为了解决这类问题，在设计关系型数据库时，还要进行“规范化”。所谓规范化，就是将一张大表分割成多张小表，然后再在小表之间建立关系，以此来达到整理数据库结构的目的。通过规范化，可以形成结构更加优良的数据库。DBMS 提供了可视化的工具，用户仅仅通过简单的操作就可以进行规范化（如图 8.7 所示）。

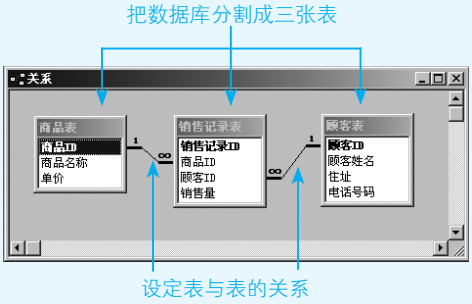

图 8.7　经过规范化处理的酒铺数据库

规范化的要点是在一个数据库中要避免重复存储相同的数据。因此在本例中，把酒铺的数据库分为“商品表”“顾客表”和“销售记录表”三张表，然后再在它们之间建立关系（用被细线连接起来的短杠表示）。通过这样的处理，既省去了多次重复录入相同的顾客姓名、住址、电话号码的麻烦，又能防止把相同的商品名称输入成不同名称的错误。如图 8.8 所示，酒铺的数据被分别存储到了三张表中。表格的最后一行只是表示在这里可以输入下一条记录，并非实际存在着这样一条记录。

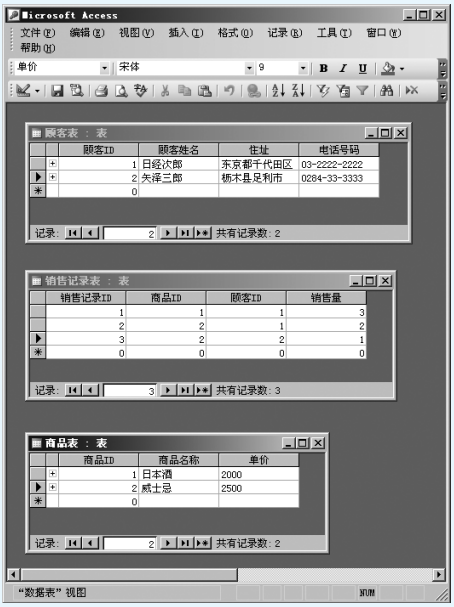

图 8.8 将数据分别存储到三张表中

8.5 用主键和外键在表间建立关系

为了在表间建立关系，就必须加入能够反映表与表之间关系的字段，为此所添加的新字段就被称为键（Key）。首先要在各个表中添加一个名为主键（Primary Key）的字段，该字段的值能够唯一地标识表中的一条记录（如图 8.9 所示）。在顾客表中添加的“顾客 ID”字段，在销售记录表中添加的“销售记录 ID”字段以及在商品表中添加的“商品 ID”字段，都是主键。

图 8.9　把字段设置为主键

正如“顾客 ID”一样，通常将主键命名为“某某 ID”。这是因为主键存储的是能够唯一标识一条记录的 ID（Identification，识别码）。如图 8.8 所示，如果顾客表中顾客 ID 是 1 的话，就能确定是日经次郎这条记录；顾客 ID 是 2 的话，就能确定是矢泽三郎这条记录。正因为这种特性，在主键上绝不能存储相同的值。如果试图录入在主键上含有相同值的记录，DBMS 就会产生一个错误通知，这就是 DBMS 所具备的一种一致并且安全地存储数据的机制。

在销售记录表上，还要添加顾客 ID 和商品 ID 字段，这两个字段分别是另外两张表的主键，对于销售记录表来说，它们就都是“外键”（Foreign Key）。通过主键和外键上相同的值，多个表之间就产生了关联，就可以顺藤摸瓜取出数据。例如，销售记录表中最上面的一条记录是（1，1，1，3），分别表示该销售记录 ID 为 1，顾客 ID 为 1 的顾客买了 3 个商品 ID 为 1 的商品。通过顾客表，可以知道顾客 ID 为 1 的顾客信息是（1，日经次郎，东京都千代田区，03-2222-2222）。通过商品表，可以知道商品 ID 为 1 的商品信息是（1，日本酒，2000）。虽然作为销售记录表主键的“销售记录 ID”字段并不是其他表的外键，但是考虑到以后有可能会与其他表发生关联，并且习惯上必定要在表中设置一个主键，所以予以保留。主键既可以只由一个字段充当，也

可以将多个字段组合在一起形成复合主键。

表之间的关系使记录和记录关联了起来。记录之间虽然在逻辑上有一对一、多对多以及一对多（等同于多对一）三种关系，但是在关系型数据库中无法直接表示多对多关系。这是因为在多个字段中以顺藤摸瓜的方式查找数据并不是那么容易的。如果将酒铺的数据库只分为顾客表和商品表，那么这两张表就形成了多对多关系。也就是说一位顾客可以购买多个商品，反过来一种商品可以被多位顾客所购买。

当出现多对多关系时，可以在这两张表之间再加入一张表，把多对多关系分解成两个一对多关系（如图 8.10 所示）。加入的这张表被称作连接表（Link Table）。在酒铺数据库中，销售记录表就是连接表。如图 8.7 所示，在表示一对多关系的连线的两端，写有“1”的一侧表示“一”，写有“∞”、即无穷大符号的一侧表示“多”。

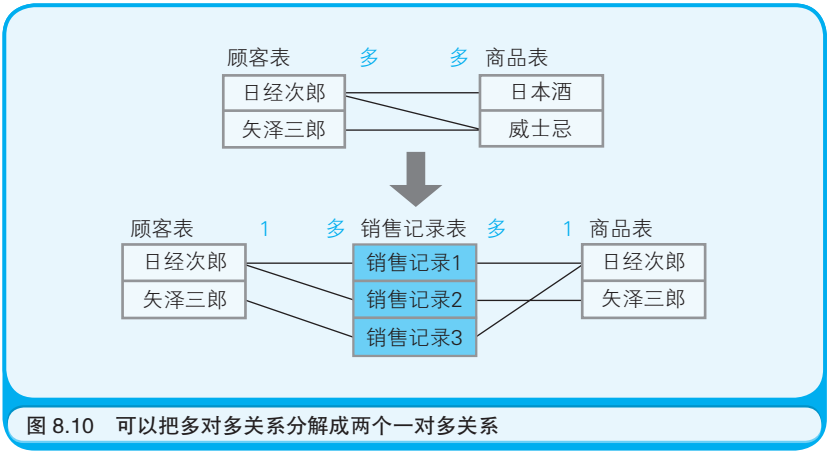

图 8.10　可以把多对多关系分解成两个一对多关系

DBMS 中还具有检查参照完整性的功能，这种机制也是为了一致并且安全地存储数据。例如，在目前的酒铺数据库中，如果从商品表

中删除了“日本酒”这条记录，那么在销售记录表中，曾经记录着买的是日本酒的那两条记录就不再能说明买的是什么商品了。但是一旦勾选了实施参照完整性的选项（如图 8.11 所示），在应用程序中再执行这类操作时，DBMS 就会拒绝执行。

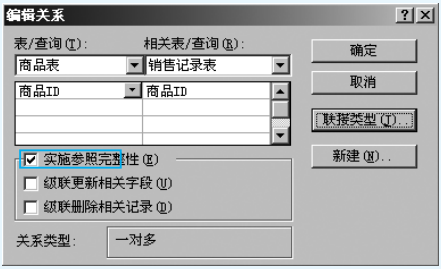

图 8.11　设置参照完整性

如果诸位是直接从编写的应用程序中读写数据文件，那么就会导致用户可以录入在主键上含有相同值的记录，或者由于没有进行参照完整性等方面的检查，使用户可以任意地执行删除数据之类的操作。而 DBMS 却能在这种问题上起到防患于未然的作用，确实是一种很方便的工具。

8.6　索引能够提升数据的检索速度

可以在表的各个字段上设置索引（Index），这也是 DBMS 所具备的功能之一。虽然索引和键这两个概念容易让人混淆，但其实两者是完全不同的。索引仅仅是提升数据检索和排序速度的内部机制。一旦在字段上设置了索引，DBMS 就会自动为这个字段创建索引表（如图 8.12 所示）。

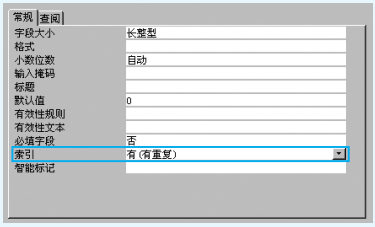

图 8.12 设置索引

索引表是一种数据结构，存储着字段的值以及字段所对应记录的位置。例如，如果在顾客表的顾客姓名字段上设置了索引，DBMS 就会创建一张索引表（如图 8.13 所示），表中有两个字段，分别存储着顾客姓名和位置（所对应的记录在数据文件中的位置）。与原来的顾客表相比，索引表中的字段数更少，所以可以更快地进行数据的检索和排序。当查询数据时，DBMS 先在索引表中进行数据的检索和排序，然后再根据位置信息从原来的数据表中把完整的记录取出来。索引所起的就是“目录”的作用。与图书的目录一样，数据库的索引也是一种能够高效地查找目标数据的机制。

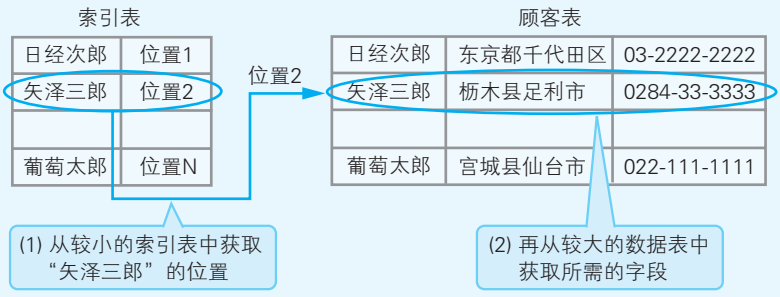

图 8.13 如果字段上有索引就可以缩短检索时间

也许会有人这样想，既然索引能够提升检索和排序的速度，那么在所有表的所有字段上都加上索引不就好了吗？实际上并不能这样做。因为一旦设置了索引，每次向表中插入数据时，DBMS 都必须更新索引表。提升数据检索和排序速度的代价，就是插入或更新数据速度的降低。因此，只有对那些要频繁地进行检索和排序的字段，才需要设置索引。在酒铺数据库这个例子中，只需要在顾客表的顾客姓名字段和商品表的商品名称字段上设置索引就足够了。如果表中充其量也就只有几千条记录，那么即使完全不使用索引，也不会感到检索或排序速度有多慢。

8.7　设计用户界面

只要通过拆表实现了规范化、设置了主键和外键、确保没有多对多关系、根据需要设置了参照完整性和索引，那么数据库的设计就告一段落了。接下来就该进入为了利用数据库中的数据而编写数据库应用程序的阶段了。只要数据库设计好了，设计一个带有用户界面的、能够操作其中数据的应用程序就很轻松了。在设计系统时，请诸位记住一个重要的顺序：优先设计数据库，然后再设计用户界面。

对数据库进行的操作的种类通常称为 CRUD。CRUD 由以下四种操作的英文名称的首字母组成，即记录的插入（CREATE）、获取（REFER）、更新（UPDATE）、删除（DELETE）。数据库应用程序只要能够对记录进行 CRUD 的操作就可以了。当然，为了满足用户的需求，为应用程序相应地增加统计、打印等功能的情况也是存在的。

由于 DBMS 具有自动生成主键和外键上的值的功能，所以在设计用户界面时，需要显示其余的字段，并要使 CRUD 操作能够通过按钮和菜单来完成。

图 8.14 展示了一个用四个按钮分别进行 CRUD 操作的例子。对于购买了多种商品的顾客，还可以通过“下一条”和“上一条”按钮交替地在界面上显示每种商品的名称、单价和销售量。请诸位注意一点，虽然数据被拆分成三个表存放，但是透过应用程序，用户感到他所处理的是一个相关数据的集合。界面中所显示的数据，是从三张表中用顺藤摸瓜的方式取出来的。

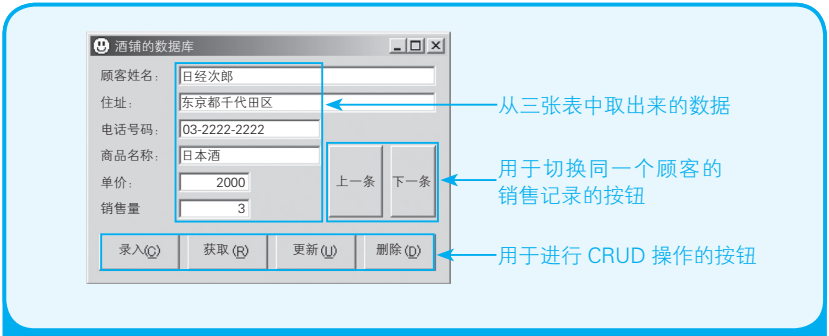

图 8.14　能够进行 CRUD 操作的用户界面示例

8.8　向 DBMS 发送 CRUD 操作的 SQL 语句

为了对数据库进行 CRUD 操作，就必须从应用程序向 DBMS 发送命令。这里所使用的命令就是 SQL 语言（Structural Query Language，结构化查询语言）。SQL 语言的标准是由 ISO（International Organization for Standardization，国际标准化组织）制订的。现在市面上几乎所有的 DBMS 都支持 SQL 语言。

一旦向 DBMS 发送了一条命令（SQL 语句），与此相应的操作就会立刻被执行。与 BASIC 或 C 语言等编程语言不同的是，使用 SQL 语言通常不需要定义变量或者考虑程序的执行流程。下面给诸位展示一

个 SQL 语句的例子，可以看出它和英文的句子很像。

```
SELECT 顾客姓名 , 住址 , 电话号码 , 商品名称 , 单价 , 销售量
FROM 顾客表 , 商品表 , 销售记录表
WHERE 顾客表 . 顾客姓名 = "日经次郎 "
AND 销售记录表 . 顾客 ID = 顾客表 . 顾客 ID
AND 销售记录表 . 商品 ID = 商品表 . 商品 ID ;
```

开头的 SELECT 所表示的就是 CRUD 中的 R 操作，也就从表中获取数据。在 SELECT 后面列出了想获取的字段的名字，用逗号分隔。在 FROM 后面，列出了用逗号分隔的表名。WHERE 后面则列出了查询条件。其中的 AND 表示多个查询条件是逻辑与的关系（条件 A 和条件 B 都成立）。而像“顾客表 . 顾客姓名”这样用“.”分隔的形式表示顾客姓名字段是属于顾客表的。在 SQL 语句的末尾放置一个分号表示语句的结束。

DBMS 不仅提供了手动向 DBMS 发送 SQL 语句的工具，而且还提供了通过可视化操作自动生成 SQL 语句的工具。将上述 SQL 语句发送到 DBMS 执行以后，结果如图 8.15 所示。日经次郎购入的商品一目了然。

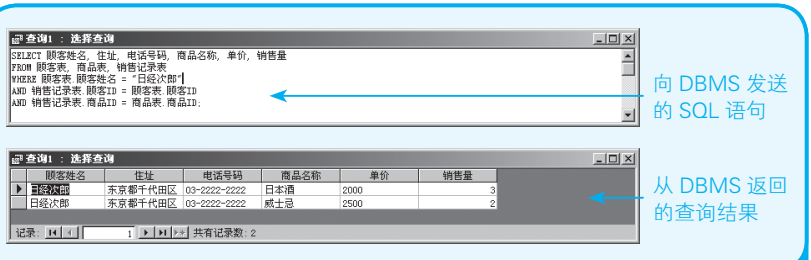

图 8.15　向 DBMS 发送 SQL 语句并执行后的结果

CRUD 中的 C、U、D 分别对应着 SQL 语言中的 INSERT（插入）、UPDATE（更新）、DELETE（删除）语句。在 SQL 语言中除了 CRUD 语句，还有新建数据库以及后面将要介绍的事务控制等语句，有兴趣的读者，可以查查 SQL 语言的文档等资料。

8.9 使用数据对象向 DBMS 发送 SQL 语句

在 Windows 应用程序中，向 DBMS 发送 SQL 语句时，一般情况下使用的都是被称为数据对象（Data Object）的软件组件（参考第 7 章所介绍的类）。一般的开发工具中也都包含了数据对象组件。在 Visual Basic 6.0 中，使用的是被称为 ADO（ActiveX Data Object，ActiveX 数据对象）的数据对象。

ADO 是以下几个类的统称，其中包括用于建立和 DBMS 连接的 Connection 类，向 DBMS 发送 SQL 语句的 Command 类以及存储 DBMS 返回结果的 Recordset 类等。图 8.14 所示的应用程序的代码如代码清单 8.1 所示。在程序启动时连接 DBMS，然后进行与各个按钮对应的 CRUD 操作，在程序结束时关闭与 DBMS 的连接。在使用 ADO 时必不可少的是 SQL 语句，其中主要是 SELECT 语句。而插入、更新、删除语句可以通过 Recordset 类所提供的 AddNew、Update、Delete 方法（类中所提供的函数）执行。可以认为这些方法在内部自动生成了 SQL 语句并发送给了 DBMS。诸位可以不去深究以下代码的细节，只要能抓住其大意就可以了。

代码清单 8.1　使用 ADO 访问数据库的示例程序（VB 6.0）

```
' 实例化 ADO 提供的类
Dim con As New ADODB.Connection
Dim cmd As New ADODB.Command
Dim rst As New ADODB.Recordset
```