
- Optimizer Hint(优化提示)：Forms将这个Oracle SQL组件加到SELECT语句中，使用Oracle的专门的优化特性加速查询。
- Update Changed Columns Only (仅更新变化了的列)：如果这个属性设置为Yes，那么Forms只将变化过的列包含在它产生的UPDATE语句中。否则，不管这些值是否变化，将更新所有的列。通常应该将其设置为No。
- Locking Mode(加锁模式)：在改变某一行的值时，这个属性确定是在提交数据库的变化时锁定该行，还是立即锁定这一行。
- Data Source and DML Target(数据源和DML目标)：这个属性用来确定哪种数据源给数据块提供数据，或者向哪里写数据用FROM子句、表名字、存储过程或者事务处理触发器。
- DML Array Size (DML数组的大小)：这个属性确定一次插入、更新或删除的记录数。

数据块还为表单的query-by-example(用示例查询)属性提供接口。用户可以将数据块放在Enter Query模式中，然后在示例记录的域中输入查询判据。当这个查询被执行时，表单根据其值自动建立SQL SELECT语句。结合使用WHERE子句和ORDER BY子句的特性，可得到综合的查询条件。在Enter Query模式中，还可以为WHERE子句写入自己附加的SQL。可以指定存储器缓冲区记录数、最大查询时间和一次取入表单的最大记录数。

3. 项(项目)

项(项目)是表单最主要的构造单元 (在本书中，为了表达清楚起见，有时使用“项目”这个词。“项”和“项目”是指同一种东西 —— 译者注)。项有许多属性 —— 多得难以在此概括。本节只提供项的功能摘要，而不涉及属性的细节。使用专用的在圆点分隔的语法来使用项：<data block>.<name>。在这里，<data block>是拥有项的数据块的名字，<name>是项的名字。在数据块中，项的名字是唯一的。

表2-1描述了不同类型的项目。说明中提到的任何特殊的属性只与特殊类型的项目有关。

表2-1 Forms中项目类型及其用途

项目的类型	说 明
ActiveX控件	显示ActiveX控件，参看第15章
Bean域	显示与表单结合的java bean的输出，参看15章
图表项目	显示来自图形形式模块的图表
复选框	让用户接通或断开布尔值
显示项	显示数据的“Read only”(只读)文本项，不允许用户改变
分层树	显示类似于Windows Explorer或对Object Navigator的树结构通过树操作数据
图像	显示一个位图或矢量图
列表项目	若干正文行的列表，可以作为弹出式列表、固定尺寸列表或组合框显示
OLE容器	显示OLE对象，参看第5章
按钮	使用户执行一个与按钮有关的触发器 (当按钮被按下时)
单选按钮组(单选)	在若干按钮中接通一个按钮，其他所有的按钮断开。这个对象拥有一组无线按钮，这些按钮一起工作，每个按钮表示一个特殊值，它表示数据库的一列
声音	显示一个表示声音的图标，单击图标则有播放
文本项	在单行或多行的域里显示文本
用户区	显示任何用程序装入这个空间的合适的内容
VBX控件	显示VBX(16 位)控件

每一个特定的类型有一组特性。例如，文本项可能有滚动条，单选按钮可能有 Other Values(其他值)属性(该属性规定对于非单选按钮组中的值，数据库查询将发生什么的事情)。因为项目是数据值，所以项目有若干和数据有关的属性。表 2-2列出了Developer表单正常的不同的数据类型。某些类型的存在是为了和以前的表单版本兼容，则未列入表 2-2中(例如EDATE、JDATE、RMONEY，等等)。如果可能，应该避免使用这些类型，而只使用表 2-2中的类型，以保证所有的项目能够直接正确对应到 Oracle 的数据类型。

表2-2 Forms中项目的数据类型

数 据 类 型	说 明	避免使用/可以使用
Alpha	任何字母符号—大写、小写或大小写混合	避免使用
Char	任何字母或数字符号(与Oracle的VARCHAR2对应)	可以使用
Date	有效的日期和时间(与Oracle的DATE类型对应)	可以使用
DataTime	有效的日期和时间(与Oracle的DATE类型对应)	可以使用

(续)

数据类型	说 明	避免使用/可以使用
Int	任何带符号或不带符号的整数值	避免使用
Long	扩展长度的字符串	可以使用
Number	带符号或不带符号的定点数或浮点数，用或不用指数/科学表示法(与Oracle的NUMBER列对应)	可以使用

有若干用于控制项目值的格式和验证的属性：

- Format mask (格式掩码)：一个广泛的各种值的说明，格式化输出或者验证输入。
- Default value (缺省值)：在创建项目时分配给它的值。
- Copy value (复制值)：在创建项目时，由其他项目复制过来的值。
- Length and Range (长度和范围)：数据的最大长度和范围。
- Must Enter (必须输入)：确定用户是否必须输入值的设置。
- LOV ：一个与项目有关的列表值。使得输入的值是一组指定的值 (参看与“记录组和LOV”部分有关的内容)
- Editor(编辑器)：正文编辑器对象。用来输入正文。

项目具有全部可视属性：字体、颜色、填充、斜面等等。关键的属性是是否显示项目。可以有一个在任何地方都不出现的项目，但是它保持有值。可以利用这些项目作为 PL/SQL中的代码的变量。一个特殊的用法是从数据库取值送入这些项目，然后根据这些值计算出其他的显示值。还可以书写工具提示，当鼠标移至某项目时，系统显示所写的提示。

项目的另外一个关键的显示特性是显示项目的画布的名称 (参看“画布和窗口”部分的内容)。通过改变画布的名称，可以完全控制项目在显示中什么时候出现和在什么位置出现。还有一个tab(标签)特性，可以使项目在标签画布视窗上按照特殊的标签移动位置。 将在后面看到有关的内容。

每个项目有一个提示符，即在画布上紧靠着项目显示的一个字符串。提示符的属性包括位置(在项目的左面、右面、上面或下面)、与项目的距离和提示符的显示特性 (字体、颜色等等)。当项目移动时，提示符也在画布上在项目的周围跟随项目移动。所以不必操心在画布上建立样板文本(boilerplate text)，也不必担心要在画布上重新排列项目时的管理问题。

有可能不允许“导航”到项目内，可能会完全禁止，或者只允许通过单击鼠标“导航”而不允许用户从项目移动到项目。可以把一个项目与另一个项目相连接，以强制导航按照与其他项目定义次序不同次序进行。

一系列属性影响Oracle Developer Forms在建立SQL语句中使用项目的方法。最重要的一个属性是项目是否与数据库的基础表的列对应。其他的属性控制查询的性能、数据操作的性能和锁定性能。

Implementation Class的属性可以超越Java Bean的控制特性。可以为项目的类型指定实施类(Java bean类的名称)。范围从Bean area到复选框、列表项目、按钮、单选按钮组或文本项目。有关Java Bean控制的详细示例请参看第15章。

最后，有个叫做calculation(计算)的组。它包含若干属性，可以通过公式或汇总其他项目的值将项目转换到计算域，然后，可以创建总计域、汇总域或者所有感兴趣的计算域。可以自动维护这些域，不用写任何触发器。有关计算项目的示例参看第 6章。

4. 关系

关系是一个对象。Oracle Developer的表单用它来构造主-从(master-detail)表单。属于主数据块的关系对象表示主记录与从记录 (detail records)之间的关系。关系的主要属性是从记录数据块的名字和表单管理关系所用的连接条件。

可以特指一些特殊的有关删除主记录的特性 (不管是否删除从记录)或当没有主记录时，插入或更新从记录。第6章讨论这些属性和与从记录的自动显示相关的设置。

5. 画布和窗口

画布是放置文本和项目的背景。每个项目准确地引用属性工作表中的一个画布。可以将一个数据块分到各个不同的画布上。就是说，数据块可以在不同的画布上显示它的项目，而不只是在单个的画布上显示。

画布并不是独立的界面对象。要查看画布及其他的项目，必须在窗口中显示画布。窗口是应用程序的一个矩形显示区，被一个框架所包围，通过 GUI 平台维护。Oracle Developer建造这些各自独立的对象，使用户可以构造窗口，从而提供一个画布的视图。这个视图是窗口内的一个矩形，覆盖整个画布或覆盖部分画布。可以通过窗口查看的那一部分，就是视图。窗口有垂直或水平的滚动条，可以在画布的周围滚动查看不同的视图。

画布有五种类型：

- Content (内容)：显示窗口的基本内容。
- Tabbed(标签的)：用一系列的带有标号标签的画布显示窗口的基本内容。
- Stacked(层叠的)：在许多的其他画布上显示那些有条件的或者可以分离的内容。可以规定在显示的窗口中是否立即显示层叠的画布，或者是使画布处于不可见，而在需要时才使它可见。
- Vertical toolbar (垂直工具栏)：将工具图标的内容显示固定在窗口左边的垂直工具栏内。窗口通过名字来指定在工具栏中使用的画布。
- Horizontal toolbar (水平工具栏) ：将工具图标的内容显示固定在窗口顶部的水平工具栏内。窗口通过名字来指定在工具栏中使用的画布。

不仅可以在画布周围滚动窗口，而且可以通过层叠画布在同一个窗口中显示多个画布。每一个层叠画布可以有它自己的滚动条，使得能够在其他画布不动的情况下滚动某一个画布。有关如何使用这个特性的细节，请参看第 6 章“ 层叠画布 ”的内容。

可以定义一个画布中具有多个标签页。这使得能够建立标签对话框，在对话框中具有多个视图，可以在标签页上单击访问这些视图，如图2-3所示。

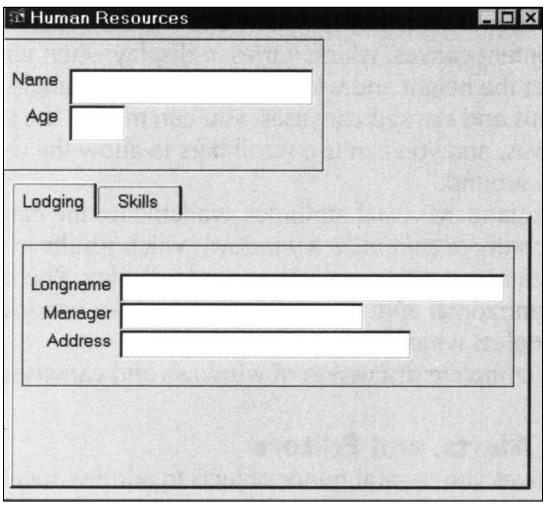

图2-3 多个标签页

标签页画布视图用一种清晰而方便的方法将大量的项目组织到单个的窗口中。通过Layout Wizard建立这些是很容易的。Layout Wizard可以引导你建立和修改所建立的数据块的画布布局。

窗口可以是模式的，也可以是非模式的。如果是模式窗口，则用户应用程序在其他窗

口中做任何事情之前要先解除这个窗口。非模式窗口可以使用户不用解除当前窗口就可以移到另一个窗口。通常，模式窗口有一些特殊的属性，如没有滚动条、窗口大小是固定的、无法使图标最小化。

当“导航”到另外的窗口的项目的时候，可以告诉 Developer将非模式窗口隐藏起来。但是，Developer不在模式期间定义两个窗口的类型，而是用这样的方法定义：

- 文档：是一个通用的非模式窗口，在其中显示一个应用程序的“文档”对象，通常是与应用程序主要部分有关的内容的画布。
- 对话框：一个通用的模式窗口，在其中显示一些选择项目或者其他的控制应用程序操作的方法。

按照性质，对话框通常是模式的。尽管在应用程序中做其他事情的时候，可以使非模式的对话框继续存在。

画布对象指定显示窗口的名字，建立画布与窗口之间的关系。窗口对象通过名字指定其主要内容的画布，即在第一次打开窗口时显示的画布。可以设置画布的高度和宽度。如果想要显示不同的项目和层叠画布，则可能要修改窗口和画布的高度和宽度。可以使用滚动条让用户移动画布和窗口。

可以使画布和窗口具有所有标准的可视属性。可以使窗口图标化和最小化，使窗口当作图标那样显示，可以指定要显示的图标。还可以指定窗口中显示垂直滚动条还是水平滚动条(通常，这些只对非模式的窗口有用)。

在第6章详细讨论窗口和画布。

6. 弹出菜单、警报和编辑器

Oracle Developer提供了几个次要对象以满足特殊需要。第一个是弹出式菜单。弹出式菜单是一个浮动菜单，当在画布或项目上单击鼠标右键时，弹出一个菜单。弹出菜单是在Windows中较新的改进，但是在其他的窗口系统中已经有较长的历史了。这些菜单让你用一种比较自然方便的方法设置对象的行为动作。让用户很快“熄灭”应用程序中某些与鼠标位置有关的活动。

警报是特殊的对话框。它用图标和一个至三个按钮来显示信息，例如“OK”和“Cancel”、“Yes”和“No”，或者其他的按钮。还可以规定三个按钮中的某一个按钮作为缺省按钮。可以用画布将其作为一个窗口来建立。但是使用警报对象可以不经过那些忙乱而很快将对话框放在一起。有三种类型的警报：Stop(停止)、Caution(小心)和Note(注意)。应用程序中显示的简单消息就是这样分类的。

编辑器是一种简单的对话框，其作用是让你将文本行输入到文本项目中。编辑器对象让你指定窗口的大小、可视化属性、编辑标题和窗口的其他属性。让你用不同的方法对不同的文本域创建不同的编辑器。

7. 记录组和LOV

记录组是一种特殊的数据结构，它与表类似，具有行和列。记录组可以是查询记录组，或者是静态记录组。用SQL SELECT语句定义查询记录组。查询的SELECT列表给出这个记录组的列结构。通过在专门的对话框中输入的列说明来定义静态记录组。有几个内部子程序在运行时操作记录组。可以为 LOV使用记录组(参看后面的内容和第 4、6章)。要传递记录到报表中或图形中去作为显示数据参数时或作为 PL/SQL数据结构时，使用记录组(参见第15章)。

提示 记录组可以从Database Server(数据库服务器)中检索数据到用户存储器，然后可以像引用数据库的表一样使用这些数据。但是，由于数据库表在用户的存储器中，就使得访问非常快而且减少了网络的信息流量。可以用这种方法，用记录组来优化应用程序。

LOV(list of value，值列表)是特殊的对话框。它显示一个记录组并且让你选择其中的一行，并返回单个值。可以将 LOV作为一种方法来使用，即从专门的一组值中进行选择，例如，LOV和文本项目结合提供了比较容易输入值的方法，并且提供一个值的列表。对照这个列表可以确定用户的数据输入。自动选择特性可以使终端用户进入字符串方式来自动搜索列表中的项目。还有，可以使用标准的窗口和画布来建立这个对话框。 LOV对象给出一个捷径和方法，在域中自动得到LOV特性而不用任何编程。LOV Wizard简化了整个过程，并且可以使用Query Builder来建立对LOV值的SQL查询。第4章讲述在表单中如何使用LOV。

8. 可重用对象

在“可重用对象”的标题下，这部分概括了几个用来构造应用程序组件的对象。这些对象同样适用于表单和菜单模块，其中的某些 (附加的库和程序单元)也适用于库模块(见图2-4)。

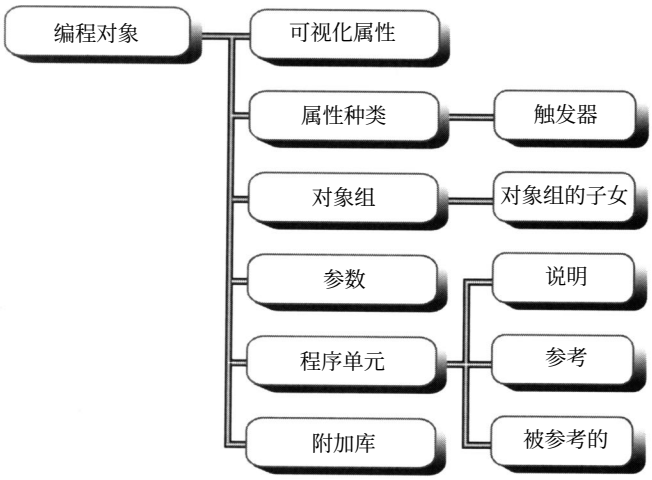

图2-4 构造应用程序部件的对象

被命名的可视属性是若干个显示属性的集合，比如字体或者加到对象 (项目、画布、数据块)上的背景色和前景色。如果从其他的对象中引用这些可视属性，那么引用的那些可视属性将取代这个对象的属性。如果改变可视属性的特性，那么在所有的对象中的属性将继承可视属性的特性。

属性类是一组对象属性(任何一个对象的属性)，目的是附加于一个对象。就像可视属性一样，当一个属性类附加于对象时，可以使得该属性类的所有属性对于所定义的对象均有意义。还有，当改变属性类时，连接了该属性类的对象可以继承或者也可以不继承这些新值，这取决于这些属性值在对象层上是否是被取代。

可以从属性类中创建子类，或者从其他对象划分子类。这种特点可以使得在对象中增加新属性或新方法，或者改变已有的属性和方法。

对象组使你包装可重用对象，为以后的复制或划分子类所用。对象组在单个标题下的模块中收集一组对象。通过复制或者划分对象组的子类，可以得到它包含的所有对象。可以将任意的对象向下组合到数据块层，但不能对块内的项目进行分组，必须在对象组中包含完整

的数据块。

对象库是一个模块。通过拖动对象并放入到库模块中的方式，集中可重用的对象集。要重新使用对象库中的对象，就将库打开并将对象拖动并放入应用程序中。利用库可以将对象组织在一个或多个标记文件夹中。一个对象库可以包含单个的对象，如块、窗口、警报、属性类等等，也包含对象组。

第10章解释命名的可视属性、属性类、对象组和对象库的细节，还说明如何有效地使用它们来建立可重用的组件，以及如何将标准的属性组放入应用程序中。

参数是在模块层上定义的表单、菜单、报表或显示的数据对象。在模块的范围内可以利用参数作为任何PL/SQL程序的变量。参数有数据类型、缺省的初始值和最大长度。可以在启动表单时设置参数。设置方式有：通过传递一个值；通过将参数放到命令行中 (PARAM=“VALUE”)；通过将参数传递到参数清单中(参看第5章)。

程序单元是在模块范围内定义的PL/SQL程序包、过程和函数。所有的Developer模块允许将程序单元定义为模块的一部分。库模块有点特殊：它将程序单元组装为模块，并且通过连接的库对象，使其可以为其他的模块使用。这是对已经定义为模块的库的引用可以调用在模块中的由模块的PL/SQL定义的程序单元。不能调用在没有连接到的其他模块或库中定义的任何程序单元。

每个子程序的程序单元都要显示它的技术要求 (过程或函数的参数或函数返回的类型)。每个程序包的程序单元要指出它的子程序 (程序包子程序)。每个程序单元还要说明这个程序调用什么程序单元(引用对象)和哪个程序单元引用当前这个单元 (被对象引用)。第12章解释程序单元(程序包、过程和函数)的细节。

2.1.2 菜单模块

菜单模块比表单模块要简单得多。如图2-5所示。

图2-5 菜单的组成

菜单模块由一组编程对象(和前述部分所讲述的相同)和一组菜单组成。每个菜单又由一组项目组成。第6章详细地讲解如何定制菜单。

菜单项目可以是一些简单的项目，或者可以是几个特殊的菜单项目形式：

- Check(复选)：旁边带有一个复选标记的菜单项目。这些项目使得通过菜单而不是通过对话框允许使用或者禁止使用所选项目。
- Radio(单选)：是一组互不相容的菜单选项。如果选择了一个，就不能选择其他的项目。
- Separator(分隔符)：这个菜单项目不做任何事情，只是分隔其他的项目。通常是菜单项目上的空间或空行。
- Magic：平台专用的菜单项目，如Cut(剪切)、Copy(复制)、Paste(粘贴)、Undo(撤消)和Help(帮助)。

每个菜单项目有一个命令。当选择该菜单项目时就执行这个命令。可以是这几件事情之一：

- Null(空)：不做任何事情。分隔符必须是一个空命令。
- Menu(菜单)：显示一个子菜单。

■ PL/SQL：菜单执行一个PL/SQL块(包括Run_Product和Host)来执行其他的Developer程序或者操作系统命令。

也可以将菜单与一个或多个安全角色联系起来。第 10章讲述安全性和菜单的详细内容。

2.2 库模块、内置软件包和数据库对象

Developer中的几个对象出现在所有的组件中：程序库模块、内部程序包和数据库对象。

程序库模块与其他模块比较起来是相当简单的。只有一个程序单元和一组附加的程序库组成。有关程序库和所包含的程序单元的细节，请参看第12章。可以在文件系统或数据库中存储库，就像存储任何模块一样。图2-6是库模块的结构。

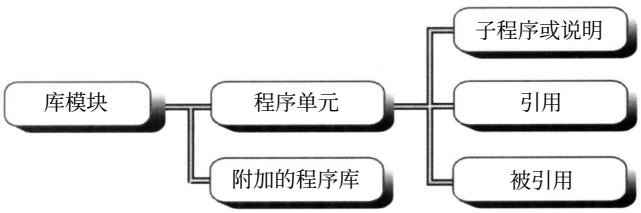

图2-6 库模块

所有的Developer生成器访问一系列内置的PL/SQL程序包，然而程序包不同于其他产品。这些程序包提供了一组范围很宽的工具来操作模块和组件中的其他对象 (Forms、Reports或Graphics)。每个程序包都有技术说明，列出程序包中的子程序以及其说明(参数及函数返回)。每个程序包在产品参考手册中有多方面的文档资料。应该查阅这些文档资料，查看程序包的类型和异常事件以及使用不同的子程序的细节。

还有，所有的生成器都可以访问连接的数据库的对象。图 2-7表明了数据库对象的对象分层结构。生成器列出数据库的所有用户。对于每个用户来说，可以看到可访问的存储的程序单元、库、表和视图。用户只能查看那些通过注册的用户名能够访问的对象。例如，如果是作为没有特权使用特殊表的用户，就不能在数据库对象中查看到在这个数据库对象中列出的表。库是存储在数据库中的库模块，与文件系统完全相反。

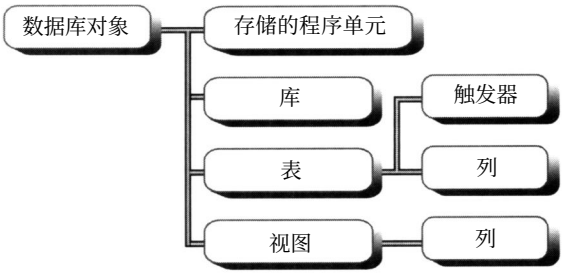

图2-7 数据库对象的分层结构

表也显示了表的数据库触发器。这些触发器是一些 PL/SQL块，在数据库中有特定的事件发生时执行它们，比如INSERT或DELETE。虽然与Developer代码的触发器类似，但是数据库触发器响应数据库事件而不响应应用程序事件。表和视图也显示它们的列以及数据类型。

可以通过从其他的模块拖动程序单元到存储程序单元的数据库对象区域创建存储程序单元。这样可以将子程序或程序包的处理移入数据库，允许开发多层体系结构。通过用这种方法移动代码常常可以改善性能或可靠性。

2.3 Reports

Developer的Reports组件是开发环境的一部分，报表模块是在开发环境中开发的。而且在这个环境中可以引用外部的查询对象，可以设置和存储调试对象。 Report Builder也包含库和

数据库对象。

报表模块有复杂的结构，如图 2-8所示。报表的基本组件是它的数据模型、参数形式、报表触发器和布局。第 7章讲述报表的不同特性。它也可以有程序单元和附加程序库，就像在本章“ 可重用对象 ”部分中讲述的那样。

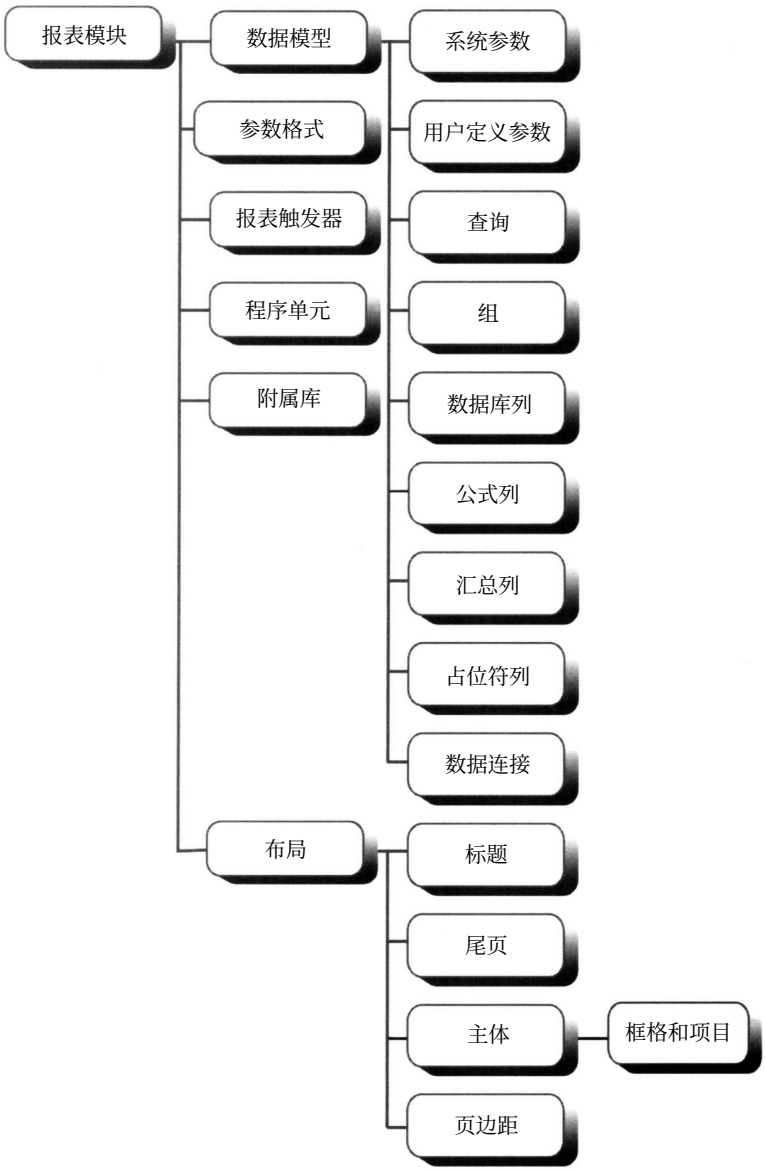

图2-8 报表模块的对象层次结构

报表的“ 数据模型 ”是数据结构及其在报表中不同的表现。可以在专门的图形编辑器中创建数据模型。

参数对象是模块变量，可以在 PL/SQL代码中和任何接受数据值的地方引用它们，就像输入的数据一样。“ 系统参数 ”是Reports自动定义的参数，“ 用户定义参数 ”是用户定义的参数。设置运行时的参数有缺省格式，或者可以在报表中建立完全不同的格式作为参数格式对象。

当运行报表，提示用户设置指定的参数的时候，报表程序运行这个格式。第 10章详细讨论报表参数、缺省参数格式、用户定义参数格式和利用参数使报表可重用的所有方法。

查询是一个SQL语句，它返回数据值，这些数据值是报表的基础。可以将查询嵌入报表，或者使用“外部查询对象”：即在独立文件中的简单 SQL文本，可以由应用程序共享。“组”将查询返回的记录标记为报表中的重复组。组的层次结构表示记录之间的嵌套。“数据库的列”是来自查询的SELECT列表的列。“公式列”是用PL/SQL块计算的特殊列。“汇总列”是报表中指定用来为多个记录存放汇总信息的列，就像小计或总计那样。这些列在高于所概括的记录的层次上编组。“占位符列”定义用来从触发器、公式或用户退出时填入的列，而不是用数据或标准的汇总填入的列。例如，可以导出专门的报告域，报告在一组地域行中那些销售最多的地区的数据。“数据连接”是用来在主-从(master-detail)报表中将一组行连接到另一组行。

报表的“布局”是它的图形结构。“标题页”在前面，接着是“主体页”，最后是“尾页”。报表的“页边距”是指在标题页、尾页和文本页的正文边界以外的区域。报表主体包含全部重复的框格和报表项目，用图式结构将数据模型放到格式化的报表之中。标题和尾页包括任何数据元素或者样板，分别使得前边和后面的报表体有意义。可以通过 Live Preview(live previewer)操作这些对象中的大多数，可以在报表形式的友好的用户界面上隐藏对象零乱的细节。

表2-3概括了几种报表的布局，每一种布局可能对伴随的数据模型有特殊的要求。

表2-3 Developer的Report Layout

Report Layout的类型	说 明
表格(Tabular)	缺省类型。这种结构是在每一页重复用列的标题组织一系列的数据行
主-从(Master-Detail)	将两个或多个组构成一些关联的、嵌套的行，目的是对应于外部的组的每一行，显示一个行的集合
表单(Form)	象表单一样组织数据，用一行格式化后的数据放在若干排上，个别的域的左边有标记
格式信签(Form Letter)	是样板文本和数据域的混合体。它用数据模型中的数据行代替样本文本，目的是产生同一表单的多个拷贝，每行一个
邮件标签(Mailing Label)	在每一页的固定尺寸的框中打印字段的重复组
矩阵(Matrix)	称为交叉表的报表更合适。它用有标题的行列网格显示数据

第4章介绍如何创建数据模型和报表布局。第 7章更详细地继续介绍有关的内容。每种布局有一些选项，可以使用相对来说少的基本布局做出数目惊人的事情。 Report Wizard帮助你通过一系列交互对话建立报表。

“报表触发器”是一些 PL/SQL代码块，这些代码在含义明确的“点”上执行：在报表之前；在报表之后；在页面之间；在参数表单和参数表单之后。在报表中也有其他的触发器，但是不独立，用在表单中的方法命名对象。通过拥有它们的项目来访问它们。第 5章详细讲述报表触发器的触发和如何书写它们的说明。

Report Builder还将调试信息结合到可以利用的对象中，可以创建调试活动 (可以在PL/SQL代码中设置断点)。调试活动在调试会话期中持续。它还包含一个在运行时查看的有关调用堆栈的。有关调试报表和在 PL/SQL调试程序中有关这些对象的细节，请参看第 13章。

2.4 Graphics

Developer的Graphics组件是开发显示模块的环境的一部分。显示模块是从数据库的数据

导出的一个或多个图表，它可能包含图形元素的任意组合，其中包含或不包含对数据库的引用。可以利用显示模块精确地显示业务数据的图形，或者将它们当作画图工具，或者两者兼用。

如图2-9所示，在对象的项目中的图形显示的结构是简单的。显示模块包括布局、模板集、查询、参数、声音、计时器，以及与其他模块所包含的相同的程序单元和附属库对象。

布局包含了在层次结构中显示的图形组件，在层次结构中表示了它们的关系。它还将这些元素与查询的列联系起来。可以作为图形产品的一部分而得到几个特殊类型的图表，如表 2-4所示。

“图表模板”是定制的一组选择项目，可以采用相同的格式在同一个显示中创建多个图表。例如，可以用一个模板建立一系列的饼图。第10章将详细讲述怎样创建可重用图表定义的模板。

查询是一个SQL SELECT语句，与在报表模块中的一样，它定义一个用图表显示的数据集。构造查询是用来返回那些用精确格式显示的所想要的图表的数据。例如，饼图不能概括各个部分的数据，而必须用 GROUP BY和SUM功能来完成这个工作。第4章讲述如何建立这种查询。

图2-9 图形显示模块的对象层次结构

参数，如同报表一样，是模块的变量，它们在 PL/SQL代码模块中到处定义和使用。参数可以在命令行中设置，或者可以通过从其他的程序如 Forms或Reports中传递参数列表来设置。第10章中有详细的内容。

除了可视化组件外，还可以将显示与声音对象结合，以创建完美的多媒体显示。

专用的计时器对象在图形显示中作为一种报警时钟。可以规定一个时间段，当该时间段结束时，计时器便执行某些PL/SQL代码。用计时器可以实现各种特殊的效果。

表2-4 Graphics Display的图表类型

图表类型	说 明
列	用垂直条显示数据组
饼	将数据组显示为圆饼的各个部分
条	用水平条显示数据组
表	用表格形式显示数据值
线	用与X值和Y值有关的线段显示数据值
散列点	用与X值和Y值有关的点显示数据值
混合	用与X值和Y值有关的线段和点显示数据值
高/低	数据元素用三个值的组合显示：高、低和结束。例如可以显示股市的价格
双Y	对每个X轴的值显示两个Y值，例如可以显示随时间变化的两个变量
甘特图	显示水平条来表示开始值和结束值，通常表示进度表。用 X轴表示时间，Y轴表示进度表中的一组任务

Graphics Builder与Report Builder相同，也综合相同的调试信息。可以创建调试活动——可以在PL/SQL代码中设置断点，这些断点在调试会话期间将持续存在。它还包含一个在运行时供查看的堆栈调度显示。第 14章详细讲述如何调试图形显示和如何在 PL/SQL调试程序中使

用这些对象。

2.5 螺母和螺栓

对于应用程序来说，每个对象都有强大的功用。但是， Developer比它所包含的那些对象的作用更大。Developer是一个系统，它将所有的对象组合在一起，成为一个工作的整体。不仅是这些对象在一起工作，还可能用其他的应用程序的对象工作，以及用其他不同于 Oracle的数据库管理系统的数据工作。

Developer的主要黏合剂是PL/SQL。这种编程语言在触发器、程序单元和其他的程序对象中使用。通过访问项目和参数 (或者甚至是全局数据)，PL/SQL导致发生与所有的对象绑定的事件。利用Developer的组件中的内部子程序，可以操作其他组件中的对象：

- 在报表和表单中嵌入图形显示。
- 传递报表或表单的数据到图形。
- 从图形显示或表单中产生报表。
- 从图形显示中将参数值读入表单。

在各个组件中的专门的子程序使得可以与标准 Oracle以外以及Developer环境以外的事件进行交互。可以使用 Host过程通过操作系统命令接口调用其他程序。在 Windows平台上，可以使用OLE2，ActiveX控件和DDE来操作其他应用程序的对象。还可以利用 Open Client Adapter访问Oracle以外的数据库管理系统，对于这些数据库管理程序可以使用 ODBC驱动程序。可以用这些方法在设计环境和运行环境中完全代替 Oracle。或者，还可以在Oracle下开发而在运行环境中却使用另一个数据库管理程序。例如，用户可以将他们的数据保存在 SQL Server、Sybase或Informix等数据库中。

所有这些选择使得Developer成为一个结构好、健壮的和向 Internet应用程序开放的坚实有力的环境。本书的第二部分和第三部分详细介绍如何原型化 (prototype)、设计、编码、测试、调试和部署终端用户的应用程序。第四部分是可以用于 Developer应用程序的所有各种对象的综合参考信息。