第一篇　表格工具 SQL * Forms 3.0

第一章　SQL * Forms 概述

1.1　SQL * Forms 的基本概念

1.1.1　什么是 SQL * Forms

SQL * Forms 是 ORACLE 关系数据库系统提供的一种用于开发和运行基于表格(Form)应用的第四代语言(4GL)工具软件。用 SQL * Forms 开发表格只须简单地使用菜单、定义表、展开表、屏幕画笔等来说明 Form 应用的需求，而后 SQL * Forms 即把所定义的应用与 ORACLE 数据字典结合生成用户应用。该工具的使用对象是应用程序开发人员和最终用户，要求工具的使用者有 ORACLE 数据库的基础知识和 SQL 语言知识。

由 SQL * Forms 开发的基于表格的交互式应用，通常称之为一个表格(Form)应用，简称为一个 Form。所谓基于表格，是指这种应用类同于人们日常办公业务中用表格对信息管理的办法，通过查表，填表，修改表，插入表，删除表等动作实现信息管理。

比如人事部门对职工的管理，在日常管理中我们可以为每一名职员设一张职员基本信息表。增加一名新职员，填写一张新表；如果要查看某一职员的情况，须在一摞职员表中依职工号码或姓名等查出该职员的基本信息表；一名职员的情况有变动时，比如学历变动，须改动该职员的基本信息表；一名职员调离时，则须撤销相应表格。这些操作分别对应于表格的插入、查询、修改和删除等。

Form 应用就是为了在计算机上类似地实现这种基于表格的信息处理过程，使办公人员以非常接近于日常办公模式的方法在计算机系统中实现对信息的处理。

所谓交互式，是指用户使用 Form 应用对数据库信息处理过程是交互的。综上所述，Form 应用可以定义为：

- 一个由 SQL * Forms 工具开发的基于表格的交互式应用。
- 使用在屏幕上呈现出类似于表格式的填充表，用户可录入，查询，修改和删除数据库中的信息。

关系数据库只是在逻辑上以表格的形式组织和存放数据。在数据库设计过程中，为了减少冗余和确保一致性，要对信息规范化。这样一来，数据库中的基表与最终用户心目中的表格差异很大，而用户对信息的所有操作，最终都是要对基表的操作。对基表的直接操作一般使用 SQL 语言(SQL * Plus)，这不仅要求用户了解 SQL 语言，而且对基表的组织应当了解。对最终用户来说，这种要求是过分的，也就是说，我们不能要求一般的办公人员一定要了解数据库基本结构和 SQL 语言。

Form 应用则是一个模拟日常表格处理的友好界面，最终用户可以通过表格形式来对

数据进行处理。Form 应用不仅界面友好，而且可以提高工作效率，减少差错率。SQL * Forms 正是为应用开发人员和最终用户提供的一个开发和运行 Form 应用的工具。

1.1.2 SQL * Forms 的功能

SQL * Forms 具有如下功能：

· 用于定义表 应用；

· 使用 Form 应用可以用类似于日常表格操作的方法实现对数据库的各种基本操作：插入，查询，修改和删除等；

· 进行合法性检查，保证输入的正确性，减少错误；

· 提高开发效率和用户的工作效率。包括为开发人员和用户提供提示信息，提供缺省值，自动完成计算和统计等。

1.1.3 SQL * Forms 的特点

SQL * Forms 具有如下特点

· 具有第四代开发工具的特点。在用 SQL * Forms 开发 Form 应用过程中，是采用交互填表方式，基本上不用编程，且提供有大量缺省操作

· 用 SQL * Forms 可以方便地生成一个 Form 应用的原型，更复杂的应用是通过在型基础上不断修改而得到。

· 为用户提供有编程接口触发器（ Trigger ）和用户出口（ User exit ），利用触发器和用户出口，开发人员可以定义比较复杂的应用。

· 不取代报表工具软件（ 对打印及 I/O 支持不够 ）和 SQL * Plus（ SQL * Plus 是面向 DBA 的，而 SQL * Forms 是面向应用开发人员和最终用户的 ）。

从以上特点可以看出，利用 SQL * Forms 我们可以方便地定义出难度在中等及中等以上的基于表格的应用。但对于更复杂的应用，由于触发器变得更复杂，需要更多的用户出口，使编程工作量大大增加，已难显示出第四代语言（4GL）工具的特点，在这种情况下应考虑选用其它工具。这也正是 4GL 工具专用性的一个特点，即每种工具都有它的适用范围。

1.2 SQL * Forms 的构成

SQL * Forms 作为 ORACLE 公司提供的一个产品，是由一组基表和一组实用程序组成的。基表用以存储用 SQL * Forms 所开发的 Form 应用的有关信息等，由于本书是面向应用程序开发人员和最终用户的，所以不去介绍基表的构成。

SQL * Forms 的实用程序有如下四个：

（1）CONVFORM（SQL * Forms - Convert）：Form 转换实用程序，用以在 Form 应用定义的不同存储格式间进行转换，可完成如下功能：

· 从基表存储的定义生成操作系统的文本文件。

· 把设计界面或文本文件中的应用定义插入到数据库基表中。

· 从低版本 SQL * Forms 的应用定义文本生成 V3.0 的文本文件。

（2）SQLFORMS（SQL * Forms - Design）：Form 应用交互式设计界面，用以定义表 s 对象，保存应用定义，生成运行文件等。

（3）GENERATE（SQL * Forms —Generate）：Form 运行文件生成实用程序，用以从

应用定义文本文件或存储应用定义的基表生成运行文件。

(4) RUNFORM (SQL * Forms —Run Form)：Form 的 Runtime 界面，用以运行已生成运行文件的 Form 应用。

作为一个应用开发人员，经常使用的是设计界面 SQLFORMS 。在设计界面中可以调用其它三个实用程序。

1.3 Form 应用的对象

Form 应用是由 SQL×Forms 对象(有时简称 Form 对象) 构成的。这些对象包含有操作和生成一个 Form 应用的所有信息。

一个 Form 应用的主要对象是 Form（表格），它包含有 Form 自身及其所拥有的其它对象。Form 对象能与其它 Form 应用、Form 应用组、菜单、报表以及其它应用相连接构成复杂应用。

一个 Form 应用由如下对象构成：

· Form(表格)：表格应用，定义表格级属性。

· Block（块）：是一组域的集合，是 Form 的逻辑构成单位，通常与数据库的一个基表相对应。

· Field（域）：在屏幕上显示一个值的显示区域及相关属性，如显示格、合法性检查、缺省值设置、提示信息、值表等。

· Trigger（触发器）：由 Form 中事件(Event) 触发的处理命令或程序段。

· Form—Level Procedure (Form 级过程)：可带参数的过程。在 Form 内可被 Trigger 或其它 Form 级过程调用，一般在不引起混淆的情况下简称为过程。

· Page（页）：显示信息的集合。一页上可以显示多个块；一个块也可以占用多个页

从 Form 应用运行的角度看，Form 应用还包括如下对象：

· Record（记录）：Form 应用所显示的从基表中提取的一行。一般记录与基表的行相对应。

· Message Line（信息行）：屏幕上的倒数第 2 行，用以显示帮助信息、系统信息和错误信息等。

· Status Line（状态行）：屏幕上的最后一行，用以显示系统状态

SQL * Forms 的对象层次结构如图 1.1 所示。

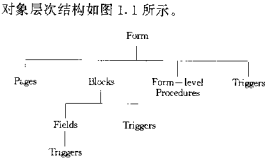

图1.1 SQL * Forms对象的层次结构

图 1.2 是 SQL * Forms 对象示意图

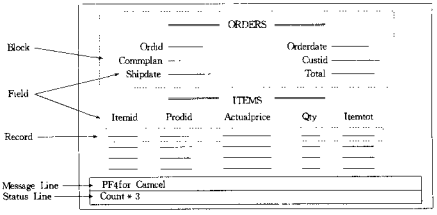

图1.2 SQL * Forms对象示意图

1.4 Form 应用的开发过程

第一步是分析用户需求。这里所说的需求，是假定系统分析员（SA）已完成需求分析，且已完成数据库设计，根据用户要求和对应的数据库用户模式（外模式），详细分析用户需要处理哪些数据、数据的组织格式及关系、处理规则、操作流程等。只有详细了解用户需要，优化用户需要，并得到用户认可，才能开发出满意的应用。这一步需要应用开发人员和用户共同完成。

第二步要把用户的需要转化为 SQL * Forms 的对象。这一步应由开发人员独立完成，也是要求比较高的一步，要求开发人员具有从用户需求中识别出所有 SQL * Forms 对象的能力。比如用户给出的是一张订货单，只要求是对订货单进行插入、查询、修改和删除，看来似乎不复杂，但一张订货单的信息可能因包含有订单基本信息，货品细目，客户基本信息等而对应于数据库的几张不同基表或视图，这样一来，对应成 SQL * Forms 对象就显得很复杂。这样的 Form 应用可能要有多个块，块之间要有主从关系（Master—Detail），协调主从关系要有一组触发器（Trigger）等。通过后面章节的例子可以看出，即使是一个训练有素的开发人员，要想一下子识别出一个用户需求对应的所有 SQL * Forms 对象（比如触发器）不是一件容易的事。好在 SQL * Forms 设计界面有许多缺省功能，开发人员可以利用缺省功能辅助生成许多对象，可以省去识别上的困难，这一点在后面的例题中读者会感觉到。

第三步是应用定义。在识别出一个用户应用（Form）的 SQL * Forms 对象后，接下来是利用 SQL * Forms 设计界面 SQLFORMS 定义该 Form 应用，即对对象的层次构成从顶向下逐层定义各对象。一般可分为四步：

- 利用缺省功能生成 Form 应用的基本框架。
- 修改缺省生成应用的对象，使之满足用户要求。
- 如果需要的话，增加触发器、过程或用户出口，从而进一步完善应用。存储定义到数据库基表并生成运行文件。详细定义过程将在第 2 章中介绍。

第四步是运行已定义好的 Form 应用。利用该应用实现对数据库的操作。

综上所述，Form 应用的开发过程如图 1.3 所示。

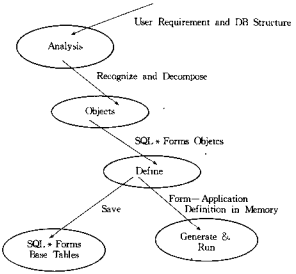

图1.3 Form应用开发过程

1.5 功能键

为了覆盖更多的软件和硬件平台，同其它 ORACLE 工具一样，SQL * Forms 对功能键也采用逻辑键定义与键位映象相结合的方法。对于任何一个功能键，均由两部分构成：

(1) 该功能键的功能。

(2) 该功能键在具体运行平台上的物理键位。前者我们称为逻辑键，后者称为键盘映象 (Map)。比如功能键 [Accept]，逻辑定义如下：

[Accept]：

- 在对话盒状态，关闭对话盒，且接纳你的输入。
- 在定义表(Define Forms) 或展开表(Define Spread Tables)状态，关闭定义表 或展开表，且接纳所有用户的修改。
- 在屏幕画笔状态接纳所有的修改，返回屏幕画笔之前的状态。

ORACLE 的所有资料只介绍逻辑键，这样就回避了具体设备的不同给我们带来的不便，因为功能键的逻辑定义不依于环境。功能键的键位映象在不同的环境中可以不同，且 DBA(数据库管理员) 可以修改键位的定义。比如 [Accept] 键可以定义为物理键 F2，也可定义为其它键。

为了方便用户，ORACLE 工具都配有 [Show Key] 功能键，该功能键用以在联机状态下显示功能键键位映象表 (Keyboard Map)。在不同环境中，[Show Key] 功能键一般对应如下物理键之一：

- CTRL+K
- ESC+K
- F1
- F8

SQL * Forms 的功能键分为两组：一组是用于设计界面的功能键，称之为设计功能键；

另一组是用于运行界面的功能键，称之为运行功能键，我们将分别在后面相应章节中进一步介绍，并在附录 A 中给出全部功能键的逻辑定义。

参考资料

(1) SQL * Forms Designer's Quick Reference Version 3.0, Part No. 3708—V3.0

(2) SQL * Forms Designer's Reference Version 3.0, Part No. 3304—V3.0

(3) SQL * Forms Designer's Tutorial Version 3.0, Part No. 3302—V3.0

(4) SQL * Forms Operator's Guide Version 3.0, Part No. 3301—V3.0

(5) SQL * Fprms Operator's Quick Reference Version 3.0, Part No. 3704—V3.0

(6) SQL Language Reference Version 6.0, Part No. 778—V6.0

(7) PL/SQL User's Guide and Reference Version 1.0, Part No. 800—V1.0

第二章　SQL * Forms 设计界面

2.1　交互式设计界面 SQLFORMS 的启动，构成与退出

交互式设计界面，即 SQLFORMS，是一个用于定义表格式应用的交互式开发工具。该界面不仅提供定义 SQL * Forms 各对象的简捷手段，而且还集成其它 SQL * Forms 实用程序来完成 Form 的生成，运行，转换等工作。也就是说，利用该界面，开发人员几乎可以完成所有与 Form 应用 开发和运行有关的工作。本教程将以 SQLFORMS 为主介绍 Form 应用的设计与开发。

交互式设计界面的启动命令如下：

SQLFORMS [选项参数] [用户名/密码]

其中可选项参数将在第 9 章详细 介绍，[用户名/密码]包括用户标识名，密码 (Password) 及数据库路径名。例如下面是一个启动指定网络地址 Server 上 ORAD7 数据库的 SQL-FORMS 设计界面的例子：

SQLFORMS scott/tiger @t:166.111.8.7:orad7

其中 Scott 为用户名，tiger 为 Password，166.118.7 为地址，orad7 为数据库名。

如果省略 [用户名/密码] 或该选项填写不正确，则 SQLFORM 给出一个注册屏幕如图 1.4 所示。

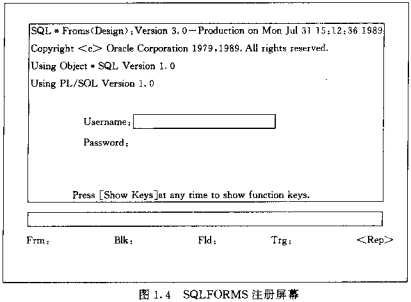

图 1.4　SQLFORMS 注册屏幕

这时在 'Username:' 后应填写正确的用户名和相应的数据库名，而后按 [Next Field] 键（可用 [Show Keys] 键查看键位映象 Map）。在 'Password:' 后填写密码而后按 [Next Field]。如填写正确则注册成功，屏幕上出现如图 1.5 所示主菜单；如果填写不成功，则在信息行显示一条错误信息，用户须重新注册。如果三次注册全失败，则 SQLFORMS 自动退出，返回操作系统。

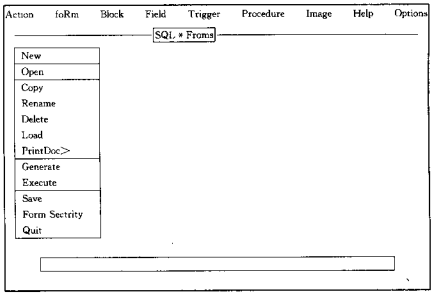

图 1.5 SQLFORMS 主菜单及文件级操作子菜单

SQLFORMS 主菜单有 9 项功能，分别为：

· Action：文件级操作。用以定义新应用，打开或删除已存在的应用，保存应用，生成运行文件、运行应用，生成文档等。

· foRm：用以维护 Form(表) 对象及与之相关的 Trigger。

· Block：用以维护 Block(块) 对象及与之相关的 Trigger。

· Field：用以维护 Field(域) 对象及与之相关的 Trigger。

· Triiger：用以定义和维护 Trigger。

· Procedure：用以维护所有 Form 级过程。

· Image：用以维护 Page(页)对象。

· Help：联机帮助系统。

· Options：用以查看和设置环境参数。

在主菜单上，用左右箭头键选择当前选项(成为反显)用[Select] 功能键选定或按选项的大写字母键选定。例如，移动项到 Action 时按 [Select]键，则弹出下拉菜单（见图 1.5)。在下拉菜单中用[Down] 和[Up]键寻找选项后，按 [Select]键。整个 Form 应用定义过程都是在主菜单下逐项定义的。

SQL * Forms 的大多数对象定义有 2 种形式：定义表 (Define Form) 和定义展开表 (Define Spread Table)。如定义 Field 对象的定义表和定义展开表分别如图 1.6 和图 1.7 所示。

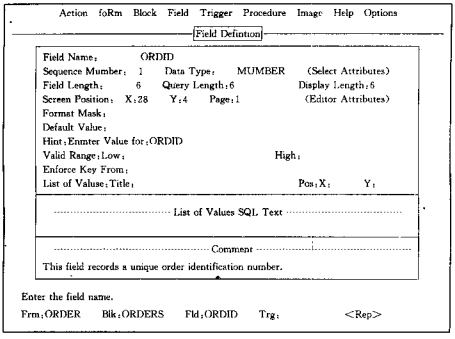

图 1.6 Field Define Form（域定义表）

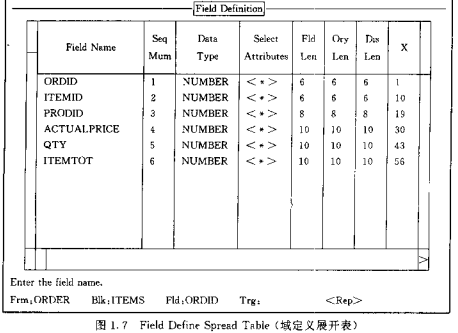

图 1.7 Field Define Spread Table（域定义展开表）

一般进入对象定义时呈现的是定义表，定义表和定义展开表之间的切换用［Charge Display Type］功能键。定义表的好处是一个对象的所有信息一目了然，全呈现在屏幕上。不足之处是一次只能显示一个对象，对象之间的关系不明显。相反，定义展开表可以把所有相应对象一起显示，对象的多少和前后关系十分明显，插入或删除操作时也很直观，但不足之处往往每一对象的信息都不完整，可能分别存于几屏上。在定义应用时，开发人员

根据不同需要，经常在两种定义格式间相互切换。

对象 Form 对于一个应用只有一个，故而其定义只有定义表格式，而对象 Block，Fields，Page，Trigger 和 Form—Level Procedure 的定义都有定义表和定义展开表两种形式。

SQLFORMS 定义交互界面的退出比较简单，只须在 Action 的下拉菜单中选取 Quit 功能即可退出 SQLFORMS，返回操作系统。

2.2 Form 应用的定义过程

假定已从用户需求中识别出 Form 应用的对象且数据库已设计实现，则 Form 对象的定义过程为如下：

(1) 注册到 SQLFORMS 交互定义界面。要求用户须有合法的用户名，密码和权限。

(2) 定义 Form 应用的 Form 对象：

- 在主菜单上用 [Select] 选取 Action
- 在 Action 下拉菜单中用 [Select] 选择 New，弹出新应用对话框，如图 1.8

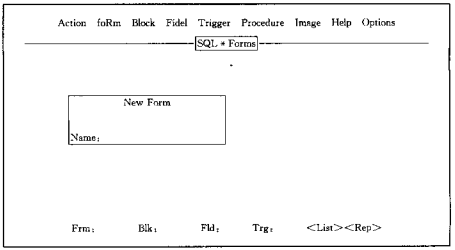

图 1.8 新应用对话盒

- 在 Name 后输入新应用的名字，而后按 [Accept] 功能键返回主菜单即完成对 Form 对象的缺省定义。
- 如果要对 Form 对象进行维护，则在主菜单选取 Form，进入 Form 对象定义表，在该表上维护 Form。之后按 [Accept] 功能键保存维护结果，返回主菜单。

(3) 定义 Form 应用的其它对象：

- 在主菜单选取 Block，利用缺省功能定义块。[Accept] 返回主菜单，则已定义完成缺省应用。
- 在主菜单分别选取 Block，Field，Trigger，Procedure，Image 等，对缺省应用逐步修改完善以满足用户需求。这些修改包括修改已有对象属性，插入或删除对象等。

(4) 保存 Form 应用定义。在主菜单上用 [Select] 选 Action 后，在下拉菜单中用 [Select] 选 Save。在接下来的对话盒中都缺省按 [Accept] 键，直至保存完毕应用定义，返回主菜单。

(5) 生成应用运行文件。在 Action 下拉菜单中选 Generate 选项，弹出生成运行文件对话盒，填入相应应用名称（一般当前应为取缺省名称），按[Accept]直至返回主菜单，即完成生成过程。

(6) 运行应用。在 Action 下拉菜单中选 Run 选项，弹出运行对话盒，填入运行应用名称（当前应用名为缺省名称）。按[Accept]，直至进入运行状态（有关运行状态的操作和退出将在第 8 章中介绍），运行退出后返回主菜单。

以上是有关 Form 应用开发过程的简单介绍。从中可以看出，Form 应用的定义过程一般先定义缺省应用，而后在缺省应用上逐步修改，使之满足用户要求，这正是第四代(4GL)工具的特点。应该指出的是，如果应用比较复杂时，对 Trigger 和 Procedure 对象的维护将是相当复杂的，从这种意义上，有时我们说 SQL * Forms 不是完整的第四代工具，而只是具备了 4GL 的大部分特性。

2.3 Form 应用的存储

Form 应用的存储有 3 种形式：

- 存于 SQL * Forms 基表中的定义记录。
- 以操作系统文件形式存放的应用定义（后缀.INP）。
- 操作系统下的二进制形式运行文件（后缀.FRM）。

用 SQLFORMS 定义的 Form 应用存于内存中，Generate(生成）过程分为两步：，第一步从内存定义生成文本文件 (Generate －1)，第二步生成运行文件 (Generate－2)。而 Action 下的 Load 功能可以把一个以文本形式存放的应用定义加载到内存中。Save 功能把内存中的应用定义保存到 SQL * Forms 的基表中，Open 功能是从基表中把一个应用定义调入内存。存储格式之间的关系如图 1.9 所示

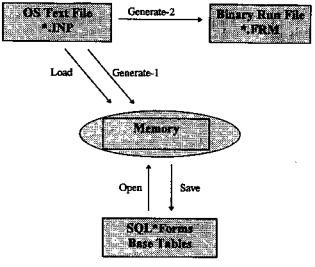

图 1.9 Form 应用的存储

2.4 设计界面的功能键

设计界面的功能键共分 4 组：

(1) 光标移动功能键：

[Beginning of Line]	[Previous Block]
[Down]	[Previous Field]
[End of Line]	[Previous Page]
[First Line]	[Previous Record]
[Last Line]	[Right]
[Left]	[Scroll Down]
[Next Block]	[Scroll Left]
[Next Field]	[Scroll Right]
[Next Page]	[Scroll Up]
[Next Record]	[Up]

(2) 编辑功能键

[Beginning of Line]	[First Line]
[Clear Field]	[Last Line]
[Copy]	[Insert Line]
[Cut]	[Insert Record]
[Delete Backward]	[Insert/Replace]
[Delete Line]	[Left]
[Delete Record]	[Paste]
[Delete Character]	[Right]
[Down]	[Search]
[End of Line]	

(3) 通用设计功能键：

[Accept]	[Next Record]
[Cancel]	[Previous Record]
[Change Display Type]	[Print]
[Copy Object]	[Refresh]
[Help]	[Screen Painter]
[Insert Record]	[Select]
[List]	[Show Keys]
[Menu]	[Zoom In]
[Navigate]	[Zoom Out]

(4) 屏幕编辑功能键

[Copy]	[Paste]
[Cut]	[Previous Block]
[Define Field]	[Previous Field]

[Down]	[Previous Page]
[Draw Box/Line]	[Previous Record]
[Left]	[Resize Field]
[Next Block]	[Right]
[Next Field]	[Select]
[Next Page]	[Undo]
[Next Record]	[Up]

SQL * Forms 共提供 53 个不同的设计功能键。这些功能键的定义请参看附录 A。为了便于学习，此处仅介绍几个重要功能键：

[Accept]： 接纳输入或修改，关闭当前操作窗口（如对话盒，展开表，画笔等）。

[Cancel]： 忽略当前输入或修改，关闭当前操作窗口（对话盒，展开表，画笔等）。

[Next Field]： 移动光标到下一个域，选择框（Check Box），或滚动区（Scoll Region）。

[Previous Field]： 与 [Next Field] 相反。

[Next Record]： 显示下一个对象记录的信息。

[Previous Record]： 与 [Next Record] 相反。

[Select]： 选取光标所在位置的值或菜单选项。

[Change Display Type]： 在 SQL * Forms 对象定义表和定义展开表间切换。

[Zoom In]： 在 SQL * Forms 对象的层次树上自上层移入下层。如当前在 Form 层，按该功能键后即移至 Block 定义层。

[Zoom Out]： 与[Zoom In] 相反。

[Show Keys]： 联机查看功能键键位映象 Map。

在以后我们不再介绍所有功能键的逻辑定义，并用[* * *]简单代替，比如用 [Accept] 表示"按功能键 [Accept]"。

2.5 生成缺省应用

2.5.1 背景条件

现在具体讨论如何用 SQLFORMS 界面来定义一个应用。假定我们要为一个体育器材品厂家开发一个订单管理系统，一张订单一般包括订单基本信息，货品细目信息，客户信息，产品信息和价格信息等，这些信息在数据库中分别对应于 ORD，ITEM，CUSTOMER，PRODUCT 和 PRICE 表。这五张表的结构见图 1.10 至图 1.14。

Name	*Null?*	*Type*
ORDID	NOTNULL	NUMBER(4)
ORDERDATE		DATE
COMMPLAN		CHAR(1)
CUSTID	NOTNULL	NUMBER(6)
SHIPDATE		DATE
TOTAL		NUMBER(8,2)

图 1.10 ORD 表

Name	Null?	Type
ORDID	NOTNULL	NUMBER(4)
ITEMID	NOTNULL	NUMBER(4)
PRODID		NUMBER(6)
ACTUALPRICE		NUMBER(8,2)
QTY		NUMBER(8)
ITEMTOT		NUMBER(8,2)

图 1.11 ITEM 表

Name	Null?	Type
CUSTID	NOTNULL	NUMBER(6)
NAME		CHAR(45)
ADDRESS		CHAR(40)
CITY		CHAR(30)
STATE		CHAR(2)
ZIP		CHAR(9)
AREA		NUMBER(3)
PHONE		CHAR(9)
REPID	NOTNULL	NUMBER(4)
CREDITLIMIT		NUMBER(9)

图 1.12 CUSTOMER 表

Name	Null?	Type
PRODID	NOTNULL	NUMBER(6)
DESCRIP		CHAR(30)

图 1.13 PRODUCT 表

Name	Null?	Type
PRODID	NOTNULL	NUMBER(6)
STDPRICE		NUMBER(8,2)
MINPRICE		NUMBER(8,2)
STARTDATE		DATE
ENDDATE		DATE

图 1.14 PRICE 表

2.5.2 应用设计及主从关系

如果把一个订单设计为由 ORD，ITEM 和 CUSTOMER 三个基表对应的三个块构成：则这三个块之间有如图 1.15 所示的主从关系。

自然，我们希望在 Orders 块显示某个订单（ ORD 表中的一个记录 ）时，该订单客户

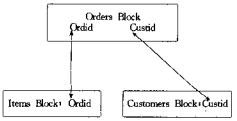

图1.15 块之间的主从关系

的相应信息（CUSTOMER 表中一个记录）应显示在 Customers 块中，该订单所定购的所有产品信息（ITEM 表中一组记录）应显示在 Items 块中。显然，在三个块显示信息之间需要协调主从关系。比如我们查询到几张订单，在浏览订单时，在 Orders 块中每换一个记录，Items 和 Customers 块中信息应相应改变。对于删除，插入等操作也有这些问题。块之间的这种逻辑关系称之为主从关系（Master—Detail）。

通过以上分析，问题似乎变得复杂了。然而可喜的是，SQLFORMS 为开发人员提供的缺省功能中，含有自动生成协调主从关系 Trigger 的能力，因而也就大大减轻了开发工作量和由于考虑主从关系给设计人员带来的烦恼。

2.5.3 生成缺省块和自动主从关系

依上面 2.5.2 小节的设计，我们逐个缺省定义 Orders，Items 和 Customers 块。在定义块之前，先定义应用的 Form 对象：

- 在主菜单中用［Select］选 Action。
- 在 Action 下拉菜单中选 New。
- 在弹出对话框中输入应用名称 MyApp（见图 1.16）。
- ［Accept］返回主菜单。

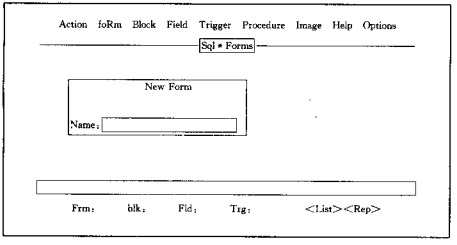

图 1.16 定义新应用

现在定义应用 MyApp 的主块 Orders：

- 在主菜单用［Select］选 Block，如图 1.17 所示
- 在 Block 下拉菜单中选 Default，弹出缺省定义表（如图 1.18 所示）
- Block Name 后填块名 Orders，在 Base Table 后填写对应的基表名 ORD，其它各项

取缺省值（填写时可用［List］）

·［Accept］完成 Orders 的定义，返回主菜单。

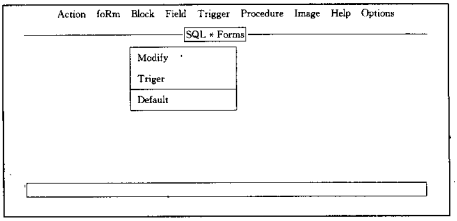

图 1.17 进入块定义

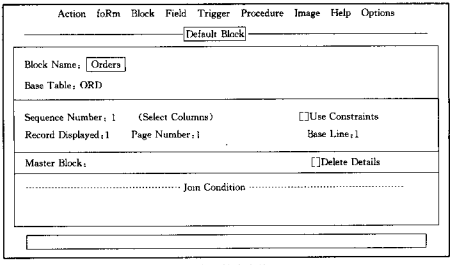

图 1.18 定义缺省主块 Orders

注意，在定义过程中，如果在状态行显示有 <List> 字样，说明当前填充域有值表（List of Value）可用，这时可按［List］功能键，即弹出值表窗口，而后用［Up］和［Down］寻找各备选值，用［Select］选取适当值，即完成该域的输入。如在 Base Table 域时，状态行显示有 <List>，此时按［List］，则弹出如图 1.19 所示值表（有关值表更复杂的用法见有关参考书）。

下面定义第一从块 Items：

·在主菜单选 Block，在下拉菜单选 Default

·填写块名 Items 及基表 ITEM

·忽略中间参数。在谁是该从块的主块处（Maser Block：）填写 Orders

·［Accept］，则 SQLFORMS 自动从数据字典中生成主从连接条件（Join Condition），如图 1.2，返回主菜单

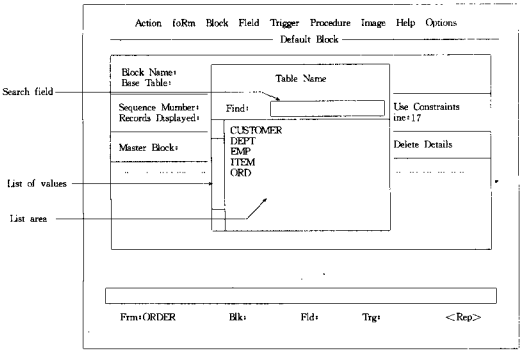

图1.19 值表

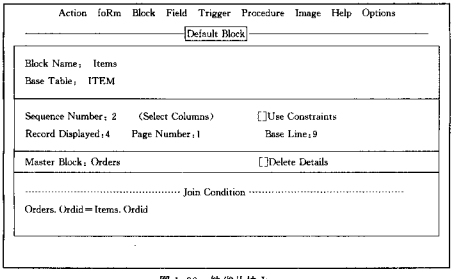

图 1.20 缺省从块 Items:

同样，定义第二个从块 Customers 如图 1.21 所示，只是连接条件须键入。页号（Page Number）改为 2，起始行号（Base Line）改为 1。