关系数据库

关系数据库是建立在关系数据模型基础上的数据库，借助于关系代数等概念和方法来处理数据库中的数据。目前主流的数据库产品几乎全部是关系型数据库，例如 oracle（中文名称叫甲骨文）公司的 oracle 数据库，微软公司的 SQL Server，赛贝斯公司的 sybase，英孚美软件公司的 informix 以及开源免费的 MySql 等等。

一、关系模型概述

关系数据模型是以集合论中的关系概念为基础发展起来的。关系模型中无论是实体还是实体间的联系均由单一的结构类型—关系来表示。在实际的关系数据库中的关系也称表。一个关系数据库就是由若干个表组成。

根据前面对数据模型的描述，关系数据模型应该由关系数据结构、关系操作集合和关系完整性约束三部分组成。

1．关系数据结构

数据模型中的数据结构描述数据的静态特性。关系模型的数据结构非常单一，在关系模型中，现实世界中的所有事物（实体）及其联系均用关系来表示。而这里所说的关系，就是平常我们常用的二维表格。如下表 1-1 即为教师关系，记录了教师的教工号、姓名、性别、年龄、职称等基本信息。

表 1-1　教师关系

教工号	姓名	性别	年龄	职称
1996000011	张伍合	男	39	讲师
2000000003	李映梅	女	35	讲师
2003000008	彭兰明	男	52	副教授
2004000001	赵巧巧	女	28	助教
1988000006	吴好	男	43	教授

2．　关系操作

关系模型中的关系操作一般包括查询和编辑两大类。查询操作有：选择、投影、连接、并、交、差等，这些操作不会改变参加运算的原关系中的数据，只是在原关系中挑选满足要求的元组或属性组成新的结果关系。编辑类操作包括插入、删除和修改，这些操作会改变原关系中的数据。

由于关系可理解为若干元组（行）组成的集合，因此关系操作的特点是集合操作方式，即操作的对象和结果都是元组（行）的集合。

3．　数据完整性约束

关系的完整性约束包括实体完整性、参照完整性和用户定义的完整性三大类。其中实体完整性是保证关系中每个实体的唯一性必须要满足的约束，在具体的 DBMS 实现中表现为表中主键的设置。参照完整性是保证关系之间的联系正常有效而同样必须要满足的约束，在具体的 DBMS 中表现为外键的设置。用户定义的完整性是实际的应用领域需要遵循的约束条件，体现了具体应用的实际约束，如给定条件的检查约束等。

二、　关系数据库的基本术语

数据以“关系”的形式表示，也就是二维表的形式表示，其数据模型就是我们所说的关系模型。用关系模型表示的数据库是关系数据库，在关系数据库中所有数据及数据之间的联系均用“关系”来表达，并且对“关系”进行各种处理之后得到的还是“关系”。

1. 属性和域

在现实世界中我们描述一个事物常常需要取其若干特征来表示，如一个学生的姓名、性别、年龄、身高、体重和籍贯等，如果用关系表示事物，我们将事物的这些特征称为属性。每个属性的取值范围所对应的一个值的集合称为该属性的域。

例如课程关系中的课程号、课程名称、学分等是属性，而课程号的取值范围是是三位的数字字符，即{001~999}，这个取值范围就是我们所说的域。

域是一组具有相同数据类型的值的集合。例如：人名的集合{张三，李四，王五}，性别的集合{男，女}，整数 1-100 的集合{1，2，…，99，100}，实数集合，1900 年以来的日期集合等都有是域。

2. 笛卡尔乘积

笛卡尔乘积的形式化定义：结定一组域 D_1，D_2，…，D_n，这些域中可以有相同的。D_1，D_2，…，D_n 的笛卡尔乘积为：

$D_1 \times D_2 \times \ldots \times D_n = \{（d_1，d_2，\ldots，d_n） \mid d_i \in D_i，i=1，2，\ldots，n\}$

其中每一个元素（d_1，d_2，…，d_n）叫作一个 n 元组或简称为元组。元组中的每一个值 d_i 叫做元组的一个分量。

假设域 A 为学生姓名={张明，李好}，域 B 为学生年龄={18，19，20}，域 C 为学生性别={男，女}则 A、B 与 C 的笛卡尔积为：

A×B×C={（张明，18，男），（张明，18，女），
（张明，19，男），（张明，19，女），
（张明，20，男），（张明，20，女），
（李好，18，男），（李好，18，女），
（李好，19，男），（李好，19，女），
（李好，20，男），（李好，20，女）}

这 12 个元组构成的二维表如表 1-2 所示。

表 1-2　A、B 与 C 的笛卡尔积

姓名	年龄	性别
张明	18	男
张明	18	女
张明	19	男
张明	19	女
张明	20	男
张明	22	女
李好	18	男
李好	18	女
李好	19	男
李好	19	女
李好	20	男
李好	22	女

其中（张明，18，男）和（李好，20，女）等都是元组，张明、18 和男是元组“（张明，18，男）”的分量，李好、20 和女是元组“（李好，20，女）”的分量。

类似的例子有，如果 A 表示某学校学生的集合，B 表示该学校所有课程的集合，则 A 与 B 的笛卡尔积表示所有学生的所有可能的选课情况。

很显然，笛卡尔乘积概括了给定域的所有可能的组合情况，不一定有实际意义。有实际意义的内容往往是其中的部分分量组成，这就是我们所说的关系。

3. 关系

关系的形式化定义：结定一组域 D_1，D_2，…，D_n，这些域中可以有相同的。$D_1 \times D_2 \times \ldots \times D_n$ 的子集叫作在域 D_1，D_2，…，D_n 上的关系。表示为：

R（D_1，D_2，…，D_n）

这里 R 表示关系的名字，n 是关系的目或度，即表中列的数目。

关系中每个分量元素叫做关系中的元组（即表中的一行或记录）。

实际上，关系是笛卡儿积的有一定意义的、有限的子集，所以关系也是一个二维表，表中的每一行对应一个元组，表的每一列对应一个域。由于域可以相同，为了加以区分，必须对每列起一个唯一的名字，称为属性。n 目关系有 n 个属性，当 n=1 时，称该关系为单元关系，当 n=2 时，称该关系为二元关系。

例如上面介绍的 A×B×C 没有什么意义，而学生关系 1 则表示每个学生的姓名、年龄和性别等有意义的信息，显然表 1-3 是表 1-2 的子集。

表 1-3 学生关系 1

姓名	年龄	性别
张明	19	男
李好	18	女

例如：对给定的三个域：D1(年份集合＝1992，1993)、D2(电影名集合=星球大战，独立日)、D3(电影长度集合＝100，120)，它们的笛卡儿积构成的集合，不是一个有意义的关系，因为，每个电影的长度是固定的，电影的出版年份也是固定的。而如表 1-4 所示的电影关系是有意义的。

表 1-4 电影关系

年份	电影名	电影长度
1993	星球大战	120
1992	独立日	100

例如表 1-5 是一个学生关系，记录了学生的学号、姓名、性别和年龄信息。表的每一列标题栏中的名字称为关系的属性，属性描述了该列数据的意义；表中除了标题行之外的每一行称之为关系的元组或记录，它表示了具体的数据信息。例如表 1-5 中有 4 个元组，记录了 4 个学生的学号、姓名、性别和年龄信息。

表 1-5 “学生”关系

属性 1　属性 2　属性 3　属性 4

学号	姓名	性别	年龄	
20120003001	张小光	男	19	←元组 1（记录 1）
20120003002	刘和平	男	20	←元组 2（记录 2）
20120003003	陈一新	女	19	←元组 3（记录 3）
20120003004	蔡忠明	男	21	←元组 4（记录 4）

表 1-6 “课程”

课程号	课程名	学分
001	大学英语	3
002	高等数学	4
003	程序设计语言	5
004	数据库技术	4

表 1-7 “成绩”关系

学号	课程号	成绩

20120003001	001	89
20120003001	003	70
20120003001	004	68
20120003002	002	90
20120003002	003	52
20120003003	001	72
20120003004	001	60
20120003004	002	80

4. 关系模式

关系模式是对关系的描述，是关系的型。关系的内容是关系的值，即一行行的数据记录（元组）。

我们知道，关系实质上是一张二维表，表中的每一行是一个元组，每一列为一个属性。一个元组就是该关系所涉及的属性集的笛卡尔积的一个元素。关系是元组的集合，因此关系模式必须指出这个元组集合的结构，即它由哪些属性组成，这些属性来自哪些域，以及属性与域之间的关系。简化的情况下，我们将关系名称和关系的属性名称的集合称为该关系的模式，记为：

关系名 (属性名 1，属性名 2，…，属性名 n)

所以表 1-5 所示的学生关系对应的关系模式为：

学生（学号，姓名，性别，年龄）

表 1-6 所示的课程关系对应的关系模式为：

课程（课程号，课程名，学分）

关系模式有时简称为模式，模式中的属性是一个集合而非有序列表，也就是说属性名的排列顺序不影响关系模式，但为了便于说明和讨论，我们一般会为这些属性根据其重要程度而规定一个“标准”顺序。

关系模式只是一个关系的框架，具有该框架结构的所有元组才是该关系的值，或者说是该关系的内容。关系的模式和关系的值共同确定了一个具体关系。在关系数据库系统中，一个关系可以只有模式而没有值，称为空关系，而绝对不可能没有模式只有值。关系模式一经定义，一般尽量不要修改，通常来说，模式是相对稳定的，而关系的值是具体的数据，它随时都可能被更新，即从关系中删除一个元组或修改一个元组中的某个分量等操作都改变了关系的值，使用权关系具有了新的当前值。

5. 关系数据库模式

在关系模型中，现实世界中的实体以及实体之间的联系都是用关系来表示的。例如学生实体，课程实体和成绩关系都分别用一个关系来表示。在一个给定的应用系统中，所有实体和实体之间的联系的关系的集合构成一个关系数据库。

一个关系数据库中往往包含多个关系，关系数据库中这些关系的集合称之为“数据库模式”，数据库设计的主要任务是确定其中需要多少个关系，每个关系有多少个属性，属性的名称和数据类型等内容，也就是设计好每个关系的模式。

如果学生成绩管理数据库包含三个关系，学生、课程和成绩，其数据库关系模式如下所示。

学生（学号，姓名，性别，年龄）

课程（课程号，课程名，学分）

成绩（学号，课程号，成绩）

即学生、课程和成绩三个关系模式的集合就构成了学生成绩管理数据库的模式。

6．其他的相关名词

（1）关系的度或目：每个关系都要给定一外名称，称为关系的名字，一般用 R 表示关系的名。给定关系中不同属性列的数目称之为关系的目或度（度数），用 n 表示关系的度或目。例如表 1-5 所表示的学生关系共有 4 个属性列，所以此关系的度 n 为 4；表 1-6 所表示的课程关系中，共有三个属性列，所以其度数 n 为 3。

（2）主键(Primary key）:关系中能唯一标识每个元组的最少属性或属性组称之为主键（也称之为：**关键字**或**主码**）。例如“学生”关系中的属性“学号”就是主键，只要学号确定了，就能知道这个学号对应的姓名、性别和年龄等信息，但学生关系中的“性别”和“年龄”不能作为主键，因为即使年龄或性别确定了，还是不能确定学生的姓名和学号等信息，同性别或者同年龄的学生太多了。当然如果这个关系中没有同姓名的学生，则姓名也可以作为主键看待，这要根据具体的语义来决定。当一个关系有多个可选的主键（称之为候选码）时，可由关系的设计者或使用者指定其中之一为主健。

在表 1-6 所示的“课程”关系中，“课程号”或“课程名”均可以作为关系的主键，但有可能不同的系别选择同样的课程名称，但课程内容有区别，所以一般使用“课程号”作为主键比较合适。而在表 1-7 所示的“成绩”关系中，单独的学号和课程号都不能作为主键，因为一个学号代表一个学生，每个学生可以选修多门课，从而对应每个学号会有多个记录，同样地，每个课程号对应一门课程，每门课程会有多个学生选修，也就会对应表中多条记录。只有学号和课程号的组合（学号，课程号）才能确定相应的成绩，所以（学号，课程号）为“成绩”关系的主键。主键可能是属性的组合，但必须是最少的。就是说少一个属性不能成为主码，但多一个就有了冗余。

（3）主属性：关系中包含在候选码中的属性称为主属性。需要注意的是，主键中的属性是主属性，如上面介绍的学生关系中的学号，课程关系中的课程号，成绩关系中的学号和课程号都是主属性。如果关系中有多个可选的主键，即候选码，即便这个候选码没有选择作为主键，那么这些候选码中包含的属性也是主属性。例如：如果学生关系中增加一个“身份证号码”属性，那么“身份证号码”也是学生关系中的候选码，即是主属性。

（4）外关键字（Foreign key）：又称为外键或外码。在一个关系数据库中包含有多个关系，如关系 R1、R2 等，如果某个关系 R1 中的某个属性在这个关系中不是主键，而这个属性在另一个关系 R2 中是主键，则该属性为 R1 的外码、外键或外关键字。

例 1-1：假设我们设计的数据库包含表 1-5 至表 1-7 所示的三个关系

学生（学号，姓名，性别，年龄）

课程（课程号，课程名，学分）

成绩（学号，课程号，成绩）

因“成绩”关系中的“学号”属性不是主码，但是这个属性在“学生”关系中是主码，所以“学号”是“成绩”关系中的外码；同样地，“成绩”关系中的“课程号”主码，而在“课程”关系中是主码，所以“课程号”也是“成绩”关系中的外码。这里需要注意的是，成绩关系中的主码是“学号”与“课程号”的组合，而不是其中的某一个属性。在这个例子中，通过成绩关系中的“学号”和“课程号”这两个外码将三个关系联系成一个整体。

例 1-2：学校教工和系部关系模式如下所示

系部（系部编号，系部名称，系部主任，办公室，联系电话）

教工（教工号，姓名，性别，年龄，职称，系部编号）

在教工关系中，“系部编号”属性不是主码，但在系部关系中，“系部编号”是主码，因此，“系部编号” 在教工关系中是外码。在这个例子中，通过“系部编号”属性将系部关系和教工关系联系起来了。

因此，在一个关系数据库中的若干关系往往是通过外码（外关键字、外键）而相互关联

的。

外码（外关键字、外键）这个概念在关系数据库中相当重要，请同学们一定好好理解并掌握其中的涵义。

7. 关系及关系数据库的特点

（1）关系的特点

根据前面的说明，我们总结出关系的特点有：

- 关系中的每一个属性值都必须是不能再分的元素。例如学生的“姓名”不能再细分为“姓”和“名”两个属性值，必须把其作为一个整体来看待。
- 每一列中的数值是同类型的数据。例如学生的年龄列为整数值，性别列从{“男”，“女”}中取值等。
- 不同的列应该给予不同的属性名。同一个关系中的两个列即使其取值范围相同也必须有不同的属性名，以便区分其不同意义。
- 同一关系中不允许有相同的元组。如果有相同的元组也只保留一个。
- 关系是行或列的集合，所以行、列的次序可以任意交换，不影响关系的实际意义。

（2）关系数据库的特点

采用关系模型建立数据库即关系数据库，具有以下特点：

- 组织数据的结构单一

在关系模型中，无论是数据还是数据之间的联系都是以我们熟悉的二维表（关系）形式来表示的，这种表示方法不仅让人容易理解且便于计算机操作和实现。

- 采用集合运算

在关系模型中，运算的对象是关系，运算的结果还是关系，而关系可以看作是行（元组或记录）的集合，所以对关系的运算可以转化为对集合的运算。

- 数据完全独立

因为关系数据库系统中的数据是由关系数据库管理系统（DBMS）进行管理的，对于程序员来说，不需要知道数据存放的具体位置和组织形式等方面的内容，只需要告诉系统要进行什么样的操作，由系统自动完成相关的任务，即程序和数据高度独立。

- 数学理论支持

在关系模型中，每个关系都是集合，对关系的运算有集合论、数理逻辑做基础，关系结构可以用关系规范化理论进行优化。总之，关系模型具有严格的数学定义，具有成熟的数学理论为依据，它是目前为止最简单有效、最受欢迎、最广泛应用的数据模型。