5-1

TP312BA1
F76

Visual Basic .NET 数据库开发入门经典

Bill Forgey
Denise Gosnell 著
Matthew Reynolds
康 博 译

清华大学出版社

(京) 新登字 158 号
北京市版权局著作权合同登记号：01-2002-3002

内 容 简 介

众所周知，几乎所有的应用程序都必须进行数据访问。通过对本书的学习，您将了解到该如何构建能够有效使用数据库进行数据访问的 Visual Basic .NET 应用程序。

本书主要介绍了数据库设计的基本原理，如何查询数据库、如何在 Windows 应用程序中访问数据库中的数据，以及如何使用 Internet 和 Web 服务来进行远程数据访问。

本书适用于具有一定编程经验的准备开发数据库应用程序的编程人员。

Bill Forgey, Denise Gosnell, Matthew Reynolds: Beginning Visual Basic .NET Databases

EISBN: 1-861005-55-5

Copyright © 2001 by Wrox Press Ltd.

Authorized translation from the English language edition published by Wrox Press Ltd.

All rights reserved. For sale in the People's Republic of China only.

Chinese simplified language edition published by Tsinghua University Press.

本书中文简体字版由清华大学出版社和英国乐思出版公司合作出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，翻印必究。

本书封面贴有清华大学出版社激光防伪标签，无标签者不得销售。

书　　名： Visual Basic .NET 数据库开发入门经典

作　　者： Bill Forgey, Denise Gosnell, Matthew Reynolds 著　康博　译

出 版 者： 清华大学出版社(北京清华大学学研大厦，邮编 100084)

http://www.tup.tsinghua.edu.cn

责任编辑： 徐燕萍

封面设计： 康博

版式设计： 康博

印 刷 者： 北京市清华园胶印厂

发 行 者： 新华书店总店北京发行所

开　　本： 787×1092 1/16　**印张：** 37.5　**字数：** 959 千字

版　　次： 2002 年 7 月第 1 版　2002 年 7 月第 1 次印刷

书　　号： ISBN 7-302-05564-5/TP・3285

印　　数： 0001～5000

定　　价： 65.00 元

目　　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第1章　关系型数据库设计

本章将介绍一些设计和实现数据库的背景知识。绝大多数的应用程序，无论是用 Visual Basic .NET 还是用其他编程语言开发的，都包含有一定存储容量的数据库，所以，深入理解优秀数据库的设计原理是至关重要的。在对数据库的整体情况进行简短介绍之后，本章将开始设计和实现一个具体类型的数据库——关系型数据库。不必担心能否这么快就理解所有的数据库术语，到本章结束时，你将清楚地理解下列问题：

- 数据库的概念
- 关系型数据库和平面文件数据库间的比较
- 关系型数据库的优点
- 如何分析业务需求以确定数据库中应包含的信息
- 基于特定的业务需求，如何确定数据库中需要包含的适当元素
- 如何定义键码和关系
- 数据规范化的目标以及可以带来哪些好处
- 如何定义索引
- 综合这些知识创建物理数据库

最后，在设计关系型数据库时，我们再回顾一下本章的重要内容。

1.1　数据库的概念

从本质上讲，数据库是一种用有组织的方式存储数据的电子化手段。数据可以是企业或个体需要记录的任何东西，而在计算机出现之前，它们只能被记录在纸张上。数据一旦被存储进来后，数据库中的数据就可以被程序作为信息进行检索、处理并显示给浏览者。用来存储数据的实际数据库结构可以有多种不同的形式。在检索和修改信息时，每一种结构都有其特定的优势。在下一节，我们将讨论用平面文件结构和关系型数据库结构存储数据时的区别，然后还要讨论每一种方法的优劣所在。

1.1.1　平面文件与关系型数据库

平面文件是数据库的最基本形式——所有的信息都存储在单一的文件中。平面文件为用户所需存储的每个信息项提供一个字段。虽然平面文件很容易创建，但效率不是太高。由于包含了大量的重复信息，这种文件确实也浪费了大量的存储空间，在多个文件都包含关联信息的复杂系统中更是如此。这将加大信息维护和检索的难度。以前你使用过的电子数据表就是平面文件数据库的一个最常见的例子。为了进一步说明该如何组织平面文件中的数据，及为什么这种结构可能会引发问题，我们来假设一个例子。

假设你使用电子数据表记录客户的订单，如表 1-1 所示：

表 1-1

订单编号	订货日期	订货说明	数量	单 位	价格	客户名字	客户地址
1000	1-Aug-01	Tofu	1	40 - 100 g pkgs	23.25	Jane Doe	123 Somewhere St., Anytown, IN 46060 USA
1000	1-Aug-01	Jack's New England Clam Chowder	1	12 - 12 oz cans	9.65	Jane Doe	123 Somewhere St., Anytown, IN 46060 USA
1000	1-Aug-01	Grandma's Boysenberry Spread	3	12 - 8 oz jars	25	Jane Doe	123 Somewhere St., Anytown, IN 46060 USA
1001	2-Aug-01	Uncle Bob's Organic Dried Pears	1	12 - 1 lb pkgs	30	John Smith	345 Anywhere St., Somewhere, IN 46001 USA
1001	2-Aug-01	Tofu	1	40 - 100 g pkgs	23.25	John Smith	345 Anywhere St., Somewhere, IN 46001 USA

请注意这个电子数据表是如何容纳订单和客户信息的，例如，Jane Doe 订购了编号为 1000 的 ToFu(食品名称，下同。译者注)，还有 Jack 的 New England Clam Chowder、Grandma 的 Boysenberry Spread。每个数据项都列在电子数据表的一个单独的行中。进一步还可看出，正如上面订单中灰色部分指示的那样，订单编号、订单日期，连同 Jane Doe 的名字和地址被多次列出，因为她们订购了多种产品而导致其信息被存储了好多次。

所以说订单编号、订单日期、客户名字和客户地址字段包含了冗余的信息，换句话说，相同的信息重复出现在了若干个地方。冗余的信息造成了数据库的臃肿，而这正是因为它包含了由相同信息组成的多个数据项。在电子数据表中记录订单信息时，这也将造成额外的工作量，这完全是相同信息被重复记录之过。更为不幸的是，多次输入这些信息将极大地增加出错的机会，例如名字或地址拼写错误。

使用平面文件的另外一个问题就是维护。那将出现什么情况呢？例如，Jane Doe 迁居了，你就必须在电子数据表中修改她的地址，如果使用这种平面文件形式，那么，你就不得不多次修改她的地址——因为每次订货都会留下地址信息。如果她恰好是一个喜欢购物的客户，那就意味着上百处都需要修改。如果她的地址仅被保存在一个地方，那么只需修改一处就可

以了，但是在这个例子中却并非如此。在这个简单的例子中，你已经亲眼目睹了这种平面文件数据库的一些最普遍的问题：数据冗余和额外的维护需求。

在理解了平面文件数据库的概念，并且意识到了这种形式可能会在什么地方引起问题后，我们再来讨论一种克服了这些缺点的数据库类型——关系型数据库。用最简单的术语表示就是：关系型数据库可以被看作是信息项的集合，这些信息项被分解到不同组中，这些组以一种或多种方式相互关联。用数据库的术语表示，这些组常被称为表。这个概念听起来挺复杂的，但事实上不难理解。修改一下先前的示例，看一下如果使用关系型数据库它将会怎么样，很快你就会发现，大多数的概念并非复杂得难以理解。

回想一下在平面文件电子数据表中存储订单和客户信息的情景。每一笔订单都是由多个订单数据项组成的，并且每位客户发出多笔订单。关系型数据库在存储这样的信息时，会将其拆分到 3 个单独的表格中：Customers、Orders 和 OrderItems。如表 1-2 所示：

表　1-2

Customers	Orders	OrderItems
Customer_Id	Order_Id	Item_Id
Customer_First_Name	Customer_Id	Order_Id
Customer_Last_Name	Order_Date	Item_Description
Customer_Address1		Quantity_Ordered
Customer_City		Item_Price
Customer_State		Quantity_Per_Unit
Customer_Zip		
Customer_Country		

Customers 表为每个客户保存单独的记录项。Orders 表为每笔订单保存单独的记录项。最后，OrderItems 表为订单中的每一条产品订购保存单独的记录项。它的含义是，每一笔订单中可以包含一条或多条产品订购记录。这样，客户信息就从订单中分离出来而被单独保存了，订单的每一条产品订购记录也从订单中分离出来而被单独保存。我们可以看到，Orders 表中包含了一个 Customer_ID 字段，这个字段和 Customers 表中的 Customer_ID 字段相关联。我们还可以看到，OrderItems 表中包含了一个 Order_ID 字段，它可以和 Orders 表中的 Order_ID 字段相关联。我们将在本章的“定义表之间的关系”一节中讨论如何使表之间产生关联。目前，我们只需知道它是一种用来消除数据冗余的机制，可以解决在平面文件形式下看到的重复的客户名字和地址等问题。关系型数据库中没有这样的重复数据。例如，想修改 Jane Doe 的地址时，我们只需修改她在 Customers 表中惟一的数据项即可。更妙的是，当 Jane Doe 订购产品时，我们不用重复输入她的地址。如果她曾经在这里订过货，那么她的详细资料就已

此为试读，需要完整PDF请访问：www.ertongbook.com

经保存在 Customers 表的一个记录中了，我们只用从现存的记录中调用她的 Customer_ID 就可以了。另一方面，如果她是一个新客户，那么我们要做的就是将她的详细资料一次性添加到 Customers 表中，其信息将被保存下来以备再次订购时重新调用。

对这一点你可能会感到疑惑，我们是如何提取出上述表格中记录的，或者准确地讲，它们意味着什么？理解该问题时无需考虑太多细节，其关键之处在于领会关系型数据库形式背后的一个高层次概念：关系型数据库使用逻辑上相关的表来存储数据，这将消除数据冗余。只要理解了这一点，我们就可以转入下面的具体问题：如何确定数据库需求及如何根据这样的需求创建关系型数据库。

1.2 确定数据库的需求

在开始设计数据库之前，必须首先进行多种调查研究并进行分析，以便确定需要记录的信息。本节就来探讨加快这一过程所需要采取的步骤。

1.2.1 业务需求分析

确定数据库需求的第一步是彻底分析企业或个人对数据库的需求。本阶段的目标是花时间去了解用户的业务，充分掌握他们希望实现的目标。虽然我们急切地希望跳过这一步直接去创建数据库的物理结构，但是我们还没傻到采用如此差的设计策略的地步。为了构建一个真正适合用户需求的数据库，预先完整地理解用户的目标将是关键性的一步。企业的具体目标将对数据库的物理结构产生重大影响。

下面是一些可用于分析工作的指导原则：

- 分析当前所有将被新系统替换的电子数据库。找出当前系统运转良好的部分和需要改善的地方。考虑如下问题以确定数据库的关键字段(order_date、item_description 等)：哪些字段是经常使用的，哪些根本就不使用，还有，是否存在遗漏的字段。你将发现某些信息实际上用不到，可以从新数据库中省略掉。或者发现丢失了某些需要添加进来的关键信息。

- 开发人员与相关业务的工作人员面对面地交谈并分组讨论当前处理流程，或者使用处理流程生成的报表。设计一些问题探明他们希望实现的目标、需要记录的信息、当前系统中的缺陷，以及目前他们处理数据库的详细情况。

- 设法获取目前数据处理过程中所使用的表格和报表的副本——无论它们是书面的还是电子的。此后，确认信息是用抽样数据填写的，这样就可以进一步澄清数据所表示的信息类型。根据这些信息及从雇员处了解到的信息，就可以开始起草处理所需信息的“高级”需求表。随后这个需求表将帮助我们确定要创建数据库中的哪些字段和表。

- 认真分析现有报表，并根据你所需的报表和实地调查的结论来创建草案。这样你就会知道数据库所需的字段，就不会根据数据库内不存在的数据生成报表了。

- 确保认真归档分析的资料、你所了解到的信息、信息的出处、信息的重要性和其他任

何你认为有关的详细资料。

一旦完成了座谈，并且也分析了当前处理过程和系统，你应编写一份关于想要实现的总体目标摘要。如果把一个典型的假想企业作为一个样例，那么它的摘要应该和下面所述的大致相同：

- 数据库将存储下述信息：出售产品、公司库存清单、未完成的和全部的销售额及客户信息等。
- 公司有若干可供订购的产品。
- 一次可以订购一种或多种产品。通常情况下，一份订单可以订购 1~3 种产品，但是任一订单中的产品都不能超过 4 种。
- 一份订单仅属于一个客户，尽管它可以包含多种产品。
- 公司希望能够通过电话获取客户的订单，这样直接将订单信息输入到数据库应用程序中。为此，公司需要能够便捷地得到一些产品信息——例如库存数量和价格——以便于在客户准备下订单时就能确认产品的可供应量。
- 公司需要数据库生成各种报表，以便显示销售总额、等待执行的订单、缺货商品和每个客户的订单总计。
- 公司需要通过电话或电子邮件的方式针对客户进行特殊的促销活动

这份摘要能高度概括所需实现的目标。和你所服务的公司分享调查结果是非常必要的，以便公司能够就你是否正确理解了他们的需求给出反馈信息。据此你也可以将这份摘要交给完全不熟悉业务处理流程的人员，这样他们能够在一个抽象的层次上理解数据库的意图。这份摘要和收集提炼的详细资料将用于进一步的数据库设计。

1.2.2　确定需要记录的信息

现在我们已经和尽可能多的人探讨过，也研究了当前的处理流程，汇集了所有的调查结果，我们检查一下截至目前得到的结论，确定需要记录的数据元素。例如，通读你的所有资料，看到认为需要记录在数据库中的信息后，把这些信息跟其他所有可能作为一个字段的数据项分开，单独地记到某处。继续这个过程直到列出所有需要记录的信息。

记下这些信息后，就不用再担心任何特殊的订单或者成组的数据项。在这一例子中，仅仅列出了所有你认为应该记录的数据。同时，在每一个数据元素旁边列出一个示例，它显示了该元素可以容纳的典型值。稍后，当你为一个具体字段指定适当的数据类型时，这个示例将派上用场。现在我们仍然处于处理过程的早期，设法对数据库的内容有一个可靠全面的感性认识是非常重要的。此时我们无须考虑其准确性。

利用前面收集到的需求，得到了表 1-3 所示的字段列表：

表 1-3

商品标识符(例如 12345)	商品说明(例如 Tofu)	商品单价(例如 $23.25)
现有商品数量(例如 50)	商品度量单位 (例如 40 – 100 g pkgs)	客户名字 (例如 ne A. Doe)
客户编号(例如 123456)	客户地址 (例如 123 Somewhere St., Anytown, IN 46060 USA)	客户 Email (例如 jdoe@yahoo.com)
客户电话号码(例如 317-111-2222)	订货项标识符 (例如 12345 for Tofu)	订货数量 (例如 3)
客户订单编号 (例如 123456)	发送订单日期 (例如 Aug. 3, 2001)	订单编号 (例如 1000)
订货日期 (例如 Aug. 1, 2001)	订货单价 (例如 $23.25)	

表中的字段没有以特殊的顺序列出，后面括号内包含的是每一字段的典型示例。这个表包含的字段连接 customers、products 和 sales orders 之间的信息。

下一节将讨论如何使用这类列表来确定数据库的结构。

1.3 确定逻辑数据库设计

在明确了数据库的高级需求和目标后，就可以从理论上进行关系型数据库设计了——该阶段通常称为逻辑数据库设计。您需要设计出一个草图，在实际创建数据库之前，它可以为详细设计指明方向。

1.3.1 定义表(实体)和字段(属性)

创建逻辑数据库设计的第一步是定义表和字段。表也称为实体，它是相关信息的逻辑组合。回想一下，在本章开头，我们将平面文件电子数据表修改为表，最终得到了 3 个表，如表 1-4 所示：

表 1-4

Customers	Orders	OrderItems
Customer_Id	Order_Id	Item_Id
Customer_First_Name	Customer_Id	Order_Id
Customer_Last_Name	Order_Date	Item_Description
Customer_Address1		Quantity_Ordered
Customer_City		Item_Price
Customer_State		Quantity_Per_Unit
Customer_Zip		
Customer_Country		

字段也称为属性，它们是表中单独的数据元素——或者可以说属性是用来共同描述实体的。如表 1-3 所示，Customers 表为每个客户提供了若干个单独的信息位：Customer_Id、Customer_First_Name 和 Customer_Last_Name 等。我们把它们称为 Customers 表的字段，或者称之为描述 Customers 实体的属性。虽然可用两个术语中的任一个来表述，但是表和字段是最常使用的，所以本章将一直使用这两个术语。

识别表和字段

理解了表和字段的定义后，让我们回过头来看一下如何从最初分析阶段收集到的信息中识别表和字段。

看一下业务需求，先前我们确定了下列字段需要记录，如表 1-5 所示：

表　1-5

商品标识符 (例如 12345)	商品说明 (例如 Tofu)	商品单价 (例如 $23.25)
现有商品数量 (例如 50)	商品度量单位 (例如 40 – 100 g pkgs)	客户名字 (例如 Jane A. Doe)
客户编号 (例如 123456)	客户地址 (例如 123 Somewhere St., Anytown, IN 46060 USA)	客户 Email (例如 jdoe@yahoo.com)
客户电话号码 (例如 317-111-2222)	订货项标识符 (例如 12345 for Tofu)	订货数量 (例如 3)
客户订购编号 (例如 123456)	发送订单日期 (例如 Aug. 3, 2001)	订单编号 (例如 1000)
订货日期 (例如 Aug. 1, 2001)	订货单价 (例如 $23.25)	

现在可以做的就是仔细检查系统涉及的所有元素，并设法把它们拆分到表和字段中。为此，需要查看一下列表，看哪些东西可以轻松地聚集到一个表中——正如目前所知道的，表是相关数据的逻辑组合。这一步并不存在严格的科学内涵。我们可以尽力将数据分散到适当的表中，但是，能否做到这一点却依赖于收集到的字段的多少和用户需求的复杂程度，一般总要尝试多次才能得到合适的结果。但在这处理过程中，我们用不着使表和字段完全正确。随后讨论的内容将帮助我们对此进行修改，这些修改措施将确保数据库符合良好设计的要求。

将上述的示例转化为一组表。通览列表中的所有元素，看它们都涉及哪类信息？例如，查看上述列表可以看到，其中所有元素都是用于描述产品、客户或者订单的。用数据库术语表示，这一步称为定义实体。实体用来描述一组相关的信息。在确定实体后，接着就可以创建实体关系图(ERD)了，实体关系图显示了描述每个实体和每个实体与其他实体之间联系的信息。

为了创建 ERD，我们只需在一个单独的方框中列出每个实体的名称，在实体的下方列出和实体相对应的每一条信息。接着添加注释和箭头，它们描述了每个实体是如何与其他实体

发生联系的，例如可以描述这样的一个事实：一笔订单可以包含一种或多种产品。下面是一个已经完成的 ERD 示例。如图 1-1 所示。

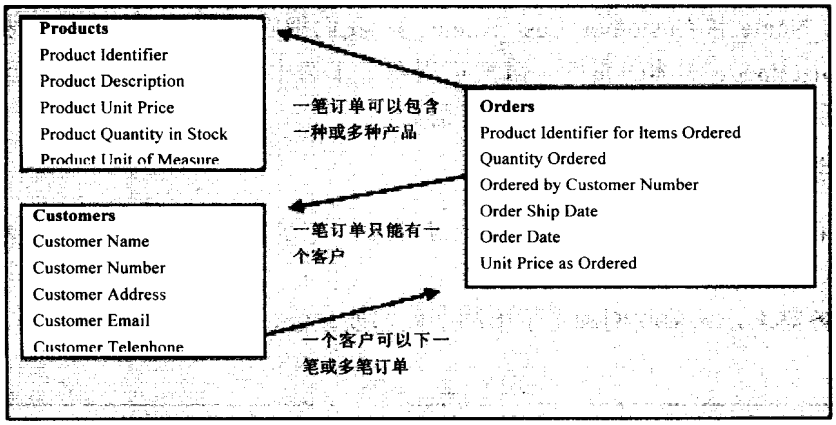

图 1-1

利用 ERD，可以毫不费力地表达数据库需要包含的表。例如，从对以上 ERD 的分析中，至少需要下面的这些表：

- Products——存储关于公司所有在售产品的信息
- Customers——存储每个客户的信息
- Orders——存储关于每一笔订单的信息

有了这些确定的表之后，接着给每个表分配字段。其实这也就是说：你需要把实体关系图中每个实体的描述信息转化成数据库中有意义的名称。

在开始之前，我们需要注意两条指导原则。第一，为每个可能需要的表准备一张新记录纸(或者是文件，如果你喜欢在计算机上工作的话)，把你考虑的每个字段都放到该表的记录纸上，使得该表看起来似乎容纳了最多的字段。第二，用有意义的名称命名字段，这些名称简练地描述了字段容纳的各类信息。因而，这类名称可以加快以后在应用程序中执行的检索任务。例如，你把客户编号称为 field1，把客户名字称为 field2。以后在应用程序中检索客户名字时，它将使你陷入尴尬的境地。除非你能记得每个字段名称所代表的含义，否则你将不得不胡乱地打开数据库字段查找客户名称字段。即使你真的知道 field2 是客户名字，那么程序代码也将被这些混乱无用的名称搅得乱作一团，使得程序更加难以理解。在很多情况下，如果第三方开发商在其应用程序中使用这样的数据库，可能会引起程序崩溃。使用适当的描述性字段名是良好数据库设计的一个组成部分，虽然它时常被忽略，但是它的确是一个永远也不能被轻视的问题。

命名字段时还有一个基本的技巧：使用适于阅读的名称。例如，不是使用全部小写字母的 customername，而是使用大小写交替出现的 CustomerName。这种大小写字母的混合有时被称为“驼峰规则”(camel case)。与采用单一的字符形式相比，它使得标识符更加易于阅读。空格通常不允许用于字段名称，但是下划线可起到空格的作用。可以使用下划线将 CustomerName 分隔成 Customer_Name。这种在标识符中用于分隔单词的标准被多种数据库语言所采用，使用这两种中的任意一种(CustomerName 或 Customer_Name)都同样能被认可。

在前面的示例中，我们使用了下划线。但是从现在开始，我们打算省略下划线。迄今为

止，作者是故意包含这两种命名格式的，为的是让读者能看到这两种格式，因此，读者可以任选一种。无论选择哪一种，都请保持前后协调一致，在整个数据库设计过程使用相同的命名标准。

将每个字段安排到最适当的表中，并赋予了有意义的名称之后，下一步的工作就是给每个字段指定示例数据、数据类型(文本、日期或数值等等)和估计的字段容量。如果它是一个文本字段，则列出它必须处理的字符数。如果它是一个数值字段，则列出它可能包含数据的最大范围。这正是前面汇集示例数据起作用的地方。你应该保证通过示例数据，使稍有文化的人就能够猜测得到该字段所包含信息的类型和容量。

牢记上述这些规则，我们列出迄今为止已确定的字段，这些字段被列在 3 个表中，分别如表 1-6、1-7、1-8 所示：

表　1-6

PRODUCTS 表

字　段	示　例	数 据 类 型	数据长度估计
ProductIdentifier	12345	Numeric	不带小数点的正数
ProductDescription	Tofu	Text	25 个字符
ProductUnitPrice	$23.25	Currency	$00.00~$10,000.00
ProductQuantityOnHand	50	Numeric	0~ 9,999
ProductUnitOfMeasure	40 – 100 g pkgs	Text	25 个字符

表　1-7

CUSTOMERS 表

字　段	示　例	数 据 类 型	数据长度估计
CustomerNumber	123456	Numeric	不带小数点的正数
CustomerName	Jane A. Doe	Text	45 个字符
CustomerAddress	123 Somewhere St., Anytown, IN 46060 USA	Text	65 个字符
Customer Telephone	317-111-2222	Text	12 个字符
CustomerEmail	jdoe@yahoo.com	Text	50 个字符

此为试读，需要完整PDF请访问：www.ertongbook.com