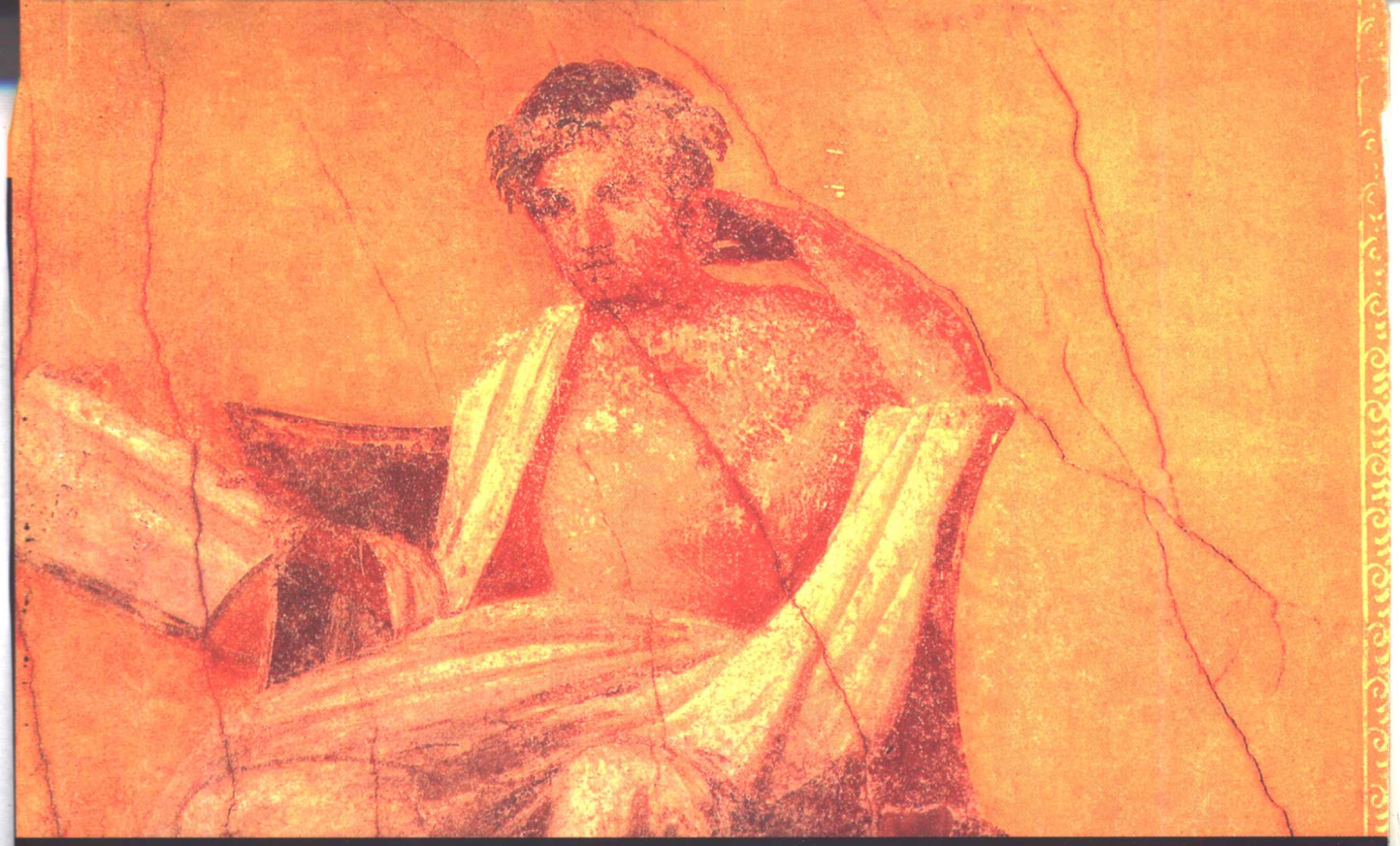

JAVA Database Programming

JAVA数据库编程指南

精通下一代Web数据库技术：

- Java程序如何访问在线数据库
- Java如何与网络数据库技术集成
- Java和JDBC
- Java和基于SQL的数据库引擎

[美] BRIAN JEPSON 著
钱 毅 张祖荫 译

WILEY
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL:http://www.phei.com.cn

JAVA Database Programming

JAVA 数据库编程指南

[美] BRIAN JEPSON 著

钱毅 张祖荫 译

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书是一本讲述 Java 数据库编程技术的书籍，主要介绍了 Java 程序如何在线地访问网上数据库；如何把 Java 语言和网络数据库技术集成在一起；还围绕 JDBC，以 tinySQL、mSQL 和 JDBC-ODBC 网桥三个驱动程序为例，详细地讲述了 Java 语言编写数据库应用的方法，并提供了可用的实用示例程序。

本书适合于 Internet Webmaster、网络管理员、程序员以及广大的 Internet 爱好者。

图书在版编目(CIP)数据

Java 数据库编程指南/(美)杰帕逊(Jepson, B.)著；
钱毅，张祖荫译 . - 北京：电子工业出版社，1998.5
书名原文：JAVA Database Programming
ISBN 7-5053-4695-4

Ⅰ.J… Ⅱ.①杰… ②钱… ③张… Ⅲ.Java 语言-数据库管理系统-程序设计 Ⅳ.TP311.13

中国版本图书馆 CIP 数据核字(98)第 03504 号

JS481/33
04

书　　名：**JAVA 数据库编程指南**
著　　者：[美] BRIAN JEPSON
译　　者：钱毅　张祖荫
责任编辑：邓露林
排版制作：电子工业出版社计算机排版室
印 刷 者：北京市天竺颖华印刷厂
出版发行：电子工业出版社出版、发行　URL：http://www.phei.com.cn
　　北京市海淀区万寿路 173 信箱　邮编 100036　发行部电话 68214070
经　　销：各地新华书店经销
开　　本：787 × 1092 1/16　印张：22.25　字数：570 千字
版　　次：1998 年 10 月第 1 版　1998 年 10 月第 1 次印刷
书　　号：ISBN 7-5053-4695-4/TP·2254
定　　价：31.00 元
著作权合同登记号　图字：01-97-0779

目　录

你认为它有多强大,它就有多强大

"他是全美国最顽固的撒克逊夫人的儿子。"

——Lennon/McCartney, 继续 Bungalow Bill 的故事

这是一个漫长的周末,我在最后的两天中不断往杯中加咖啡,终于机械地完成并校对了80页的程序设计示例。当我感到已经喜欢上它时,伴随它产生了某个故事。现在是星期天的晚上8点,我接连喝了十二盎司的 Cider Jack,这是我第一次在周末喝这么多酒,但可以肯定,这不会是最后一次。从开始考虑示例到现在,这也是我在一周中第一次喝酒。

我并不是为了追求某种醉酒的感觉而喝 Cider Jack 的,事实上,这种滋味肯定不好受。Cider Jack 是一种很好的酒,但我更喜欢喝带有酒花的麦芽酒,喝了这种好酒能获得逼真的感觉,这是因为在 AS220 酒店招待 Brown 和 RISD 的学生时,用的就是同样的酒,而当时的数量难以计算。在正常情况下,周末的大部分时间我都在那儿度过。如果有例外,也是有不同的原因,第一个原因是 AS220 酒店每年都停业一个月,这次碰巧遇上了。另外一个原因要归结于已出现的可能性与事先安排巧合的结果,我再一次离开了我的家庭,在纽约开始了一项新的工作。

纽约是一个充满邪恶的城市,但正如普洛维顿斯的人们所说的,"这儿并不是没有普洛维顿斯好",当然,他们是在普洛维顿斯说这句话。虽然如此,我还是放弃了罗德岛美好的夏天,而在纽约度过了这段难熬的日子,参加了一个十分优秀的 Intranet 项目的工作,并编写了一本关于 Java 的好书,这是我要谈论的最后一个内容。读者手上的这本书是一个夏季的产物,流汗是值得的。我在 Waverly 和 MacDougal 的 Washington Square 饭店开始这项工作,在闪烁不定的荧光灯下,继续进行工作以缩短时间。然而,本书最好的部分是在 TriBeCa 租用的房间中完成的。

在完成了本书的大量工作并不断进行尝试时,给我带来了许多乐趣,最后终于完成了我一直想做的事情:编写 SQL 数据库。本书后面读者会遇到 tinySQL,它是百分之百用 Java 编写的一个 SQL 数据库,我最大的愿望就是在一个大的 applet 上描述一个有雨的周末,一旦我完成了这些神圣的工作,这本书就以不可阻挡的步伐开始进行。

关于 Web 结点

本书所有示例的源程序都能从 http://www.wiley.com/compbooks/中下载,除显示所有示例的源程序外,Web 页也包含与 FAQ(频繁要求回答的问题)的连接和相关的结点,在那儿,读者能下载许多工具,诸如 mSQL、JDBC 部件以及其他感兴趣的内容,但肯定会碰到与这个结点有关的麻烦,也可能是关于这本书或软件的简单问题,读者可以在 bjepson@ids.net 上通过 E-mail 与我联系。

根据 Abbie Hoffman 的 Woodstock Nation 惯例,增加某些乐曲以提高本书的真实性,我认为是一个很好的想法,好的乐曲是重要的。舒适的心情加上优美的音调会使工作出现奇迹,这称为"调整环境"。

·The Beatles,"Revolver"(英文版), Sgt. Pepper's Lonely Hearts Club Band 和 The Beatles(这也很有名,但在正式场合一般不称作"The White Album")

·fIREHOSE,"if'n"

·Fugazi,"Repeater"

·The Grateful Dead,"Anthem of the Sun,"Grateful Dead (第一张唱片)和 Live/Dead

·The Jimi Hendrix Experience," Axis: Bold as Love "和"Electric Ladyland"

·Van Morrison, "Astral Weeks"

·Pixies, "Surfer Rosa"

·Various Artists, A Bitter Pill to Swallow: A Providence Music Sampler

·The Who, "Quadrophenia"

为让读者尽快阅读本书,写到这儿该结束了,但仍要表达我的感谢之意,感谢 John Wiley & Sons 公司的编辑人员:Phil SutherLand, Kathryn Malm、Pam Sobotka 和 Angela Murphy,正是他们的帮助,使这本书得以实现。他们负责整理我那杂乱而又不清晰的言语,并把它们组织成连贯一致、合乎逻辑的书稿。也要感谢 SMT 计算协会人员给予的精神支持,而且在洗衣店与三位蠢猪和一位驿马车夫以及现在臭名昭著的 Lenny Bruce 发生摩擦后,他们在短时间内就提供了保释金。特别感谢 Scott Schoen、Josh Marketos、Shawn Wallace 和我的猫 Oscar,在不断变化的混乱的计划(无论是真实的还是想象的)的进行过程中,他们都提供了好的伴侣和合作。最后,尤其要感谢我的妻子 Pam,她忍耐我"夜不停蹄"地敲击键盘,并且一直在承诺"再给我一点时间..."

第一章　数据库和 Java 程序设计语言

Java 是数据库应用程序的开发工具

"One, Two, (Five) Three, Four"

——The Beatles, "Sgt. Pepper's Lonely Hearts Club Band"(主题曲)

开发工具总是受到人们的青睐。到目前为止,读者可能已听到了关于 Java 众多的赞美之词。Java 是一种强劲、结构合理、安全简单、面向对象、分布式、多线程和动态的开发语言,所有这些特性使 Java 作为一种数据库开发工具,也同样吸引人。本书将尽可能地描述 Java 的每一种特性,并告诉读者如何使 Java 更利于数据库的开发。然而,我们坚信最好是考虑一下开发人员需要什么样的数据库开发工具以及 Java 是如何来简捷实现的。读者不必着急,本书中有关这个问题的讨论不会太多。

易于维护

Java 强大的面向对象的特性使开发那些能组织成数据库应用程序的部件成为可能。软件包的谱系结构能保证十分容易地维护部件的统一和完整。另外,利用 javadoc 实用程序能方便地开发自身编制的代码。

与广泛的数据库服务器具有互联性

Java 问世不久,众多与之相关的数据库互联软件也孕育而生。通过 Java 各种有效的数据库 API(应用程序接口)和使用 JDBC(Sun 公司用 Java 进行数据库开发的 API)的驱动程序,用户能开发与众多数据库服务器产品相匹配的应用程序和 applet。

兼容的 API

使用 JDBC 能开发与数据库无关的 Java applet 和应用程序。许多厂商支持 JDBC,这其中包括那些在 JDBC 以前的时代发布过数据库互联软件包的厂商。使用 JDBC 来建立适用于众多数据库服务器产品的应用程序就轻而易举了。

快速原型化

为了使客户满意,必须尽可能快地实现半工作化原型。尽管用户实际上并不想做什么,但是他们还是喜欢激活按钮或在相关的域内输入信息。Java 面向对象的本质能帮助开发人员方便开发可重用部件以及其他开发人员授权使用的部件。一旦开发人员建立了一整套工具,就能迅速地建立原型。

Java 包含了上述这些特性,因此它作为开发数据库应用程序的工具来说无疑是吸引人的。

如果读者仍对 Java 持有怀疑，本书将提供一些示例，帮助大家学习。

人们是怎样使用 Java 的？虚构的示例

Fiction 先生和我在 Riverrun 共进了一顿丰盛的鸡肉晚餐之后又在 TriBeCa 一同喝咖啡。Fiction 点了龙蒿叶鸡肉，而我却喜欢鸡肉饼，对我来说它们都是美味佳肴。但是我们不得不最后撤掉它们以便留有余地上甜点心——一种美味的用块菌调味的卷心菜。当我们谈到 Java 时，我们设想了下面几个示例：

示例一：Applet Happy

哪种解决方法适用于 Kite 先生，具有一个大型 Web 结点的小型经纪业老板？他获得一个关于库存与商品价格的实时信息系统，它们均保存在 mSQL 数据库中。他还有一个 CGI 图表允许用户按照日期和产品查询某类商品的报价，但他现在想实现一个 applet 以实时地显示一个关于价格变动的图表。通过 mSQL-JDBC，甚至是 mSQL-Java 程序库，Kite 先生能很快地将资源丰富的 applet 组织起来，以满足日益增长的客户的需要。

示例二：Users on the Go

Maxwell Edison 在一家药品供应公司的 MIS 部门工作。尽管他正在上夜校（主修药物），他还是希望开发一个工具以允许他的销售人员在每天工作结束时生成一份报告。由于销售人员通常在不同城市里，因此他们必须远程完成这项工作，有时全在旅店的房间里。Maxwell 决定最好的办法是让销售人员拨号进入 PPP 服务器，然后使用带有嵌入式 Java 的客户应用程序发送他们每日的信息。一旦完成这一过程后，销售人员可以在第二天早上通过拨号利用综合数据修改他们的应用程序。这样做太容易了！每一位销售人员在其桌面运行 Windows 95，但数据库服务器却是运行带有 mSQL Linux 的服务器！

示例三：Big-Time Corporate Intranet

Rita 是大型 Frobozz 500 公司的 IT 成员。她的任务是实现一个单一的数据库应用开发工具，该工具能在 Macintosh、Win 32 以及 Solaris 操作系统上运行。她分别考察了 FoxPro、Galaxy、Delphi 以及 PowerBuilder 等产品，但它们均不能使她满意。她最终选择了 Java，因为 Java 是与体系结构无关、并且易于使用的语言。

使用带有数据库的 Java

为了把 Java 作为由 Java 客户机类，例如 JDBC 支持的众多数据库服务器的客户来使用，读者必须熟悉某些概念。当然，首先必须理解数据库设计的基本知识、结构化查询语言（SQL）以及将数据库表中包含的信息映射到 Java 数据结构和对象中去的方法。下一章将讨论数据库的基本知识和 SQL。这儿先向读者介绍一组动态数据结构，以后的章节将讲述基于 Java 数据库的有关信息。

动态数据结构

在使用数据库时,用户会发现必须使用某些可用的动态结构。Hashtable(哈希表)和 Vector 类允许用户很方便地用 Java 来描述行和列。

Java 程序 Dynamic.java 将 java.util.Vector 对象说明成行。Vector 只是一个对象的简单的可增长数组。在这个示例中,每一行包含一个 Hashtable。一个 Hashtable 允许用户按关键字存储数据,如同在 Perl 的综合数组中一样。用户能够使用 put( )方法按关键字增加一项数据,并用 get( )方法按关键字读取数据。关键字可以是任何对象,但在目前情况下,我们使用 String 对象。

一个简单的 Hashtable 示例是一个晚餐聚会上每个人所用饮料的列表。下面这段代码实现这一功能:

```
import java.util.*;

public class Drinks {
  public static void main(String argv[]) {
    Hashtable beverages = new Hashtable();
    beverages.put("Vera", "Ginger Ale");
    beverages.put("Chuck", "Rob Roy");
    beverages.put("Dave", "Sloe Gin Fizz");
  }
}
```

一旦实现了这一代码,用户就能检索到 Dave 正在喝什么:

```
String DaveDrinks = (String) beverages.get("Dave");
System.out.println(DaveDrinks);
```

因为哈希表(Hashtable)能包含各种类型的对象,所以当用户将它赋给 String 类型的对象时,他必须将 get( )返回的对象显式地转换成 String。

下面是 Dynamic.java。因为还未向读者介绍有关数据库的内容,所以我们先将一些虚假的数据放在数据结构中。该示例提供给读者关于用 Java 数据结构来表示数据库表而可能采用的方式的一个想法。用户能用 rows.size( )获得行数,而用 rows.elementAt( )可检索某一行的数据。由于 Vector 中的每一个对象是一个 Hashtable,用户能将 Hashtable 赋给一个临时对象,例如 foo,然后根据每一个关键字调用 foo.get( )来检索相应行的数据。

```
import java.util.*;

public class Dynamic {

  public static void main(String argv[]) {

    // a new Vector object to hold all the rows
```

```
        Vector rows = new Vector( );

        // populate the data structure with some bogus
        // values that might appear in a table

        popData( rows, "Brian", "Jepson");
        popData( rows, "Mr.", "Kite");
        popData( rows, "Mr.", "Mustard");
        popData( rows, "Japhy", "Ryder");

        // process each row in the "table"

        for (int i = 0; i < rows.size( ); i++) {

        // get the Hashtable that is contained in each row

        Hashtable foo = (Hashtable) rows.elementAt(i);

        // print out the row number - but add one to it
        // since the index offset is zero, but most
        // people are used to seeing records/rows start
        // at one.

        System.out.println("Row" + (i + 1));

        // since foo is the Hashtable for the current row,
        // you can get the column by invoking the get ( )
        // method with the name of the column you want.

        System.out.println(" first_name = " + foo.get("first_name"));
        System.out.println(" last_name = " + foo.get("last_name"));

      }

      // exit cleanly

      System.exit(0);

    }

    // a convenience method to add items to the "table"

    public static void popData (Vector rows, String first, String last) {
```

```
        // create a new Hashtable

        Hashtable columns = new Hashtable( );

        // add the data to the new Hashtable

        columns.put("first_name", first);
        columns.put("last_name", last);

        // add the Hashtable to the Vector

        rows.addElement(columns);

        }

    }
```

该程序产生的输出如下：

```
Row 1
  first_name = Brian
  last_name = Jepson
Row 2
  first_name = Mr.
  last_name = Kite
Row 3
  first_name = Mr.
  last_name = Mustard
Row 4
  first_name = Japhy
  last_name = Ryder
```

本书包含了不少示例，特别是调用了许多 Hashtable 和 Vector。本书中有许多工作源代码；这是一本十分便利的书，因此用户必须在计算机中阅读它。我们试图向读者提供两级文档：代码中的综合文档；以及围绕它在本书正文中所作的更加详细的论述。我们希望这样做能使读者清楚地理解那些难懂的内容。

第二章　数据库和数据库设计概述

"I look at the floor, and I see it needs sweeping…"

——The Beatles, "While My Guitar Gently Weeps"

当用户将 Java 和数据库进行集成的时候，他不仅仅要理解如何与数据库进行连接，而且还必须理解执行命令。我们的目标是实现一个真正灵活的系统，该系统能不断满足用户的需求，而对于那些维护人员来说，它也是易于理解的。用户会发现遵循某些规则是有益的，因为自从数学家 E·F·Codd 在 1970 年发表关于关系数据库管理系统的数学模型以来，这些规则就一直指导着数据库开发人员。他的基本理论发表在一篇题为"大型分布式数据银行的关系型数据模型"的论文中(Communications of the ACM 13, NO. 6, June 1970)，读者可在 Web 网的 http://www.acm.org/classics/nov95/上找到它。

表和数据库设计

表是行的集合。一行分为一列或多列。表的集合称为数据库。表很像人们在某一天里记下的列表，在一个风和日丽的夏天，读者有可能感到特别烦闷，那么就会列出房间里所有地毯和衣服上的油污物。其列表如下：

Item	Type	Merlot	Ketchup	Jelly	Coffee	Unknown
Handknit Sweater	Clothing	Yes	No	Yes	Yes	No
Couch	Furniture	No	Yes	No	Yes	Yes
Teddy Bear	Faithful Companion	No	No	Yes	Yes	No
Silk Tie	Clothing	No	Yes	Yes	No	No

屋里肯定还隐藏着许多污物，这儿我们只是给出一个关于表的概念。尽管读者会试图去列出其生活中的许多更重要的污物，但关系型模型认为用这种方式组织起来的表运用起来十分困难。当设计映射到该列表结构的数据库表时，如果有人在沙发上溅上了一点新的、有意思的物质时就会遇上麻烦；此时必须增加一列，就要修改表结构。

关系型模型和第一标准格式

Codd 的论文建立了数据标准化的关键字准则。数据标准化是优化表中数据的一个优化过程。标准化过程中的每一步都要将数据转换为一个不同的标准格式。正如采用越好的马掌就会越快一样，重要的是尽可能地标准化数据。目前存在五种标准格式；本章我们只讨论其中的三种，对于大多数据库开发而言，这已经足够了，对于使用第三种以上标准格式的人，他们通常有三只眼睛，能洞察未来 15 秒中的事情，这一点实际上是惹人烦的。

对于遵循第一种标准格式的数据库设计来说，必须删除重复列。在前面关于污物的列表

中,有一列是表项,一列是表项的类型,另一列是每种污物的类型。污物的集合可以进行优化(没有提及高科技的清洁服务社)。用一组重复的列来存储数据效率太低,对每一列而言,浪费了存储空间;对每一表项而言,为每一种可能的污物准备一列,而不管该表项的结果如何。另外,如果一个酒鬼(或一个恶作剧的)客人倒上了别的东西,如菊芋(mmm…是菊芋),那么数据库的结构在物理上要进行修改。这是一个大问题:如果为一个系统制定了完备的规格说明,那么就不应该对表结构进行修改,除非产生了相当新颖的需求。不同类型的污物不能满足"相当新颖的需求"这一要求。该数据库毕竟是关于污物的,因此数据库应该能处理它们。

为了使该表遵循第一标准格式,它必须分成两个表。创建一个新表,称为 STAIN。该表包含污物的名称和两个新列,第一个是污物 ID,第二个是表项 ID,称为 ITEM 的源表丢失了所有的污物列,但它获得一个新列:表项 ID,表项 ID 将污物列表与表项连接起来。ITEM 表的"新内容"如下:

Item	Type	Item ID
Handknit Sweater	Clothing	1
Couch	Furniture	2
Teddy Bear	Faithful Companion	3
Silk Tie	Clothing	4

STAIN 表如下所示:

Stain Name	Stain ID	Item ID
Merlot	1	1
Jelly	2	1
Coffee	3	1
Ketchup	4	2
Coffee	3	2
Unknown	5	2
Jelly	2	3
Coffee	3	3
Ketchup	4	4
Jelly	2	4

以这种方式组织的表的可读性不强,但它却使程序设计人员易于增加和操作数据。增加的新列提示了标准化数据设计中采用的某些方法。在一个设计完美的图表(图表是以某种形式相连的表的集合)中的每一个表都应当有一个列作为主键,主键是唯一的标识符,它可以是一列也可以是列组,它们的值对于每一行来说都是唯一的。ITEM 表以 Item ID 作为主键,该表中没有两行具有相同的 Item ID。STAIN 表中的主键包括两列:Item ID + Stain ID。没有两行可以有相同的 Item ID + Stain ID。污物图表不再包含有令人讨厌的非标准化表,它现在包含两个

表,它们都以第一标准格式存在。

第二标准格式

STAIN 表的主键(Item ID + Stain ID)满足主键定义的要求,每一行中这两列的值都不相同。读者也许注意到 Stain Name 有重复,它是一个非独立的值,它只依赖于主键(Stain ID)多值化的其中一个部件。第二标准格式下的应用程序将消除这种情况,也就是说,它消除冗余数据。现在只要在定义污物和表项关系的地方就会出现 Stain Name。如果一个污物名称的拼写有所不同,就会引起报表的不规则,而用户也得不到正确的污物总数。更糟糕的是,如果标识了"不知道"的污物,用户就必须修改 STAIN 表中的所有记录,这样做无异于将它当作交叉索引表来处理。

为了使图表中的表遵循第二标准格式,用户必须另外创建一个表。我们将它称为 STAIN XREF,它只包含 Stain ID 和 Item ID。STAIN 表中现在只包含 Stain Name 和 Stain ID。STAIN 表如下所示:

Stain Name	Stain ID
Merlot	1
Jelly	2
Coffee	3
Ketchup	4
Unknown	5

STAIN XREF 表如下所示:

Stain ID	Stain ID
1	1
2	1
3	1
4	2
3	2
5	2
2	3
3	3
4	4
2	4

第三标准格式

第三标准格式与第二标准格式存在某种相似之处,但它更精确一些。它试图删除那些实际上独立于键值的列。在第二标准格式的示例中,我们删除的列部分依赖于键值。回顾一下 ITEM 表,请读者注意它有三列:Item Name、Type 和 Item ID。其中只有 Item Name 依赖于 Item