李洪波　邹海林　李洪国　编著

企业级数据库集成应用系统开发

清华大学出版社

李洪波　邹海林　李洪国　编著

企业级数据库集成应用系统开发

清华大学出版社
北京

内 容 简 介

全书共分 9 章，第 1 章介绍商业银行数据库开发，第 2 章介绍电影数据开发实训，第 3 章介绍 SQL Server 2008 R2 应用基础，第 4 章介绍 SQL Server 2008 R2 应用实训，第 5 章介绍运用 Visual C++ 2008 DLL 开发商业银行数据库访问层，第 6 章介绍运用 Visual C++ 2008 ATL COM 开发商业银行业务逻辑层，第 7 章介绍运用 Visual C++ 2008 MFC 开发商业银行用户应用程序，第 8 章介绍商业银行 Web 页面数据输出到 Excel 和 PDF 文件，第 9 章介绍了基于 Visual C# .NET 三层应用程序开发。

本书适合作为高等院校计算机专业、信息管理与信息系统专业的教材，也可作为企业级数据库集成应用系统开发爱好者的参考书。本书配有课件、源程序、模块文件，可从清华大学出版社网站（www.tup.com.cn）下载。

图书在版编目（CIP）数据

企业级数据库集成应用系统开发/李洪波，邹海林，李洪国编著. —北京：清华大学出版社，2014
ISBN 978-7-302-35534-2

Ⅰ. ①企… Ⅱ. ①李… ②邹… ③李… Ⅲ. ①数据库系统－系统开发 Ⅳ. ①TP311.13

中国版本图书馆 CIP 数据核字（2014）第 034929 号

责任编辑：白立军
封面设计：傅瑞学
责任校对：李建庄
责任印制：王静怡

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 装 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：27.75　　**字　　数**：691 千字
版　　次：2014 年 6 月第 1 版　　**印　　次**：2014 年 6 月第 1 次印刷
印　　数：1～2000
定　　价：49.50 元

产品编号：051703-01

前言 Foreword

当前，随着信息技术的飞速发展，人们已被带入了网络时代。数据库应用系统开发应与网络环境相适应，建立网络环境下的数据库应用系统。当今的网络数据库应用系统模式，主要有 Client/Server(C/S)模式、Browser/Server(B/S)模式或两者的混合 3 种模式。C/S 应用模式基于 Windows 应用，而 B/S 模式基于 Web 应用。

现代的数据库应用系统也是一个集成应用系统。首先数据库系统自身的查询语言并不能表达所有查询要求，因为 SQL 没有提供通用编程语言一样的表达能力。也就是说，有可能存在一些查询，可以用 C、C++、Java 或 COBOL 写出，而用 SQL 做不到。要写出这样的查询，需要将 SQL 嵌入到一种更强大的语言中。其次，非声明性的动作不能用 SQL 实现。例如，打印一份报告，和用户交互，或把一次查询的结果送到一个图形用户界面中。一个应用程序通常包含多个部件，查询或者更新只是其中一个部件，而其他部件则可用通用编程语言实现。对于一个集成的应用来说，用编程语言写出的程序必须能够访问数据库。

数据库应用的一个重要方面是建立在数据库之上的管理信息系统，用数据库组织和管理信息系统中的数据资源。开发一个企业级管理系统的典型架构模式为四层体系结构，即从数据库、数据访问层、业务逻辑层再到用户层，以满足开发、运行和维护管理信息系统工程化的需要。依赖企业级数据库开发一个信息系统，要将其分解为若干模块，这需要一个团队来完成，团队中的每个成员负责其中的部分模块。每个成员依据系统设计蓝图，编程实现自己负责的模块。当各个模块完成时，需要进行联调。

综上所述，企业级数据库应用系统从环境角度看为网络环境，从功能角度看由若干功能模块集成，从层次的角度看由前后相互衔接的分层模块协作完成，这种数据库应用系统被称之为集成应用系统。为此，模拟商业银行企业数据库应用的真实环境，编写企业级数据库集成应用开发系统图书，以缩小毕业生实际应用能力与企业实际需求之间的差距，向社会输出具有岗位胜任能力的合格劳动力。这样，大学生在校期间就能直观地形成工程化、模块化的概念，使得他们进入企业后无须过长时间的培训即可上手，降低企业用人成本，提高毕业生的社会认可度。

本书设计的总体思路是依赖商业银行和电影明星两个案例，面向 Windows 和 Web 两种应用，依托商业银行的支行信息管理功能，基于分层架构体系，涉及 Client/Server 和 Browser/Server 两种模式，运用 Visual C++ 2008、C# .NET 和 ASP.NET 开发工具，开发数据库集成应用系统。这样，可提高学生集成应用的能力，培养学生独立地分析问题和解决问题的能力。

本书的特点如下。

1. 贯通动态 SQL 于各层

动态 SQL 贯穿数据访问层、业务逻辑层以及用户层，实现用户对数据库的任意操作。

2. 融合动态链接库和 ATL COM 组件两类软构件

第 5 章数据访问层以及第 9 章数据访问层和业务逻辑层，采用动态链接库开发。第 6 章业务逻辑层采用 ATL COM 组件技术开发。

3. 面向 Web 应用和 Windows 桌面两类用户

第 8 章 Web 页面输出到 Excel 文件和 PDF 文件运用 ASP.NET 开发 Web 应用程序。第 7 章 MFC 开发的应用程序和第 9 章 C# .NET 开发的应用程序均为 Windows 桌面应用。此外，所有的 Win32 客户端应用测试均为 Windows 桌面应用。

4. 兼具 C/S 和 B/S 两种应用模式

第 3 章开篇是远程客户端连接数据库服务器，为典型的 C/S 结构。第 5～7 章以及第 9 章的 Win32 客户端测试应用程序或 Windows 窗体应用程序，属于 C/S 结构。第 8 章的 Web 应用属于 B/S 模式。

5. 集成 Excel 和 PDF 两种输出方式

第 8 章提供当前页面数据输出到 Excel 文件和 PDF 文件两种输出方式。

6. 贯通数据访问层、业务逻辑层和应用层三层架构

本书提供 3 套完整的三层架构应用。第一套是第 5～7 章，即基于 MFC 的 Windows 桌面应用，沿着数据访问层、业务逻辑层、Windows 桌面应用路线展开。第二套是第 8 章，即基于 ASP.NET 的 Web 应用，沿着数据访问层、业务逻辑层、Web 应用路线展开。第 3 套是第 9 章基于 C# 开发的桌面应用，沿着数据访问层、业务逻辑层、Windows 桌面应用路线展开。

7. 运用 ASP.NET、C++ 和 C# 3 种开发语言

第 8 章采用 Visual Studio 2010 的 ASP .NET 开发，第 5～7 章采用 Visual C++ 2008 开发，第 9 章运用 Visual Studio 2010 的 C# .NET 开发。

8. 互补 2 个典型案例

全书内容的展开依托商业银行的支行信息管理，自训题依托电影数据库展开，实现商业银行和电影数据库之间的互补。

本书全篇由李洪波统稿，其中第 1 章由王惠敏编写，第 2～8 章由李洪波编写，第 9 章由李洪国编写。第 1～8 章的程序由李洪波完成，第 9 章的程序由李洪国完成。鲁东大学信息管理与信息系统专业 2010 级的贾斌和杨虹 2 位同学参与部分文档的搜集、整理和模块测试。全书由邹海林教授主审。

本书配套的课件、源程序、模块文件、案例数据库可在清华大学出版社网站(www.tup.com.cn)下载。

感谢清华大学出版社编辑的辛苦工作，使得本书能与读者见面。书中难免有疏漏之处，敬请读者来信指正。

编者

2014 年 3 月

目录 Contents

第 1 章 商业银行数据库开发 …… 1
1.1 数据库分析与设计 …… 2
1.1.1 需求分析 …… 2
1.1.2 概念设计 …… 2
1.1.3 逻辑设计 …… 6
1.1.4 物理设计 …… 6
1.1.5 完整性设计 …… 7
1.1.6 安全性设计 …… 14
1.2 商业银行数据库编程 …… 18
1.2.1 商业银行数据库编程 …… 20
1.2.2 数据库建模相关的 Transact-SQL 语法基础 …… 36
1.3 SQL Server 2008 R2 版本、组件、安装、启动与登录 …… 92
1.3.1 SQL Server 2008 R2 版本和组件 …… 92
1.3.2 SQL Server 2008 R2 的安装 …… 95

第 2 章 电影数据库开发实训 …… 106
2.1 需求分析 …… 106
2.2 概念设计 …… 106
2.3 逻辑设计 …… 107
2.4 实训题目 …… 108

第 3 章 SQL Server 2008 R2 应用基础 …… 109
3.1 开启 Internet 远程连接 …… 109
3.1.1 SQL Server Management Studio 里的数据库服务器配置 …… 109
3.1.2 SQL Server Configuration Manager 的配置 …… 113
3.1.3 防火墙设置 …… 116
3.2 附加与分离 …… 116
3.2.1 采用 Transact-SQL 分离与附加 branch 数据库 …… 116
3.2.2 采用 SQL Server Management Studio 分离与附加 branch 数据库 …… 117
3.2.3 sp_detach_db Transact-SQL 基础 …… 121
3.3 备份与恢复 …… 122

3.3.1 利用 SQL Server Management Studio 完全备份 branch 数据库 …… 124
3.3.2 利用 BACKUP 命令备份 branch 数据库 …… 126
3.3.3 利用 SQL Server Management Studio 还原 branch 数据库 …… 128
3.3.4 利用 RESTORE 命令还原 branch 数据库 …… 129
3.3.5 BACKUP Transact-SQL 基础 …… 129
3.3.6 RESTORE Transact-SQL 基础 …… 145
3.4 简单查询 …… 155
3.4.1 SELECT 子句 …… 156
3.4.2 WHERE 子句 …… 159
3.4.3 FROM 子句 …… 159
3.4.4 聚集函数 …… 166
3.4.5 GROUP BY 子句 …… 167
3.4.6 HAVING 子句 …… 171
3.4.7 ORDER BY 子句 …… 171
3.4.8 IDENTITY 属性 …… 173
3.4.9 集合操作符 …… 174
3.4.10 CASE 表达式 …… 176
3.4.11 字符串模式匹配 …… 182
3.5 嵌套子查询 …… 186
3.5.1 集合间的成员关系——[NOT] in …… 187
3.5.2 集合间的比较——SOME|ANY|ALL …… 188
3.5.3 测试空关系——[NOT] EXISETS …… 189
3.6 视图 …… 190
3.7 数据库修改 …… 200
3.7.1 Insert …… 200
3.7.2 Delete …… 209
3.7.3 UPDATE …… 214
3.8 SQL 程序基本结构 …… 225
3.8.1 顺序结构 BEGIN…END(Transact-SQL) …… 225
3.8.2 选择结构 IF…ELSE(Transact-SQL) …… 226
3.8.3 循环结构 WHILE(Transact-SQL) …… 227
3.8.4 向客户端输出用户消息 …… 229
3.8.5 用户自定义函数 CREATE FUNCTION …… 230
3.8.6 存储过程 …… 244

第 4 章 SQL Server 2008 R2 应用实训 …… 257

第 5 章 运用 Visual C++ 2008 DLL 开发商业银行数据访问层 …… 259
5.1 概论 …… 259

5.2 非 MFC DLL …… 260
5.2.1 DLL 导出函数 …… 260
5.2.2 DLL 导出类 …… 266
5.3 MFC 规则 DLL …… 269
5.3.1 概述 …… 269
5.3.2 MFC 规则 DLL 的创建 …… 270
5.3.3 规则 MFC DLL 的对外简单输出 …… 271
5.3.4 MFC 规则 DLL 的调用 …… 275
5.4 数据访问层的开发 …… 276
5.4.1 类 ADOConn 的头文件——ADOConn.h …… 277
5.4.2 类 ADOConn 的源文件——ADOConn.cpp …… 279
5.4.3 Win32 控制台应用测试程序 …… 283
5.5 自训题 …… 284

第 6 章 运用 Visual C++ 2008 ATL COM 开发商业银行业务逻辑层 …… 285
6.1 业务逻辑层概述 …… 285
6.2 ATL COM 概述 …… 286
6.2.1 什么是 ATL …… 286
6.2.2 ATL 基本技术 …… 288
6.2.3 ATL 基本使用 …… 290
6.2.4 Win32 客户端应用程序调用 ATL COM 组件 …… 296
6.3 开发商业银行业务逻辑层组件 BusinessLogic …… 299
6.3.1 接口 IBranchBusiness 对外发布的方法 …… 299
6.3.2 接口 IBranchBusiness 对外发布的方法具体实现 …… 301
6.3.3 接口 IBranchBusiness 的 Win32 控制台应用程序测试客户端 …… 305
6.4 自训题 …… 307

第 7 章 运用 Visual C++ 2008 MFC 开发商业银行用户应用程序 …… 308
7.1 MFC 技术基础 …… 308
7.1.1 概述 …… 308
7.1.2 常用的 MFC …… 310
7.1.3 MFC 类别 …… 312
7.1.4 MFC 相关 …… 318
7.2 基于 MFC 的 BranchWindowsApplication 应用程序开发 …… 321
7.2.1 MFC 应用程序 BranchWindowsApplication 项目的建立 …… 321
7.2.2 商业银行数据库的连接与断开——CBranchWindowsApplicationApp 应用类的完善 …… 324
7.2.3 “支行管理”菜单项的添加与命令消息响应函数 OnBranch …… 328
7.2.4 对话框的添加与相关消息响应实现 …… 330
7.2.5 导出到 Excel 文件功能的实现 …… 346

7.2.6 支行管理运行结果界面 …… 351
7.3 自训题 …… 352

第 8 章 商业银行 Web 页面数据输出到 Excel 和 PDF 文件 …… 353
8.1 ASP.NET 数据源控件 …… 353
8.1.1 ASP.NET 网页开发的基础模型 …… 353
8.1.2 ASP.NET 的常用数据源控件 …… 355
8.2 ASP.NET 数据绑定控件 …… 369
8.2.1 GridView 数据绑定控件 …… 370
8.2.2 DetailsView 数据绑定控件 …… 381
8.2.3 FormView 数据绑定控件 …… 385
8.2.4 ListView 数据绑定控件 …… 387
8.3 Web 页面数据输出到 Excel 和 PDF 文件的实现 …… 394
8.3.1 业务逻辑层代理类的开发 …… 398
8.3.2 支行信息浏览与导出界面的实现 …… 400
8.3.3 支行信息浏览与导出功能的后台 C# 实现 …… 404
8.4 自训题 …… 409

第 9 章 基于 Visual C#.NET 三层应用程序开发 …… 410
9.1 数据访问层实现 …… 411
9.2 业务逻辑层实现 …… 415
9.3 Windows 桌面应用的实现 …… 422
9.3.1 连接窗体设计 …… 422
9.3.2 编辑窗体设计 …… 425
9.3.3 Program 类设计 …… 429
9.3.4 基于 Visual C#. NET 的 Excel 访问输出实现 …… 429
9.4 自训题 …… 431

参考文献 …… 432

第1章

商业银行数据库开发

数据库服务器的开发首先需进行周密的数据库设计。数据库设计(Database Design)是指根据用户的需求,在某一具体的数据库管理系统上,设计数据库的结构和建立数据库的过程。它要求能够更有效地表达语义关系的数据模型,为各阶段的设计提供自动或半自动的设计工具和集成化的开发环境,使数据库的设计更加工程化、更加规范化和更加方便易行,使得在数据库的设计中充分体现软件工程的先进思想和方法。

一般,数据库的设计过程大致可分为5个步骤。

1. 需求分析

调查和分析用户的业务活动和数据的使用情况,弄清所用数据的种类、范围、数量以及它们在业务活动中交流的情况,确定用户对数据库系统的使用要求和各种约束条件等,形成用户需求规约。

2. 概念设计

对用户要求描述的现实世界(可能是一个工厂、一个商场或者一个学校等),通过对其分类、聚集和概括,建立抽象的概念数据模型。这个概念模型应反映现实世界各部门的信息结构、信息流动情况、信息间的互相制约关系以及各部门对信息存储、查询和加工的要求等。所建立的模型应避开数据库在计算机上的具体实现细节,用一种抽象的形式表示出来。以扩充的实体——联系模型(E-R模型)方法为例,第一步先明确现实世界各部门所含的各种实体及其属性、实体间的联系以及对信息的制约条件等,从而给出各部门内所用信息的局部描述(在数据库中称为用户的局部视图)。第二步再将前面得到的多个用户的局部视图集成为一个全局视图,即用户要描述的现实世界的概念数据模型。

3. 逻辑设计

逻辑设计的主要工作是将现实世界的概念数据模型设计成数据库的一种逻辑模式,即适应于某种特定数据库管理系统所支持的逻辑数据模式。与此同时,可能还需为各种数据处理应用领域产生相应的逻辑子模式。这一步设计的结果就是"逻辑数据库"。

4. 物理设计

根据特定数据库管理系统所提供的多种存储结构和存取方法等依赖于具体计算机结构的各项物理设计措施,对具体的应用任务选定最合适的物理存储结构(包括文件类型、索引结构和数据的存放次序与位逻辑等)、存取方法和存取路径等。这一步设计的结果就是"物理数据库"。

5. 验证设计

在上述设计的基础上,收集数据并具体建立一个数据库,运行一些典型的应用任务来验证数据库设计的正确性和合理性。一般地,一个大型数据库的设计过程往往需要经过多次

循环反复。当设计的某步发现问题时，可能就需要返回到前面去进行修改。因此，在做上述数据库设计时就应考虑今后修改设计的可能性和方便性。

6. 运行与维护设计

在数据库系统正式投入运行的过程中，必须不断地对其进行设计过程各个阶段的设计描述评价、调整与修改。

下面介绍本商业银行数据库的需求分析、概念设计、逻辑设计和物理设计。

1.1 数据库分析与设计

1.1.1 需求分析

用户需求的最初规格说明可以基于同数据库用户的交谈和设计者自己对企业的分析。这一设计阶段产生的描述是定义数据库概念结构的基础。下面列出了银行企业的主要特点。

(1) 银行有多个支行。每个支行位于某个城市，由唯一的网点名标识。银行监控每一个支行的资产。

(2) 银行的客户通过 customer_ID(身份证号)来唯一标识。银行存储每个客户的姓名、联系电话和其居住的街道和城市。客户可以有存款，也可以有贷款。客户可能同银行的某个雇员发生联系，该员工作为此客户的贷款负责人或私人助理。

(3) 银行雇员通过 employee_ID(身份证号)来唯一标识。银行管理结构存储每个员工的姓名、联系电话、居住的街道和城市及其经理的 employee_ID 号码。银行还需要知道雇员开始工作的日期，由此日期可以推知雇员的雇佣期。

(4) 银行提供两类存款账户——支票账户和储蓄账户。账户可以由两个或两个以上客户共有，一个客户也可以有两个或两个以上的账户。每个账户被赋予唯一的账户号。银行记录每一个账户的余额和每个账户拥有者访问该账户的最近日期。另外，每个储蓄存款账户有其利率，而每个支票账户也可以有透支额。记载每个储蓄账户和支票账户中每次支出或者存入的金额、日期、支取/存入方式等的交易记录。

(5) 每笔贷款由某个支行发放，能被一个或多个客户共有。一笔贷款用唯一的贷款号标识。银行需要知道每笔贷款所贷金额以及逐次还款情况。虽然贷款的还款号并不能唯一地标识所有贷款中的某个特定还款，但可以唯一标识对某一贷款的所还款项。对每次的还款需要记录其日期和金额。

1.1.2 概念设计

概念模型是数据库系统的核心和基础。由于各个机器上实现的 DBMS 软件都是基于某种数据模型的，但是在具体机器上实现的模型都有许多严格的限制。而现实应用环境是复杂多变的，如果把现实世界中的事物直接转换为机器中的对象，就非常不方便。因此，人们研究把现实世界中的事物抽象为不依赖于具体机器的信息结构，又接近人们的思维，并具有丰富语义的概念模型，然后再把概念模型转换为具体的机器上 DBMS 支持的数据模型。概念模型的描述工具通常是使用 E-R 模型图。该模型不依赖于具体的

硬件环境和 DBMS。

概念结构是对现实世界的一种抽象。抽象是对实际的人、物、事和概念进行人为处理，抽取所关心的本质特性，忽略非本质的细节，并把这些特性用各种概念精确地加以描述，这些概念组成了某种模型。通过概念设计得到的概念模型是从现实世界的角度对所要解决的问题的描述，不依赖于具体的硬件环境和 DBMS。

E-R 图为实体-联系图，提供了表示实体型、属性和联系的方法，用来描述现实世界的概念模型。构成 E-R 图的基本要素是实体、属性和联系，其概念及其定义如表 1.1 所示。E-R 图中用到的符号如图 1.1 所示。

表 1.1　E-R 模型概念表

概　念	定　义
实体	现实世界中可区别于其他对象的“事物”或“对象”
实体集	具有相同性质的属性集合
强实体集 弱实体集	一个实体集可能没有足够的属性去形成主码用矩形表示，这样的实体集为弱实体集；与此相对，有主码的实体集称为强实体集
属性	属性是实体集中每个成员所拥有的描述性性质。为某实体集设计一个属性表明数据库为该实体集中的每个实体存储相似的信息，但每个实体在每个属性上都有各自的值
域	每个属性都有一个可取值的集合，称为该属性的域
联系	多个实体间相互关联
角色	实体在联系中作用
联系集	同类联系的集合
识别联系	弱实体集必须与另一个称为标识实体集或者属主实体集的实体集关联才有意义。将弱实体集与其标识实体集相联系的联系为识别联系
映射基数	指明一个实体通过一个联系集能同时与多少实体相关联。映射基数在描述二元联系时很有用。二元联系映射基数有“一对一”、“一对多”、“多对一”和“多对多”4 种
完全参与	实体集中的每个实体都参与到联系集的至少一个联系中
部分参与	实体集中只有部分实体都参与到联系集的联系中
一般化-高层实体 特殊化-底层实体	实体集中有时包含一些子集，子集中的实体在某些方面区别于实体集中的其他实体。该实体集称为高层实体，而其中的子集称为底层实体。由底层实体到高层实体是一般化的过程，相反是特殊化的过程

数据需求规格说明是建造数据库概念模式的出发点。根据需求分析所列的特点标识实体集及其属性，如表 1.2 所示。

根据表 1.2 实体集的设计模式，定义如表 1.3 的联系集、映射基数和参与约束。根据表 1.3 给出的抽象概念，结合图 E-R 图中的符号表示，给出如图 1.2 所示的商业银行 E-R 图。在图 1.2 中，“雇员”实体使用了两次，其中仅有“雇员”实体名称但没有属性的那个矩形是为了避免连线之间的交叉。

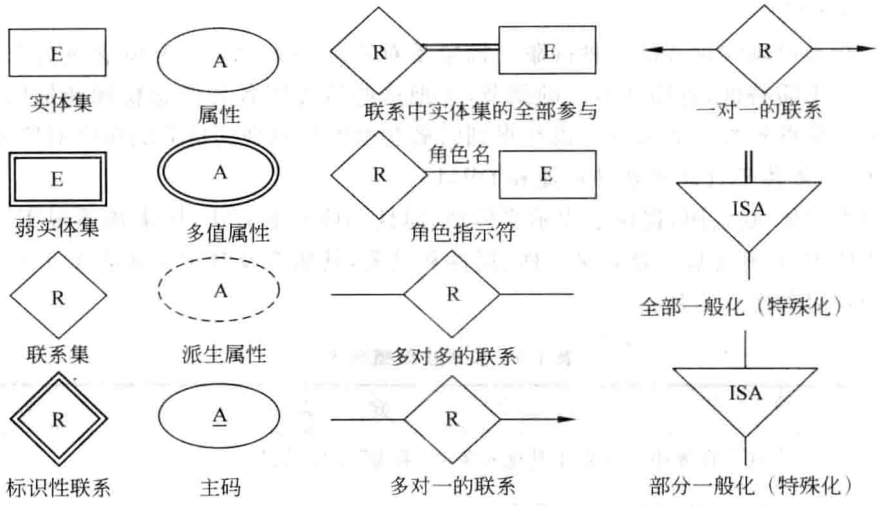

图 1.1 E-R 图中的符号

表 1.2 商业银行数据库的实体集

实体集		属性
强	支行	网点名、城市和总资产
	人	姓名、电话、城市、街道和身份证号
	客户	继承人的所有属性
	雇员	除继承人的所有属性外，还有入职时间以及派生属性——工作年限
	贷款账户	贷款号、贷款金额、开户日期、账户余额
	存款账户	账号、开户日期、账户类型、账户余额
	储蓄账户	利率
	支票账户	透支额
	出纳员	出纳号、密码、身份证号
弱	还款记录	还款号、日期、摘要、币种、钞/汇、金额、余额、出纳员
	存取记录	序号、日期、摘要、币种、钞/汇、金额、余额、出纳员、备考

表 1.3 商业银行数据库的联系集及其约束

联系集	参与实体集	约束	
		映射基数约束	参与约束
借有	客户、贷款账户	多对多	客户部分参与、贷款账户完全参与
借出	贷款账户、支行	多对一	支行部分参与、贷款账户完全参与
联络	雇员、客户	多对一	雇员部分参与、客户完全参与
存有	客户、存款账户	多对多	客户部分参与、贷款账户完全参与

续表

联系集	参与实体集	约　束	
		映射基数约束	参与约束
拥有	存款账户、支行	多对一	支行部分参与、存款账户完全参与
属于	雇员、支行	多对一	雇员完全参与、支行部分参与
存取	存款账户、存取记录	一对多	交易记录完全参与、存款账户部分参与
还贷	贷款账户、还款	一对多	还款完全参与、贷款账户部分参与
操作 1	出纳员、存取记录	一对多	存取记录完全参与、出纳员部分参与
操作 2	出纳员、还款记录	一对多	还款记录完全参与、出纳员部分参与
转还	存款记录、还款记录	一对一	还款记录与存款记录均部分参与
是	出纳员、雇员	一对一	出纳员完全参与、雇员部分参与

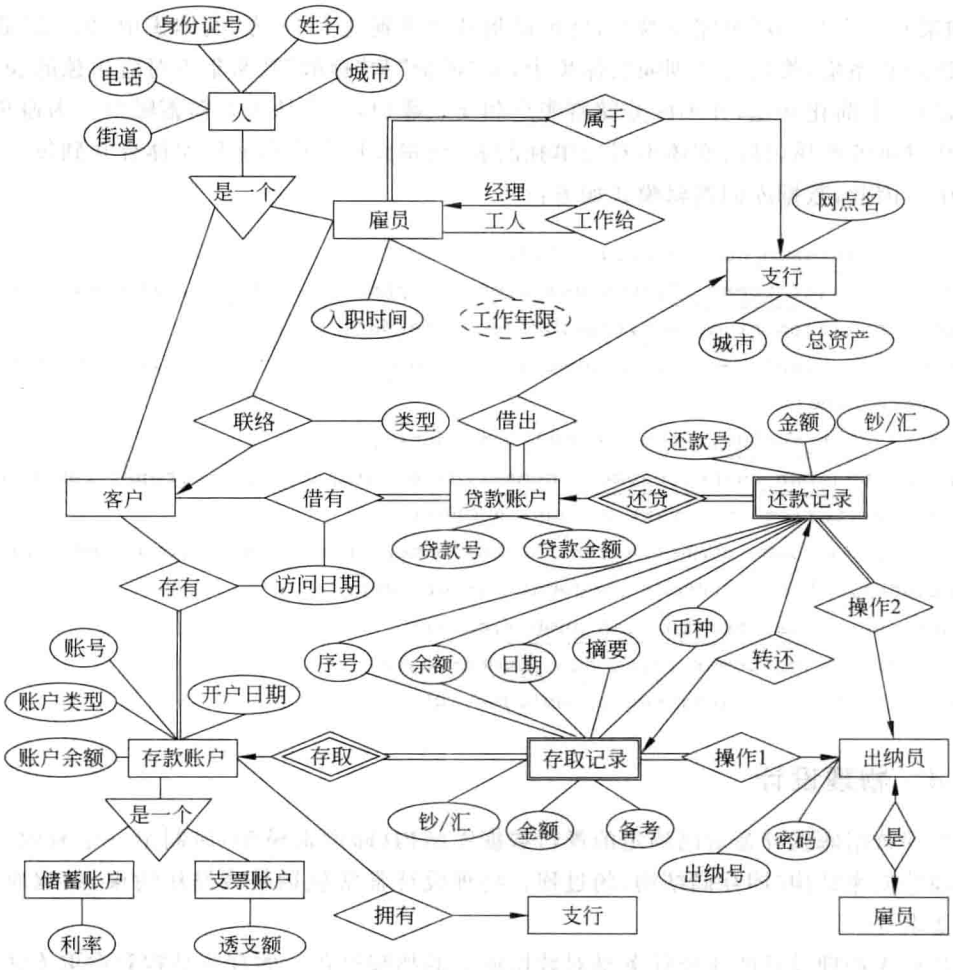

图 1.2　商业银行的 E-R 图

1.1.3 逻辑设计

符合 E-R 数据库模式的数据库可以表示为一些关系模式的集合。数据库的每个实体集和联系集都有唯一的关系模式与之对应，关系模式名即为相应的实体集或联系集名称。

E-R 模型和关系数据库模型都是现实世界的逻辑表示。由于两种模式采用类似的设计原则，我们可以将 E-R 设计转换为关系设计，即逻辑设计。逻辑设计的目的是从概念模型导出特定的数据库的逻辑结构(数据库的模式和外模式)，该逻辑结构可以被 DBMS 处理，这些模式在功能、性能、完整性和一致性约束及数据库可扩充性等方面均应满足用户提出的要求。根据系统的 E-R 图进行逻辑结构设计，就是以最小化冗余和方便查询为目标，将 E-R 模型转换为关系模型，即二维表。

一个强实体集对应单独一个表。对于联系集，根据其映射基数决定联系集是否对应一个表。对于多对多的联系集对应一个表，如“借有”和“存有”联系集；对于一对多或者多对一的联系集，可以将其合并到“多方”实体集中，如“借出”、“联络”、“拥有”、“属于”、“存取”、“还贷”、“支出 1”、“支出 2”、“操作 1”和“操作 2”联系集就不对应单独的表；对于一对一的联系集，如果有一个“一方”是完全参与，则可以将其合并到这个“一方”实体集中，如“是”联系集。对于识别联系集，将其合并到弱实体集中，如“还贷”和“存取”联系集不对应单独的表。对于存款账户，为简化问题，并不区分储蓄账户和支票账户，一律认为是储蓄账户。为避免连接，按属性继承的性质，高层实体不对应单独的表，而将高层实体和底层实体合并到每一个底层实体中。因此，数据库的逻辑模式如下：

```
branch=(branch_name,city,assets);
customer=(customer_ID,customer_name,telephone_num,street,city,employee_ID);
account=(account_number,type,branch_name,date,balance);
employee=(employee_ID,employee_name,telephone_num,street,city,enter_date,
branch_name);
loan=(loan_number,amount,branch_name,date);
payment=(loan_number,payment_number,date,currency_system,paper_remit,amount,
balance,teller_id,remark,account_number);
access=(account_number,access_number,date,currency_system,paper_remit,amount,
balance,teller_id,remark,account_number_pay);
borrower=(customer_ID,loan_number,date);
depositor=(customer_ID,account_number,date);
teller=(teller_id,password,employee_ID)
```

1.1.4 物理设计

物理数据库设计是对已确定的逻辑数据库结构(即逻辑模型)研制出一个有效、可实现的物理数据库结构(即存储结构)的过程。物理设计常常包括某些操作约束，如响应时间与存储要求等。

数据库物理设计的主要任务是对数据库中的物理设备上的存放结构和存取方法进行设计。数据库物理结构依赖于给定的计算机系统，而且与具体选用的 DBMS 密切相关。

1. 物理设计的步骤

物理设计可分为 5 步，前 3 步为结构设计，后两步为约束设计和程序设计。

(1) 存储记录的格式设计。对数据项类型特征进行分析，对存储记录进行格式化，决定如何进行数据压缩或代码化。可使用“垂直分割方法”。对含有较多属性的关系，根据其中属性的使用频率进行分割，或使用“水平分割方法”，对含有较多记录的关系，按某些关系进行分割。把分割后的关系定义在相同或不同类型的物理设备上，或相同设备的不同区域上，从而使访问数据库的代价最小，提高数据库的性能。

(2) 存储方法设计。物理设计中最重要的一个考虑是，把存储记录在全范围内进行物理安排，存放的方式有顺序存放、杂凑存放、索引存放及聚簇存放等。

(3) 访问方法设计。访问方法设计为存储在物理设备上的数据提供存储结构和咨询路径，这与数据库管理系统有很大关系。

(4) 完整性和安全性考虑。根据逻辑设计提供的对数据库的约束条件、具体的 DBMS 的性能特征和硬件环境，设计数据库的完整性和安全性措施。

2. 物理设计的性能

根据数据库性能开销(cost)，即时间、空间及可能的费用来衡量，则在数据库应用系统生命周期中总的开销包括规划开销、设计开销、实施和测试开销、操作开销和运行维护开销。对物理设计来说，主要考虑操作开销，即为使用户获得及时、准确的数据所需的开销和计算机资源的开销。主要考虑如下几点。

(1) 查询的响应时间。响应时间定义为从查询开始到查询结果显示之间所经历的时间。一个好的应用设计可以减少 CPU 服务时间和 I/O 服务时间。例如，如果有效地使用数据压缩技术，选择好的访问路径和合理安排记录的存储等，都可以减少服务时间。

(2) 更新事务的开销。主要包括修改索引、重写物理块或文件等方面的开销。

(3) 报告生成的开销。主要包括检索、重组、排序和结果显示方面的开销。

(4) 主存存储空间开销。包括程序和数据所占有的空间的开销，可以对缓冲区分配进行适当的控制，以减少空间开销。

(5) 辅助存储空间。可以控制索引块的大小、装载因子、指针选择项和数据冗余度等。

这里主要介绍完整性设计和安全性设计。

1.1.5　完整性设计

完整性是为保证数据库中数据的正确性和相容性，对关系模型提出的某种约束条件或规则。完整性通常包括声明式完整性约束和程序式完整性约束。

1. 声明式完整性约束

声明式完整性约束包括域约束、非空约束、主码约束、唯一性约束、检查约束、参照约束，其中的域约束、主码约束和参照约束是关系模型必须满足的完整性约束条件，完全可以在创建表时声明，也可以在改变表结构时声明。域约束是最基本的约束，表模式中的数据类型即为各个属性具体的域约束，它表示了每个属性允许的取值范围，是单个属性上的约束，如表 1.4 中的“数据类型”一栏。非空约束是对域的进一步裁剪，表示属性上的取值不允许为空，即把域中的空值剔除了，如表 1.4 中的“是否为空”一栏，标识为 N 的为非空约束。主码约束是整个表上的约束，在每个表中，标为主码的属性组取值互不相同，用它唯一地标识了

表中一个元组或者一个记录，如表 1.4 中的“是否主码”一栏，标识为 Y 的为主码。在数据列上“检查约束”需要一个非凡的布尔条件或者将数据列设置成 true，至少一个数据列的值是 null，“检查约束”用于增强表中数据内容的简单的商业规则，用户使用“检查约束”保证数据规则的一致性。“检查约束”是一个表中所有元组必须满足的一个谓词条件，仅限于所定义的表，不能跨表。“唯一性约束”是除了“主码约束”之外的一些属性组合在表中也是唯一的，必须加以标识。域约束、非空约束、主码约束、唯一性约束、检查约束是在单表中的约束。

根据数据字典、域约束、主码约束、非空约束、检查约束和数据库的逻辑模式，表模式进一步的详细信息如表 1.4 所示，表中的属性名一律用英文表示，其表示 E-R 图中的属性见说明一览。表 1.4 中的“检查约束”是示意性的，比较简单，完全可以有更复杂的涉及单表多个属性和系统函数操作的谓词表达式。

表 1.4 数据库中表模式的声明性约束

branch						
字段名称	数据类型	长度	是否为空	是否主码	检查约束	说 明
branch_name	char(50)	50	N	Y		网点名
city	char(20)	20	Y	N		城市
assets	float	8	Y	N	assets≥0	总资产
customer						
customer_ID	char(18)	18	N	Y		身份证号
customer_name	char(10)	10	N	N		姓名
city	char(20)	20	Y	N		城市
street	char(20)	20	Y	N		街道
telephone_num	char(12)	12	Y	N		电话
employee_ID	char(18)	18	Y	N		经纪人
employee						
employee_ID	char(18)	18	N	Y		身份证号
employee_name	char(10)	10	Y	N		姓名
city	char(20)	20	Y	N		城市
street	char(20)	20	Y	N		街道
telephone_num	char(12)	12	Y	N		电话
enter_date	datetime	8	Y	N		入职时间
branch_name	char(50)	50	Y	N		网点名
loan						
loan_number	char(20)	20	N	Y		贷款号
date	datetime	8	N	N		开户日期