第5章

数据库设计与治理

本章学习目标

- 了解数据库的概念
- 了解数据模型
- 了解数据库治理
- 了解数据字典
- 掌握数据库设计的标准规范
- 了解图谱数据库

本章先向读者介绍数据库的概念，再介绍数据模型，接着介绍数据库治理，最后介绍数据字典、数据库设计的标准规范以及图谱数据库。

5.1 数据库概述

5.1.1 数据库介绍

在企业制造和生产过程中，需要把各种材料和成品按照一定的规格分门别类地存储在仓库中。计算机系统对现代化数据的处理过程有相似的地方，即把各种数据先分类，再将结果分别存放在数据仓库中，也就是数据库中。数据库技术是计算机领域中的重要技术之一，它将各种数据按一定的规律存放，以便于用户查询和处理。

例如，把学校的学生、课程、上课教师及学生成绩等数据有序地组织并存放在计算机中，就可以构成一个数据库。学生可以登录学校教务系统网站查询成绩，教师也可以登录该网站查询上课情况。因此，数据库是由一些彼此关联的数据集合构成的，并以一定的组织形式存放在计算机中。

5.1.2 数据库管理系统

数据库管理系统(Database Management System，DBMS)是一种操作和管理数据库的软件，它是数据库的核心，主要用于创建、使用和维护数据库。在 DBMS 中普通用户可以登录和

查询数据库，管理员可以建立和修改数据库等。

数据库管理系统主要包含以下功能：

（1）数据定义。DBMS 提供了各种数据定义语言，用户可以定义数据库中的各种数据对象。

（2）数据操纵。DBMS 提供了大量的数据操纵语言，用户可以对数据库中的数据表进行各种操作，例如对数据表进行创建、删除、插入、修改以及查询等操作。

（3）数据管理与安全保护。在 DBMS 中数据库的建立、运行和维护由数据库管理系统统一管理，以保证数据的完整性。此外，为了确保数据库的安全，只有被赋予权限的用户才可以访问数据库的相关数据。图 5-1 显示了数据库、数据库应用系统与数据库管理系统的关系。

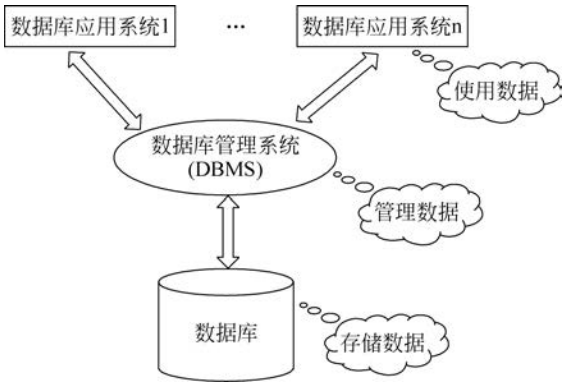


图 5-1　数据库、数据库应用系统与数据库管理系统的关系

从图 5-1 可以看出，数据库、数据库应用系统与数据库管理系统共同构成了数据库系统。其中，数据库管理系统是整个数据库系统的核心，编程人员可以通过数据库管理系统来操纵整个数据库系统。

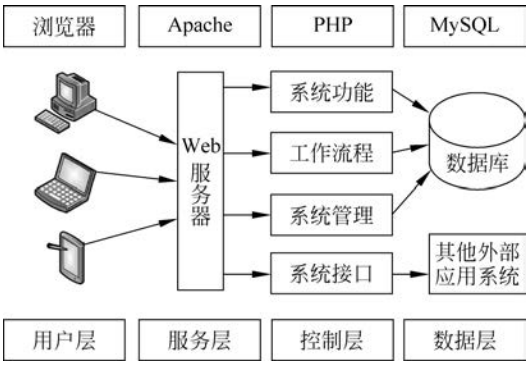


图 5-2　使用 MySQL 存储数据

按照目前流行的分类方式，数据库可根据数据结构的匹配关系分为关系型数据库和非关系型数据库。其中，关系型数据库是数据库应用的主流，许多数据库管理系统的数据模型都是基于关系数据模型开发的，而非关系型数据库在特定的场景下可以发挥出难以想象的高效率和高性能，是对传统关系型数据库的有效补充。目前在市场上比较流行的 DBMS 有 SQL Server、Oracle、MySQL、Sybase 以及 Access 等。图 5-2 所示为使用 MySQL 存储数据。

5.1.3　数据库系统的结构

数据库系统在总体结构上一般体现为三级模式，分别是模式、外模式和内模式。

（1）模式。模式又叫概念模式或逻辑模式，它是数据库中全体数据的逻辑结构和特征的描述。模式位于三级结构的中间层，它以某一种数据模型为基础，表示了数据库的整体数据。在定义模式时，不仅要考虑数据的逻辑结构，例如数据记录的组成，数据项的名称、类型、长度等，还要考虑与数据有关的安全性、完整性等要求，并定义数据之间的各种关系。值得注意的是，一个数据库只有一个模式。

（2）外模式。外模式又叫子模式或用户模式，它是一个或几个特定用户所使用的数据集

合(外部模型),是用户与数据库系统的接口,是模式的逻辑子集。外模式面向具体的应用程序,定义在逻辑模式之上,但独立于存储模式和存储设备。在设计外模式时应充分考虑应用的扩充性。当应用需求发生较大的变化,相应的外模式不能满足其视图要求时,该外模式就必须做相应改动。值得注意的是,一个数据库可以有多个外模式。

(3) 内模式。内模式又叫存储模式,它是数据在数据库系统中的内部表示,同时也是数据库最低一级的逻辑描述。内模式描述了数据在存储介质上的存储方式和物理结构,对应实际存储在外存储介质上的数据库,主要包含记录的存储方式、索引的组织方式、数据是否压缩存储、数据是否加密、数据存储记录结构的规定等。值得注意的是,一个数据库只有一个内模式。

图 5-3 显示了数据库系统的模式结构。

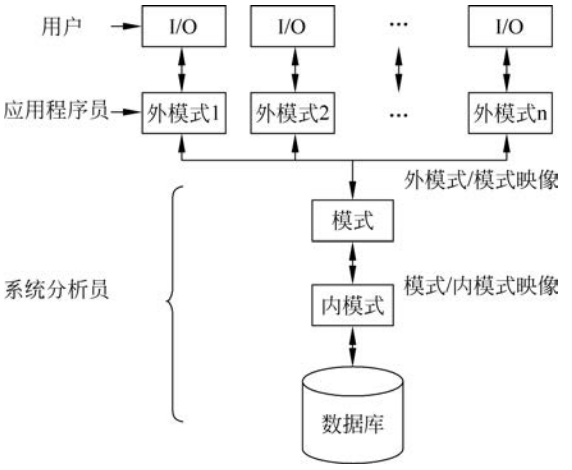


图 5-3 数据库系统的模式结构

5.1.4 数据模型

模型是现实世界中某些特征的模拟和抽象,模型一般可以分为实物模型与抽象模型。实物模型通常是客观事物的外观描述或功能描述,例如汽车模型、飞机模型、轮船模型、火箭模型等。抽象模型通常是客观事物的内在本质特征,例如模拟模型、数学模型、图示模型等。图 5-4 所示为模型的分类。

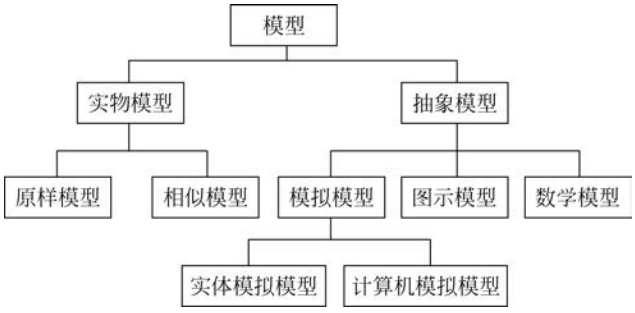


图 5-4 模型的分类

通常在建立数据库模型时会涉及 3 种具体的数据模型,分别是概念模型、逻辑模型和物理模型。

1) 概念模型

概念模型是现实世界到数据世界的一个过渡层次,它是数据库设计人员进行数据库设计的有力武器,也是数据库设计人员与用户交流的语言,因此概念模型具有简单、易懂的特点。

概念模型的相关概念如下：

(1) 实体。实体是客观存在并相互区别的事物与事物之间的联系，例如一棵树、一个水杯、一本图书等都是实体。

(2) 实体集。实体集是指同类实体的集合，例如全体学生就是一个实体集。

(3) 属性。属性是指实体所具有的某一特性，例如学生的学号、姓名、性别、年龄、籍贯等都是学生实体的属性。

(4) 联系。联系是指实体与实体之间以及实体与组成它的各个属性间的关系。在具体的表示中，一般常用 E-R 图来描述实体集及其之间的联系，即用矩形框表示实体，用圆角矩形表示属性，用菱形表示实体与实体之间的联系，用线段连接实体集与属性。

(5) 关键字。在数据库的一个表或一个文件中可能存储着很多记录，为了能唯一地标识一个记录，必须在一个记录的各个数据项中确定出一个或几个数据项，它们的集合称为关键字(keyword)。关键字是唯一标识实体的属性集，例如学生的学号在数据库设计中就是学生实体的关键字。关键字可以同时包含多个属性，也可以包含一个属性。

(6) E-R 图。E-R 图也称为实体-联系图，它提供了表示实体类型、属性和联系的方法，用来描述现实世界的概念模型。在 E-R 图中用矩形表示实体名，矩形框内写出实体名；用椭圆表示实体的属性名，并用无向边将其与相应的实体连接起来；用菱形表示实体之间的联系，在菱形框内写出联系名，并用无向边分别与有关实体连接起来，同时在无向边旁标上联系的类型(1∶1、1∶n 或 m∶n)。图 5-5 所示为 E-R 图中不同图形的含义，图 5-6 所示为用户与角色之间的 E-R 图。

图 5-5　E-R 图中不同图形的含义

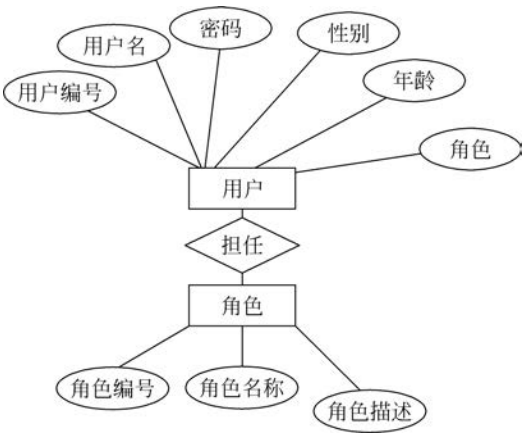


图 5-6　用户与角色之间的 E-R 图

2) 逻辑模型

从定义上讲，逻辑模型是以概念模型为基础，对概念模型的进一步细化、分解。逻辑模型通过实体和实体之间的关系描述业务的需求和系统实现的技术领域，是业务需求人员和技术人员沟通的桥梁和平台。

逻辑模型的设计是数据仓库实施中最重要的一步，因为它直接反映了业务部门的实际需求和业务规则，同时对物理模型的设计和实现具有指导作用。它的特点是通过实体和实体之间的关系勾勒出整个企业的数据蓝图和规划。逻辑模型一般遵循第三范式，与概念模型不同，它主要关注细节性的业务规则，同时需要解决每个主题域包含哪些概念范畴及跨主题域的继承和共享问题。图 5-7 所示为逻辑模型。

值得注意的是，逻辑数据建模不仅会影响数据库设计的方向，还间接影响最终数据库的性能和管理。如果在实现逻辑数据模型时投入得足够多，那么在进行物理数据模型设计时就有许多可以选择的方法。

3) 物理模型

物理模型是对真实数据库的描述，是逻辑模型的延伸。物理模型能够针对逻辑模型所说

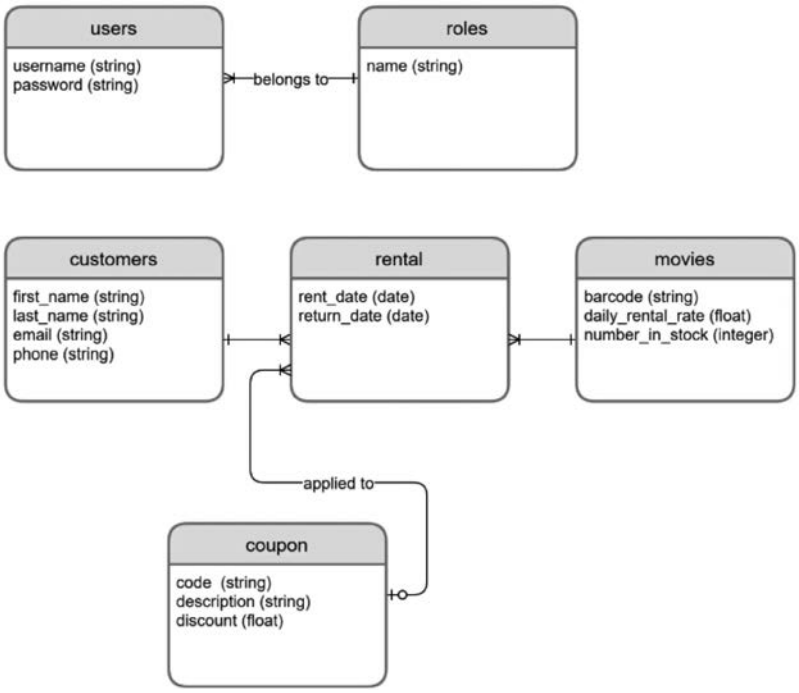


图 5-7 逻辑模型

的内容在具体的物理介质上实现出来。

具体来说,物理模型是在逻辑数据模型的基础上考虑各种具体的技术实现因素进行数据库体系结构设计,真正实现在数据库中存放数据。物理数据模型的内容包括确定所有的表和列、定义外键用于确定表之间的关系、基于用户的需求可能要进行反范式化等内容。在物理实现上的考虑可能会导致物理数据模型和逻辑数据模型有较大的不同。图 5-8 所示为物理模型。

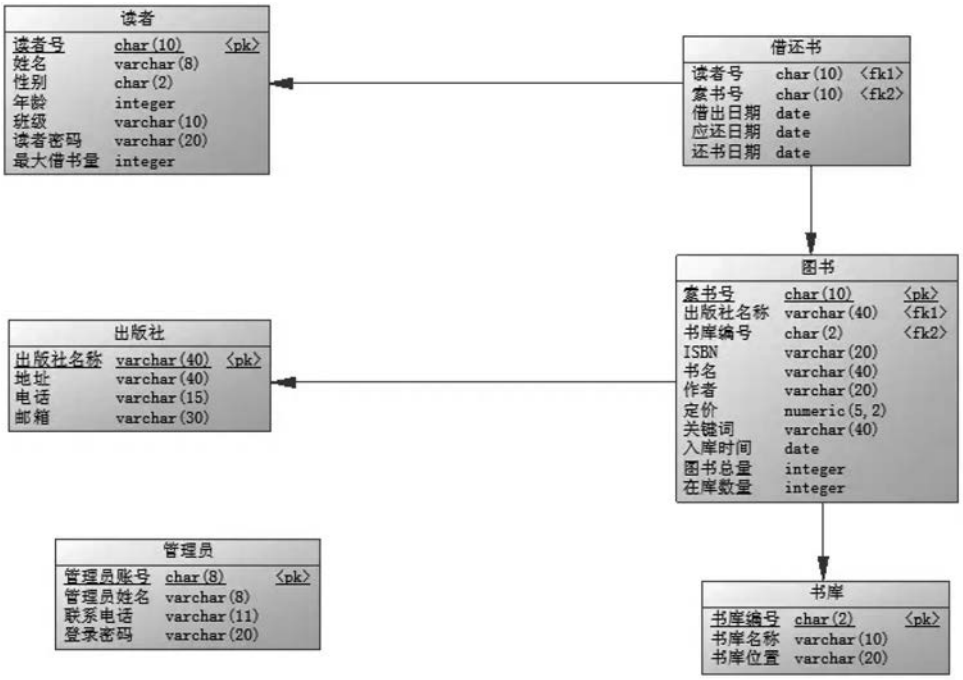


图 5-8 物理模型

在物理模型确定以后，就可以进一步确定数据的存放位置和存储空间的分配，最后生成定义数据库的 SQL 命令。值得注意的是，在逻辑结构中人们无须顾及具体的数据库实现，只要关注业务含义即可。一旦到了逻辑模型向物理模型转化的阶段，进行数据库的选择就是一件不可以忽略的事情。进行物理数据库设计要考虑在最终数据库平台上实现的具体部署模型，这部分设计将严重依赖于目标 RDBMS 的功能和实现手段，而不同的数据库平台会有不同的解决方案。

概念模型、逻辑模型、物理模型的区别见表 5-1。

表 5-1　概念模型、逻辑模型、物理模型的区别

模型	概念模型	逻辑模型	物理模型
属性	不需要完整定义实体的属性	完整定义实体的属性	确定字段名、长度、数据类型、是否可以为空、初始值等
主键	无须确定主键	无须确定主键	确定主键

5.1.5　关系数据库的设计流程

关系数据库是一种基于关系模型的数据库，关系模型折射现实世界中的实体关系，将现实世界中各种实体及实体之间的关系通过关系模型表达出来。关系数据库的设计常包含需求分析、概念结构设计、逻辑结构设计、物理结构设计、编码设计和运行维护 6 个步骤。

（1）需求分析。需求分析阶段的工作主要是充分进行调查研究，了解用户的需求，了解系统的运行环境，并收集数据，为以后的步骤做准备。

（2）概念结构设计。概念结构是整个系统的信息结构，概念结构设计阶段的主要工作是对收集到的数据进行分析，确定实体、实体属性及实体之间的联系，并画出实体-联系图。

（3）逻辑结构设计。逻辑结构设计阶段的主要工作是将概念结构设计的实体图转换为与 DBMS 相对应的数据模型，并对该模型进行优化。

（4）物理结构设计。物理结构设计阶段的主要工作是为设计好的逻辑数据模型选取一个较合适的物理结构，并对该物理结构进行评估和优化。

（5）编码设计。编码设计阶段的主要工作是利用 DBMS 的数据定义语言将数据库描述与实现出来，反复调试所编写的程序并保证能够运行。

（6）运行维护。运行维护阶段的主要工作是通过输入大量数据测试该数据库系统的各项性能，以便发现问题，改正问题。

5.2　数据库治理

5.2.1　数据库治理概述

在对关系数据库进行治理的时候有两个关键因素，即数据字典和数据库设计标准。其中，数据字典存储有关数据的来源、说明、用途、格式和与其他数据的关系等信息，是数据库治理的基础；数据库设计标准则对数据库命名、数据表命名、数据表中的字段命名、数据表索引以及各种数据操作和数据约束等做了规范。表 5-2 所示为数据字典，图 5-9 所示为数据关系。

表 5-2 数据字典

数据字典名称	说 明
dba_tablespaces	关于表空间的信息
dba_ts_quotas	所有用户表空间限额
dba_free_space	所有表空间中的自由分区
dba_segments	描述数据库中所有段的存储空间
dba_extents	数据库中所有分区的信息
dba_tables	数据库中所有数据表的描述
dba_tab_columns	所有表、视图以及簇的列
dba_views	数据库中所有视图的信息
dba_synonyms	关于同义词的信息查询
dba_sequences	所有用户序列信息
dba_constraints	所有用户表的约束信息
dba_indexs	关于数据库中所有索引的描述
dba_ind_columns	在所有表及簇上压缩索引的列
dba_triggers	所有用户的触发器信息
dba_source	所有用户存储过程信息

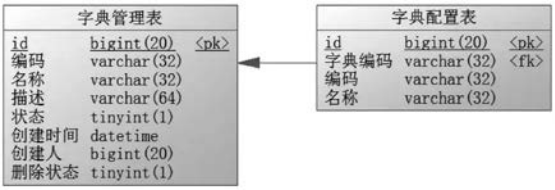


图 5-9 数据关系

5.2.2 数据字典设计

1. 数据字典概述

数据字典(Data Dictionary)是一种用户可以访问的记录数据库和应用程序元数据的目录,是对于数据模型中的数据对象或者项目的描述的集合。数据字典存放数据库所用的有关信息,在数据库设计初期将数据库中各类数据的描述集合在一起,用于在开发、维护或者其他需要的时候使用。

数据字典通常包括数据项、数据结构、数据流、数据存储以及处理过程 5 个部分。数据字典通过对数据项和数据结构的定义来描述数据流和数据存储的逻辑内容。

(1) 数据项。数据项是数据的最小组成单位,是不可再分的数据单位,若干个数据项可以组成一个数据结构。

(2) 数据结构。数据结构反映了数据之间的组合关系,一个数据结构可以由若干个数据项组成,也可以由若干个数据结构组成,或由若干个数据项和数据结构混合组成。

(3) 数据流。数据流是数据结构在系统内传输的路径。

(4) 数据存储。数据存储是数据结构停留或保存的地方,也是数据流的来源和去向之一。

(5) 处理过程。处理过程是具体处理逻辑,一般用判定表或判定树来描述。

2. 数据字典的作用

人们可以把数据字典看作指南,它为数据库提供了“路线图”,而不是“原始数据”。换句话

说，数据字典通常是指数据库中数据定义的一种记录，类似一个数据库的数据结构，但其内容要比数据库的数据结构描述丰富得多。

数据字典有以下几个主要作用：

(1) 提高开发效率，降低研制成本。数据字典是数据库开发者、数据监管人和用户之间的共同约定，是系统说明书的一个重要组成部分。一个统一的数据字典有助于开发者建立数据模型以及程序和数据库之间的数据转换接口，为规范化设计和实施数据管理系统铺平了道路。

(2)促进数据共享，提高数据的使用效率。通过数据字典，用户可以方便地知道每项数据的意义，了解数据的来源和使用方法，从而帮助用户迅速地找到所需的信息，并按照正确的方法使用数据。

(3)控制数据的使用。在某些特定的场合，可以通过对数据字典的控制达到控制数据使用的目的。

3. 数据字典的设计实例

某开发者设计了一张表 user，该表中包含一个属性“证件”，其属性值为“身份证”，并且该表中已经保存了大量数据，如图 5-10 所示。

如果某一天客户对属性“身份证”不满意，需要将其更换为“居民身份证”，作为这个项目开发人员，要把代码里和数据库中所有的“身份证”都变成“居民身份证”，这是一个很麻烦的操作。

<user表>

id	name	证件	地址
1	李四	身份证	河北
2	张三	签证	北京

图 5-10 user 表中的属性

如果事先设计了数据字典就可以很轻松地完成这一操作。在数据字典设计中可以为表 user 中的“证件”新建一个属性表——“证件表”，将属性值(证件类型)和主体(证件)分离，而主体表(表 user)只保存属性的代码，这样就可以达到快速修改表 user 中属性的目的，如图 5-11 所示。

<user表>

id	name	证件	地址
1	李四	1	河北
2	张三	3	北京
3	王六	2	广东
4	刘洋	1	天津

<证件表>

证件id	证件内容
1	身份证
2	居住证
3	签证

图 5-11 快速修改 user 表中的属性

从该例可以看出，数据字典是一种通用的程序设计方法。可以认为，不论什么程序都是为了处理一定的主体，这里的主体可能是人员、商品、网页、接口、数据库表甚至是需求分析等。当主体有很多属性，每种属性有很多取值，而且属性的数量和属性取值的数量是不断变化的，特别是当这些数量的变化很快时，就应该考虑引入数据字典的设计方法。数据字典的实际应用如图 5-12 所示。

图 5-12 数据字典

4. 数据字典的应用

设计商品数据库，数据项设计见表 5-3、数据结构设计见表 5-4、数据流设计见表 5-5、数据存储设计见表 5-6、处理过程设计见表 5-7。

表 5-3 数据项

数据项名	数据项含义	别　　名	数据类型	取值范围	取值含义
spbh	唯一标识每一个商品	商品编号	char(15)	0～999 999 999 999 9	前 3 位是厂商所在国家的国际代码，它和 4～7 位一起构成厂商识别代码，即厂商的注册号，8～12 位是商品项目代码，最后一位是校验码
spmc	标识商品的名称	商品名称	char(30)		
jj	标识商品的进价	进价	money		
gysbh	唯一标识商品的供应商	供应商编号	char(8)		前 3 位是厂商所在国家的国际代码，它和 4～7 位一起构成厂商识别代码，即厂商的注册号
kcsl	标识库存的数量	库存数量	char(8)		
gysmc		供应商名称	char(30)		
dz		地址	char(30)		
lxfs		联系方式	char(15)		
lxr		联系人	char(16)		
ygbh		员工编号	char(8)		前两位是部门号，3～10 位是加入日期，11～13 位是顺序编号
ygxm		员工姓名	char(8)		
zw		职务	char(8)		
bmbh		部门编号	char(4)		顺序编号
dhdbh		订货单编号	char(8)		
dgrq		订购日期	smalldatetime		
jhrq		交货日期	smalldatetime		
rkdh		入库单号	char(8)		
rksl		入库数量	char(8)		
rkrq		入库日期	smalldatetime		
ckdh		出库单号	char(8)		
cksl		出库数量	char(8)		
ckrq		出库日期	smalldatetime		
gkbh		顾客编号	char(10)		
sgsp		所购商品	char(60)		

续表

数据项名	数据项含义	别　名	数据类型	取值范围	取值含义
zj		总价	numeric(5,1)		
rq		日期	smalldatetime		

表 5-4　数据结构

数据结构名	含义说明	组　成
商品	商品管理子系统的主体数据结构，定义了商品的有关信息	商品编号、商品名称、供应商、单价
供应商	进货管理子系统的主体数据结构，定义了供应商的有关信息	供应商编号、供应商名称、联系方式、地址、联系人
员工	员工子系统的主体数据，定义了员工的有关信息	员工编号、员工姓名、年龄、职务、所属部门
顾客	销售子系统的主体数据，定义了顾客的有关信息	顾客编号、所购商品
订货单	订货子系统的主体数据	订货单编号、供应商编号、采购员工号、订购日期、交货日期
入库单	入库子系统的主体数据	入库单号、商品编号、入库数量、入库日期
出库单	出库子系统的主体数据	出库单号、商品编号、出库数量、出库日期
销售单	销售子系统的主体数据	顾客编号、商品编号、总价、日期

表 5-5　数据流

数据流名	说　明	数据流来源	数据流去向	组　成	平均流量	高峰期流量
入库单	供应商供应的货物	供应商供货处理	入库单存储	入库单号、商品编号、供应商编号、入库数量、入库日期	每天 20 个	每天 100 个
出库单	库中商品出库	出库处理	出库单存储	出库单号、商品编号、出库数量、出库日期	每天 20 个	每天 100 个
销售单	商店中商品出售	销售处理	销售单存储	顾客编号、商品编号、总价、日期	每天 20 个	每天 100 个

表 5-6　数据存储

数据存储名	说　明	流入的数据流	流出的数据流	组　成	数据量	存取方式
入库	商品入库	入库单	入库单	入库单	1 000 000 个记录	随机存取
出库	商品出库	商品信息、出库信息	出库单	出库单	1 000 000 个记录	随机存取
销售	商品销售	商品信息、销售信息	销售单	销售单	1 000 000 个记录	随机存取

表 5-7 处理过程

处理过程名	说 明	输入数据流	输出数据流	处 理
入库	商品存入仓库	入库单	入库单	记录入库单号、商品编号、入库数量、入库日期
出库	商品从仓库中取出	出库单	出库单	记录出库单号、商品编号、出库数量、出库日期
销售	商品从商店中售出	销售单	销售单	记录顾客编号、商品编号、总价、日期

5.2.3 数据库设计

1. 数据库设计的标准规范

在数据库设计中数据模型标准规范见表 5-8，通用命名规范见表 5-9。

表 5-8 数据模型标准规范

规则	描 述
规则 1	标注出表与表间的关系，例如一对一、一对多等
规则 2	在数据库设计文档中，表与字段都要增加备注，以表明表与字段的具体含义及用处
规则 3	若无特别说明，每个表的索引不得超过 5 个
规则 4	基于多表关联的视图，必须在字段名前指定表别名
规则 5	在存储过程中必须有异常捕获代码
规则 6	如果在存储过程中使用了游标，则当存储过程正常或者异常退出时必须关闭所有打开的游标
规则 7	如果在存储过程中有更新，必须在异常捕获代码中做回退操作
规则 8	在函数中如果进行了事务处理，必须有异常捕获代码
规则 9	在函数中如果使用了游标，则在函数正常或者异常退出时必须关闭所有打开的游标
规则 10	在函数中如果对数据进行了更新操作，必须在异常捕获代码中做回退操作
规则 11	对于需要同步到数据仓库的表，原则上必须包含同步频率以及同步机制
规则 12	频繁出现在 where 子句里的字段建议建立索引
规则 13	用来和其他表关联的字段建议建立索引
规则 14	在 where 子句里作为函数参数的字段不能创建索引
规则 15	在建立索引的时候，建议考虑 select 和 insert、update、delete 的平衡
规则 16	一般建议查询数据量在 10%以下时使用索引
规则 17	选择性更高的字段放在组合字段索引的前导字段
规则 18	如果字段的查询频率相同，则把表中数据的排列顺序所依据的字段放在前面
规则 19	函数尽量只实现复杂的计算功能，不对数据库进行更新操作

表 5-9 通用命名规范

规则	描 述
规则 1	任何数据库对象的命名，一般情况下不建议使用汉字
规则 2	同类业务的表以相同的英文开头
规则 3	命名不得使用数据库保留字
规则 4	命名应使用富有意义的英文，一般情况下不建议使用拼音命名
规则 5	临时表以“tmp_”开头，后接功能描述
规则 6	同种用途的字段，在同一个业务的所有表中应有同样的字段类型和字段长度，并尽量保持一致的字段命名
规则 7	对于数据不定长的字段，字段类型定义为 varchar 类型
规则 8	在数据库中同一信息的字段名一样，字段类型也一样

续表

规则	描　　述
规则 9	函数以“func_”开头，后接函数的功能
规则 10	在表中引用其他表的主键时，用部分表名加 Id

2. 数据库设计的工作流程

数据库设计的工作流程根据先后次序分为数据库设计、数据库安装、数据采集与历史数据迁移、数据库测试与调整四方面。

1）数据库设计

数据库设计工作分为数据库的概念设计、逻辑设计、物理设计 3 个层次。

（1）概念设计。概念设计是将需求分析得到的各专业应用领域的数据描述抽象为信息结构的过程，需要在充分理解业务流程和应用环境的基础上对这些业务信息用 E-R 图的形式设计出来，同时要考虑系统正常运行所必需的用户管理、运行平台等功能需要用到的支撑表。概念设计是整个数据库设计的关键所在。

（2）逻辑设计。在逻辑设计阶段将概念设计的 E-R 图转换为具体的数据库产品支持的数据模型，例如关系模型，需要在进行数据库管理系统选择的基础上针对选定的 DBMS 将字段的类型、名称等做调整，要求数据库逻辑设计的范式至少满足 3NF。

（3）物理设计。在物理设计阶段将为逻辑设计模型选取一个最适合应用环境的物理结构（包括存储结构和存取方法），并根据 DBMS 的特点和处理的需要，针对特定数据库管理系统的优化，进行物理存储安排，设计索引，形成数据库内模式。

2）数据库安装

将数据库物理设计模型应用特定 DBMS 工具生成数据库 SQL 脚本或使用嵌入式 SQL 语言工具生成数据库脚本，并在生产环境下运行脚本生成大数据平台系统相关数据库物理结构。

3）数据采集与历史数据迁移

（1）基础数据采集。组织内部负责采集工作人，对外部数据通过采集加工系统录入、审核，最终存入采集加工数据库和存储共享数据库中。

（2）历史数据清洗和迁移。对于旧系统遗留下来的业务数据或者从业务上引入的数据，为了使这些数据能够利用到新系统中，首先对这些数据做数据质量清洗工作，包括检查数据是否有重复、缺失等问题；其次对这些数据做格式转换，做旧系统数据库表结构和新系统数据库表结构的映射关系；最后将对应好的数据关系数据通过 ETL 工具或者开放软件的方式迁移到新系统数据库中。

4）数据库测试与调整

在模拟环境数据库（ODS）中注入模拟数据，通过使用应用系统相应子功能和专业数据库测试工具对数据库的响应速度、存储空间、安全性等性能进行单独测试；对测试过程中出现的数据库结构问题、索引问题、数据库性能参数问题进行查找分析，制定相应的修改方案对数据库进行调整优化。

3. 数据库设计的命名规范

（1）数据库对象的名称，不建议使用汉字。

例如以下语句不符合规范（表名和字段名使用了汉字）：

```
create table hr.用户
(
用户名 varchar2(100),
password varchar2(16)
);
```

以下语句符合规范：

```
create table hr.wap_user
(
username varchar2(100),
password varchar2(16)
);
```

(2) 命名应使用富有意义的英文，一般情况下不建议使用拼音命名。

例如以下语句不符合规范(表名和字段使用了拼音首字母简写)：

```
create table wap.wap_yonghu
(
yhm varchar2(100),
password varchar2(16)
);
```

以下语句符合规范：

```
create table wap.wap_user
(
username varchar2(100),
password varchar2(16)
);
```

(3) 同类业务的表以相同的英文开头(在逻辑上清晰，且可避免维护过程中对该类表的误操作)。

例如以下语句不符合规范(假定 wap_user 表和 user_login_log 表都属于 wap 类业务)：

```
create table wap.wap_user
(
username varchar2(100),
password varchar2(16)
);
create table wap.user_login_log
(
username varchar2(100),
logindate date
);
```

以下语句符合规范：

```
create table wap.wap_user
(
username varchar2(100),
password varchar2(16)
);
```

```
create table wap.wap_user_login_log
(
username varchar2(100),
logindate date
);
```

(4) 临时表以“tmp_”开头,后接功能描述。

例如以下语句不符合规范:

```
create global temporary table wap.tab_tmp1
(
username varchar2(100),
password varchar2(16)
);
```

以下语句符合规范:

```
create global temporary table wap.tmp_wap_user
(
username varchar2(100),
password varchar2(16)
);
```

(5) 同种用途的字段,在同一个业务的所有表中应有同样的字段类型和字段长度,并尽量保持一致的字段命名。

例如以下语句不符合规范(username 字段在两个有业务关系的表中字段长度不一致,易导致业务接口冲突):

```
create table wap.wap_user
(
username varchar2(100),
password varchar2(16)
);
create table wap.wap_user_login_log
(
username varchar2(80),
logindate date
);
```

以下语句符合规范:

```
create table wap.wap_user
(
username varchar2(100),
password varchar2(16)
);
create table wap.wap_user_login_log
(
username varchar2(100),
logindate date
);
```

5.3 图谱数据库

5.3.1 知识图谱概述

在数据治理过程中，当海量的信息以文本形式出现时，通过搜索所产生的结果的数据量也需要大量人工进行操作判断和应用。通常人工操作不利于大数据应用，这时候需要一些自动化的工具帮助数据管理者和数据应用者在海量的信息源中迅速找到真正需要的信息和线索或者数据关系，而对这些关系最好的描述就是知识图谱。

1. 认识知识图谱

知识图谱是指用可视化的方式描述人类随时间拥有的知识资源及其载体，绘制、挖掘、分析和显示科学技术知识以及它们之间的相互联系，在组织内创造知识共享的环境，以促进科学技术研究的合作和深入。

知识图谱本质上是一种揭示实体之间关系的语义网络。知识图谱以结构化的形式描述客观世界中的概念、实体及其关系，将互联网的信息表达成更接近人类认知世界的形式，提供了一种更好地组织、管理和理解互联网海量信息的能力。知识图谱给互联网语义搜索带来了活力，同时也在智能问答中显示出强大威力，已经成为互联网知识驱动的智能应用的基础设施。知识图谱最初是由谷歌提出用来优化搜索引擎的技术，在不断发展中外延也一度扩大。盘点目前知识图谱的发展，其实它已经助力了很多热门的人工智能场景的应用，例如语音助手Siri、聊天机器人、智能问答等。知识图谱与大数据和深度学习一起，成为推动互联网和人工智能发展的核心驱动力之一。

2. 知识图谱的构建

在构建知识图谱时，由于信息来源具有多样性，如何对半结构化、非结构化的信息进行处理，抽取出有效的知识单元是一个重要的议题。图 5-13 显示了从原始数据到形成知识图谱经历了知识抽取、知识融合(实体对齐)、模型构建、质量评估等步骤。

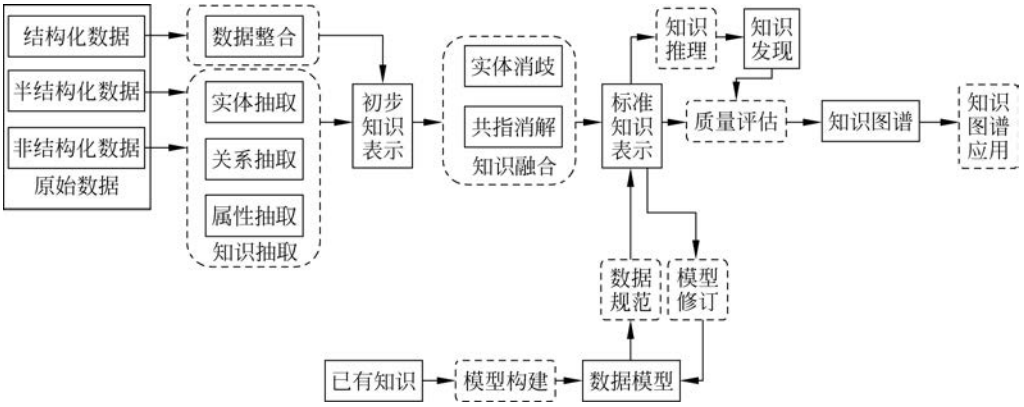


图 5-13 知识图谱的构建过程

此外，为了发现知识间的关系，更好地展示各单元，需要对样本数据进一步处理，即进行简化分析，当前采用较多的方式为关联分析、因子分析、多维尺度分析、自组织映射图(SOM)、寻址网络图谱(PTNET)、聚类分析、潜在语义分析、最小生成树法等。

知识图谱可以通过对关系的梳理或者对语义分析的知识抽取来绘制。知识图谱的构建可

以基于结构化信息规范资源描述框架；对半结构化数据，采用模式学习的方式来实现自动化信息抽取，这里需要通过人工调整或者新增模式等方法进行数据或者关系准确性描述；对非结构化数据，例如图像，可以基于深度学习图像智能识别化来进行定义，比如犯罪人员经常出现的场所附近的摄像头所采集的图像，哪些人经常出现在一个图像里面，然后进行数据关联并且分析应用。

5.3.2 知识图谱与图谱数据库

在知识图谱的存储研究中，目前主要是 RDF 数据库和图形数据库。从顶向下设计的 RDF 数据库没有从底向上设计的图形数据库成功，因此图形数据库在存储知识图谱的知识单元和单元关系上效果最佳。目前，图形数据库并没有一套完整的标准，但是大部分图形数据库都包含了节点、关系、属性 3 个元素。节点可以用来存储知识单元，关系可以用来展示知识单元之间的联系，属性可以表征知识单元的相关特性。

常见的图谱相关数据存储在 Neo4j 图形数据库中，以便于知识图谱的绘制和查询。Neo4j 是一个稳定且成熟的，具有较高性能的图形数据库，并且具有完整的 ACID 支持、高可用性、可扩展性，通过 Neo4j 的遍历工具可以高速检索数据。

Neo4j 的查询语言是一种可以对图形数据库进行查询和更新的图形查询语言——Cypher（也称为 CQL），它类似于关系数据库的 SQL。Cypher 的语法并不复杂，然而它的功能却非常强大，可以实现 SQL 难以实现的功能。例如，六度分割理论中曾指出任何两个人之间所间隔的人不会超过 6 个。因此，只要数据足够完整，采用 Cypher 可以很容易地找到任何两个人之间是通过哪些人联系起来的，而这一点 SQL 很难实现。

Neo4j 的特点如下：

（1）用 Cypher 作为查询语言。

（2）遵循属性图数据模型。

（3）通过使用 Apache Lucene 支持索引。

（4）支持 UNIQUE 约束。

（5）包含一个用于执行 Cypher 命令的 UI——Neo4j 数据浏览器。

（6）支持完整的 ACID（原子性、一致性、隔离性和持久性）规则。

（7）采用原生图形库与本地 GPE（图形处理引擎）。

（8）支持将查询的数据导出为 JSON 和 XLS 格式。

（9）提供了 REST API，可以被编程语言（例如 Java、Spring、Scala 等）访问。

（10）提供了可以通过 UI MVC 框架（例如 Node JS）访问的 Java 脚本。

（11）支持 Cypher API 和 Native Java API 两种 Java API 来开发 Java 应用程序。

Neo4j 的优点如下：

（1）很容易表示连接的数据。

（2）检索/遍历/导航更多的连接数据是非常容易和快速的。

（3）非常容易地表示半结构化数据。

（4）查询语言 Cypher 的命令是人性化的可读格式，非常容易学习。

（5）使用简单而强大的数据模型。

（6）不需要复杂的连接来检索连接的/相关的数据，因为很容易检索其相邻节点或关系细节没有连接或索引。

Neo4j 存储如图 5-14 所示。

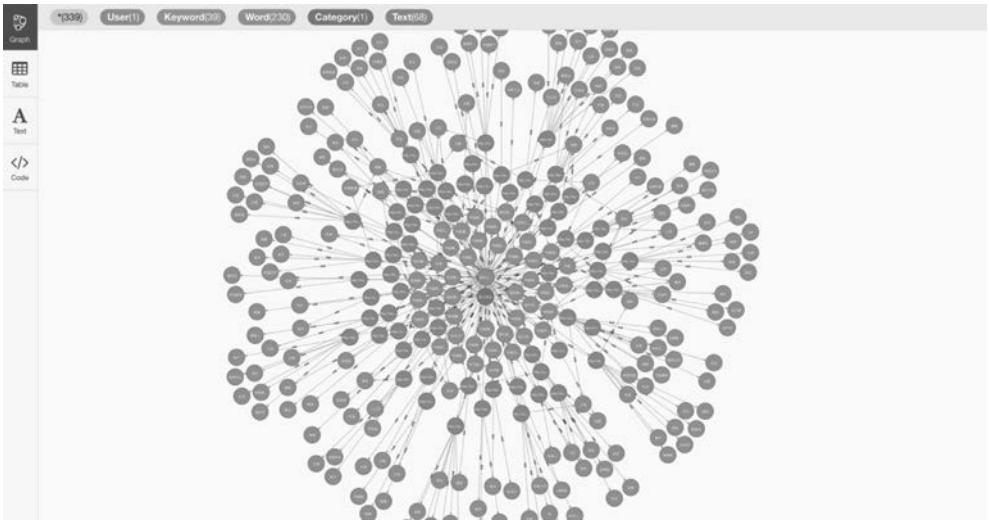


图 5-14 Neo4j 存储

5.4 本章小结

(1) 数据库技术是计算机领域中的重要技术之一,它将各种数据按一定的规律存放,以便于用户查询和处理。

(2) 数据库管理系统(Database Management System,DBMS)是一种操作和管理数据库的软件,它是数据库的核心,主要用于创建、使用和维护数据库。

(3) 数据库系统在总体结构上一般体现为三级模式,分别是模式、外模式和内模式。

(4) E-R 图也称为实体-联系图,它提供了表示实体类型、属性和联系的方法,用来描述现实世界的概念模型。

(5) 通常在建立数据库模型时会涉及 3 种具体的数据模型,分别是概念模型、逻辑模型和物理模型。

(6) 数据字典(Data Dictionary)是一种用户可以访问的记录数据库和应用程序元数据的目录,是对于数据模型中的数据对象或者项目的描述的集合。

5.5 实训

1. 实训目的

通过本章实训了解数据库治理的特点,能进行简单的与数据库治理有关的操作。

2. 实训内容

(1) 在 MySQL 当前会话中使用命令 show profiles 分析 SQL 语句执行时的资源消耗情况,该命令可以用于 SQL 的调优测量。在默认情况下,该命令处于关闭状态。

① 启动 MySQL 并输入命令“show profiles;”,可以看到在默认情况下该命令是关闭的,如图 5-15 所示。

② 查看当前的 MySQL 版本是否支持,命令为“show variables like"%pro%";”。

该命令默认是关闭的,在使用前需要开启,如图 5-16 所示。

```
mysql> show profiles;
Empty set, 1 warning (0.00 sec)
```

图 5-15　查看该命令的状态

```
mysql> show variables like "%pro%";
+----------------------------+-------+
| Variable_name              | Value |
+----------------------------+-------+
| have_profiling             | YES   |
| profiling                  | OFF   |
| profiling_history_size     | 15    |
| protocol_version           | 10    |
| proxy_user                 |       |
| slave_compressed_protocol  | OFF   |
| stored_program_cache       | 256   |
+----------------------------+-------+
7 rows in set (0.00 sec)
```

图 5-16　查看当前的 MySQL 版本是否支持

③ 开启功能，输入命令“set profiling=1;”，如图 5-17 所示。

④ 输入命令“show variables like "%pro%";”，查看到该命令已经开启，如图 5-18 所示。

```
mysql> set profiling=1;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

图 5-17　开启功能

```
mysql> show variables like "%pro%";
+----------------------------+-------+
| Variable_name              | Value |
+----------------------------+-------+
| have_profiling             | YES   |
| profiling                  | ON    |
| profiling_history_size     | 15    |
| protocol_version           | 10    |
| proxy_user                 |       |
| slave_compressed_protocol  | OFF   |
| stored_program_cache       | 256   |
+----------------------------+-------+
7 rows in set (0.00 sec)
```

图 5-18　已经开启该命令

⑤ 运行 SQL 命令，例如：

```
help profiles;
show variables like "%pro%";
select * from emp group by id%10 limit 150000;
select * from user group by id%10 limit 150000;
```

输入命令“show profiles;”，查看 SQL 语句执行时的资源消耗情况，如图 5-19 所示。

```
mysql> show profiles;
+----------+-------------+-------------------------------------------------+
| Query_ID | Duration    | Query                                           |
+----------+-------------+-------------------------------------------------+
|        1 | 0.04312350 | help 'profiles'                                  |
|        2 | 0.00176800 | show variables like "%pro%"                      |
|        3 | 0.00008400 | select * from emp group by id%10 limit 150000    |
|        4 | 0.00014250 | select * from user group by id%10 limit 150000   |
+----------+-------------+-------------------------------------------------+
4 rows in set, 1 warning (0.00 sec)
```

图 5-19　查看 SQL 语句执行时的资源消耗情况

⑥ 分析第 4 条 SQL 语句，输入以下命令：

```
show profile block io,cpu for query 4;
```

运行结果如图 5-20 所示。

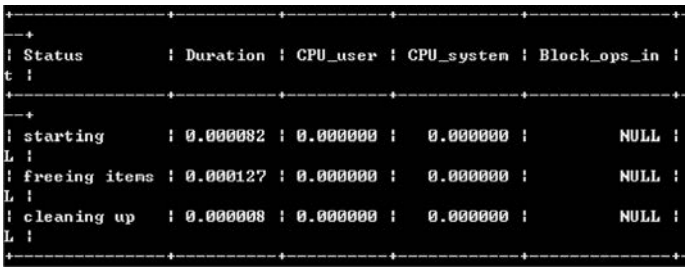
图 5-20 中字段的含义如下。

Status：SQL 语句执行的状态。

Duration：SQL 执行过程中每一个步骤的耗时。

CPU_user：当前用户占有的 CPU。

CPU_system：系统占有的 CPU。



Status	Duration	CPU_user	CPU_system	Block_ops_in	t
starting	0.000082	0.000000	0.000000	NULL	L
freeing items	0.000127	0.000000	0.000000	NULL	L
cleaning up	0.000008	0.000000	0.000000	NULL	L

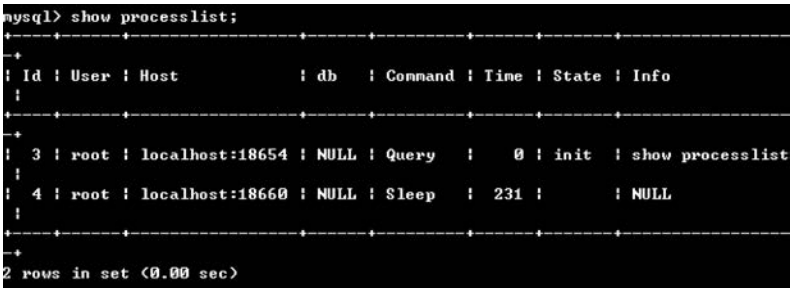
图 5-20 查看分析结果

Block_ops_in：I/O 输入。

Block_ops_out：I/O 输出。

(2) 使用命令 show processlist 显示 MySQL 中哪些线程正在运行。需要注意的是，如果使用者有 SUPER 权限，则可以看到全部的线程。

① 运行 MySQL，输入命令“show processlist;”，如图 5-21 所示。



```
mysql> show processlist;
+----+------+-----------------+------+---------+------+-------+------------------
-+
| Id | User | Host            | db   | Command | Time | State | Info
 |
+----+------+-----------------+------+---------+------+-------+------------------
-+
|  3 | root | localhost:18654 | NULL | Query   |    0 | init  | show processlist
 |
|  4 | root | localhost:18660 | NULL | Sleep   |  231 |       | NULL
 |
+----+------+-----------------+------+---------+------+-------+------------------
-+
2 rows in set (0.00 sec)
```

图 5-21 显示哪些线程正在运行

图 5-21 中字段的含义如下。

Id：用户登录 MySQL 时系统分配的 connection_id，可以使用 connection_id()函数查看。

User：显示当前用户。如果不是 root，则该命令只显示用户权限范围的 SQL 语句。

Host：显示该语句是从哪个 IP 的哪个端口上发的，可以用来跟踪出现问题语句的用户。

db：显示进程目前连接的是哪个数据库。

Command：显示当前连接执行的命令，一般取值为 Sleep(休眠)、Query(查询)、Connect(连接)等。

Time：显示状态持续的时间，单位是秒。

State：显示使用当前连接的 SQL 语句的状态。

Info：显示这个 SQL 语句。

② show processlist 只能列出前 100 条线程，如果想列出全部线程，可以使用命令 show full processlist 来实现，如图 5-22 所示。

(3) MySQL 数据库安全。

①下载并安装 MySQL，设置 root 密码为 123456，进入 MySQL 中创建数据库 test1、test2 和 test3，如图 5-23 所示。

② 使用命令 create user 创建新的 MySQL 账户 user1 和 user2，user1 的密码为 1234，user1 的密码为 12345，如图 5-24 所示。

```
mysql> show full processlist;
+----+------+-----------------+------+---------+------+-------+------------------
------+
| Id | User | Host            | db   | Command | Time | State | Info
      |
+----+------+-----------------+------+---------+------+-------+------------------
------+
|  3 | root | localhost:18654 | NULL | Query   |    0 | init  | show full proces
slist |
|  4 | root | localhost:18660 | NULL | Sleep   |  258 |       | NULL
      |
+----+------+-----------------+------+---------+------+-------+------------------
------+
2 rows in set (0.00 sec)

mysql>
```

图 5-22　使用 show full processlist 列出全部线程

```
mysql> create database test1;
Query OK, 1 row affected (0.01 sec)

mysql> create database test2;
Query OK, 1 row affected (0.00 sec)

mysql> create database test3;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| test               |
| test1              |
| test2              |
| test3              |
| world              |
+--------------------+
9 rows in set (0.00 sec)
```

图 5-23　进入 MySQL 中创建数据库

```
mysql> create user
    -> 'user1'@'localhost' identified by '1234',
    -> 'user2'@'localhost' identified by '12345';
Query OK, 0 rows affected (0.02 sec)
```

图 5-24　创建新的 MySQL 账户

③ create user 会在 MySQL 系统自身的 MySQL 数据库的 user 表中添加新记录。输入以下命令查看刚才添加的记录：

```
use mysql
show tables;
select * from user
```

运行结果如图 5-25 和图 5-26 所示。

```
mysql> use mysql
Database changed
mysql> show tables;
+---------------------------+
| Tables_in_mysql           |
+---------------------------+
| columns_priv              |
| db                        |
| event                     |
| func                      |
| general_log               |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
| innodb_index_stats        |
| innodb_table_stats        |
| ndb_binlog_index          |
| plugin                    |
| proc                      |
| procs_priv                |
| proxies_priv              |
| servers                   |
| slave_master_info         |
| slave_relay_log_info      |
| slave_worker_info         |
| slow_log                  |
| tables_priv               |
| time_zone                 |
| time_zone_leap_second     |
| time_zone_name            |
| time_zone_transition      |
| time_zone_transition_type |
| user                      |
+---------------------------+
28 rows in set (0.01 sec)
```

图 5-25　查看 MySQL 库中的表

图 5-26 查看已经创建的新账户

④ 新的 SQL 用户不能访问其他 SQL 用户的表，也不能创建自己的表，除非被授予权限。在 MySQL 中给用户授予权限可以用命令 grant 来实现。

授予 user1 在 test 数据库中的所有表的 SELECT 权限，如图 5-27 所示。

```
mysql> grant select
    -> on test .*
    -> to user1@localhost;
Query OK, 0 rows affected (0.02 sec)
```

图 5-27 授予 user1 在 test 数据库中的所有表的 SELECT 权限

⑤ 授予 user2 在 test 数据库中的所有数据库权限，如图 5-28 所示。

```
mysql> grant all
    -> on test .*
    -> to user2@localhost;
Query OK, 0 rows affected (0.00 sec)
```

图 5-28 授予 user2 在 test 数据库中的所有数据库权限

(4) 使用 Power Designer 实现数据模型。

① 下载并安装 Power Designer 16.5，启动界面如图 5-29 所示。

② 在菜单栏中选中 File|New Model，并选择 Conceptual Data Model，如图 5-30 所示。

③ 在打开的界面中，将右侧的 Toolbox 栏中的 Entity 图标拖动到设计面板中，如图 5-31 所示。

④ 双击 Entity 图标，在弹出的 General 标签页和 Attributes 标签页中分别设置内容如图 5-32 和图 5-33 所示。General 标签页设置表的基本信息；Attributes 标签页设置该表的属性。

⑤ 继续拖动 Entity 图标到设计面板中，命名为“学生表”，具体设置信息如图 5-34 和图 5-35 所示。

⑥ 使用 Relationship(关系)为班级表和学生表建立关系，具体设置如图 5-36 和图 5-37 所示。在图 5-37 中设置一对多关系(班级对学生)。

最终效果如图 5-38 所示。

图 5-29　Power Designer 16.5 启动界面

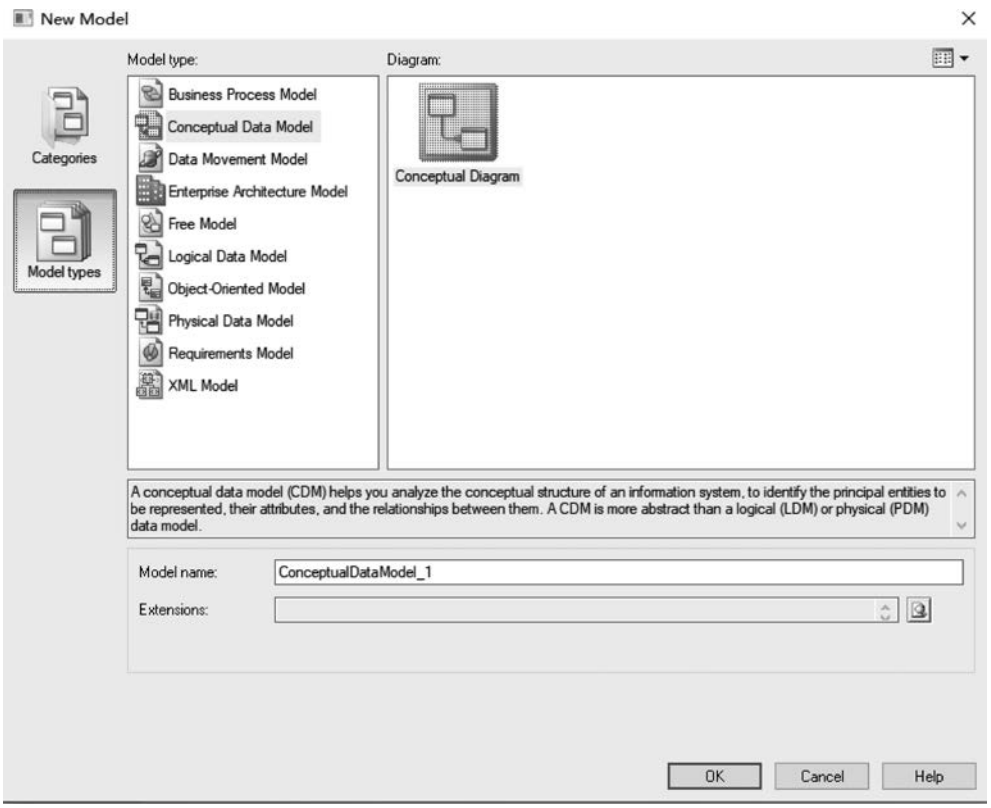


图 5-30　新建模型

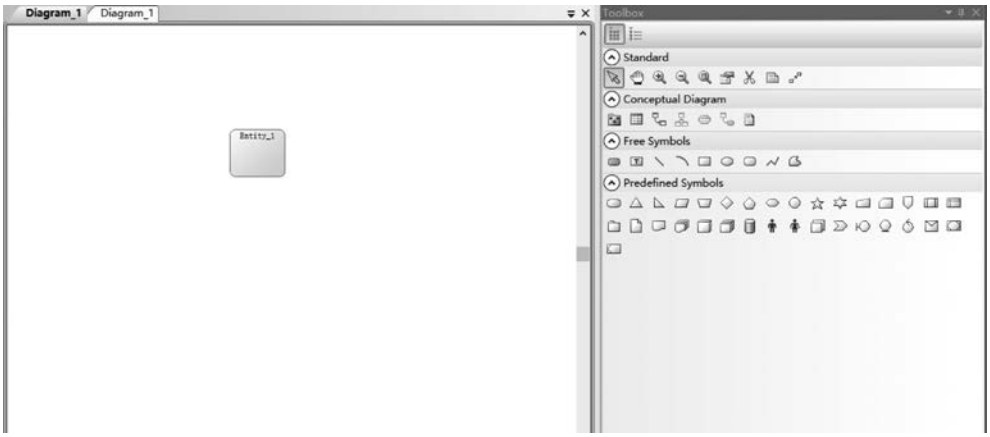


图 5-31 拖动 Entity 图标

图 5-32 设置表基本信息

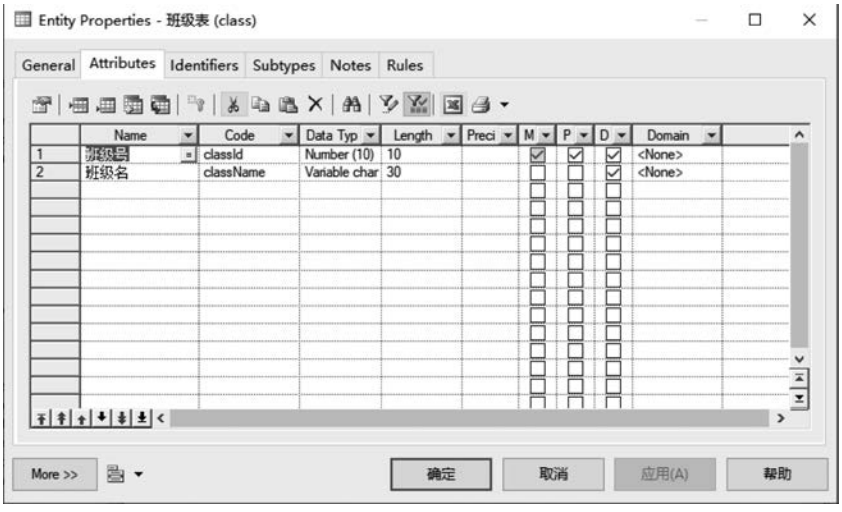


图 5-33 设置表属性

图 5-34　设置表基本信息

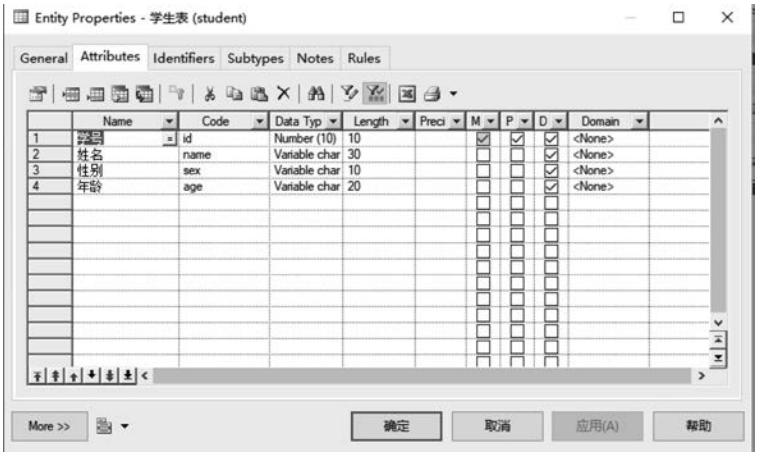


图 5-35　设置表属性

图 5-36　为班级表和学生表建立关系

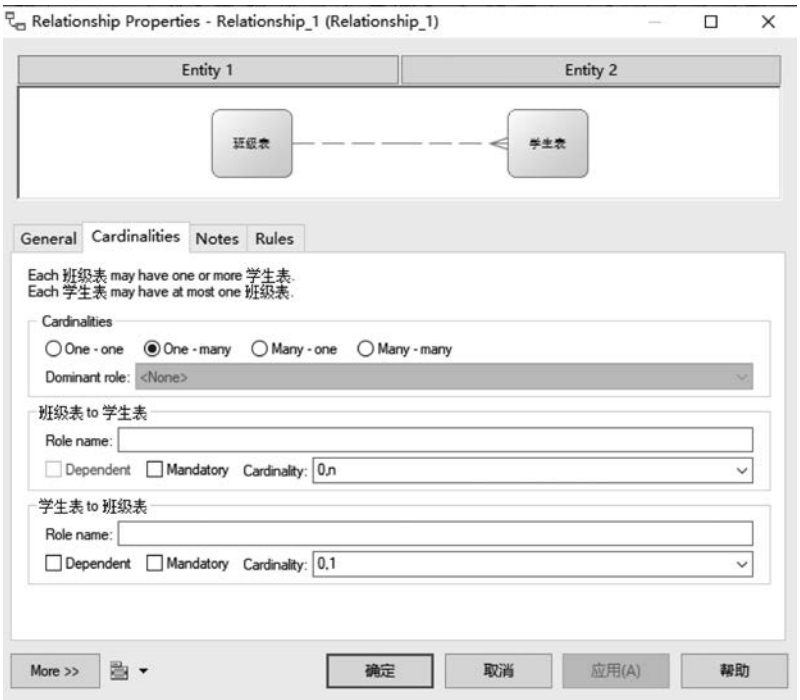


图 5-37 设置一对多关系

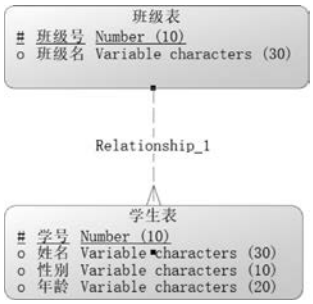


图 5-38 最终的数据模型

习题 5

(1) 请阐述什么是数据库。

(2) 请阐述数据库系统的结构。

(3) 请阐述什么是数据字典。