数据库系统概论（2）

- **第二章 关系模型介绍（半重点）**
- 关系模型利用表的集合来表示数据和数据间的联系
  - **2.1 关系数据库的结构**
    - 关系数据库由*表(table)*的集合构成，每个表有唯一的名字。
    - 在关系模型的术语中，*关系(relation)*用来指代表；而*元组（tuple) 用来指代行；*类似地，*属性(attribute)* 指代的是表中的列。
    - 示例如图2-1

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

图 2-1 *instructor* 关系

    - 我们用*关系实例(relation instance)*这个术语来表示一个关系的特定实例.也就是所包含的一组特定的行。
    - 对于关系的每个属性，都存在一个允许取值的集合，称为该属性的域(domain) 。
    - 我们要求对所有关系r而言，r的所有属性的域都是原子的。如果域中元素被看作是不可再分的单元，则域是*原子的(atomic)*。通俗来说，原子性即最小可分。
    - 空(null)值是一个特殊的值，表示值未知或不存在。
  - **2.2 数据库模式**
    - 当我们谈论数据库时，我们必须区分数据库模式(database schema)和数据库实例(database instance),前者是数据库的逻辑设计，后者是给定时刻数据库中数据的一个快照。
    - 不同关系模式中可存在相同属性，可将不同关系的元组联系起来。

代模式，也指代实例。在需要的时候，我们会显示地指明模式或实例。例如“*instructor* 模式”或“*instructor* 关系的一个实例”。然而，在模式或实例的含义清楚的情况下，我们就简单地使用关系的名字。

考察图 2-5 中的 *department* 关系，该关系的模式是：

department (*dept_name*, *building*, *budget*)

请注意属性 *dept_name* 既出现在 *instructor* 模式中，又出现在 *department* 模式中。这样的重复并不是一种巧合。实际上，在关系模式中使用相同属性正是将不同关系的元组联系起来的一种方法。例如，假设我们希望找出在 Watson 大楼工作的所有教师的相关信息。我们首先在 *department* 关系中找出所有位于 Watson 的系的 *dept_name*。接着，对每一个这样的系，我们在 *instructor* 关系中找出与 *dept_name* 对应的教师信息。

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

图 2-5 *department* 关系

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

图 2-4 *instructor* 关系的无序显示

- 例子如下所示：

我们继续看大学数据库的例子。

大学里的每门课程可能要讲授多次，可以在不同学期授课，甚至可能在同一个学期授课。我们需要一个关系来描述每次课的授课情况或分段情况。该关系模式为：

section (*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)

图 2-6 给出了 *section* 关系的一个示例。

我们需要一个关系来描述教师和他们所讲授的课程段之间的联系。描述此联系的关系模式是：

teaches (*ID*, *course_id*, *sec_id*, *semester*, *year*)

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

图 2-6 *section* 关系

图 2-7 给出了 *teaches* 关系的一个示例。

正如你可以料想的，在一个真正的大学数据库中还维护了更多的关系。除了我们已经列出的这些关系：*instructor*、*department*、*course*、*section*、*prereq* 和 *teaches*，在本书中我们还要使用下列关系：

- *student* (*ID*, *name*, *dept_name*, *tot_cred*)
- *advisor* (*s_id*, *i_id*)
- *takes* (*ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*)
- *classroom* (*building*, *room_number*, *capacity*)
- *time_slot* (*time_slot_id*, *day*, *start_time*, *end_time*)

2.3 码

我们必须有一种能区分给定关系中的不同元组的方

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

图 2-7 *teaches* 关系

- **2.3 码**
  - 码不论是主码、候选码或超码，都是整个关系的一种性质，而不是单个元组的性质 。
  - 一个元组的属性值必须是能够唯一区分元组的。换句话说， 一个关系中没有两个元组在所有属性上的取值都相同。

- ***超码(super key)***是一个或多个属性的集合，这些属性的组合可以使我们在一个关系中唯一地标识一个元组。超码中可能包含无关紧要的属性。
- 我们通常只对这样的一些超码感兴趣，它们的任意真子集都不能成为超码。这样的最小超码称为**候选码(candidate key)**。
- 我们用**主码(primary key)**这个术语来代表被数据库设计者选中的、主要用来在一个关系中区分不同元组的候选码。
- 一个关系模式可能在它的属性中包括另一个关系模式的主码。这个属性在r1上称作参照r2的**外码(foreign key)** 。关系r1也称为外码依赖的参照关系(referencing relation) , r2叫做外码的被参照关系(referenced relation) 。
- 从section到teaches的约束是参照**完整性约束(referential integrity constraint)**的一个例子。参照完整性约束要求在参照关系中任意元组在特定属性上的取值必然等于被参照关系中某个元组在特定属性上的取值。

- **2.4 模式图**
  - 一个含有主码和外码依赖的数据库模式可以用*模式图(schema diagram)*来表示。
  - 图2-8展示了我们大学组织的模式图。每一个关系用一个矩形来表示，关系的名字显示在矩形上方，矩形内列出各属性。 主码属性用下划线标注。外码依赖用从参照关系的外码属性到被参照关系的主码属性之间的箭头来表示。

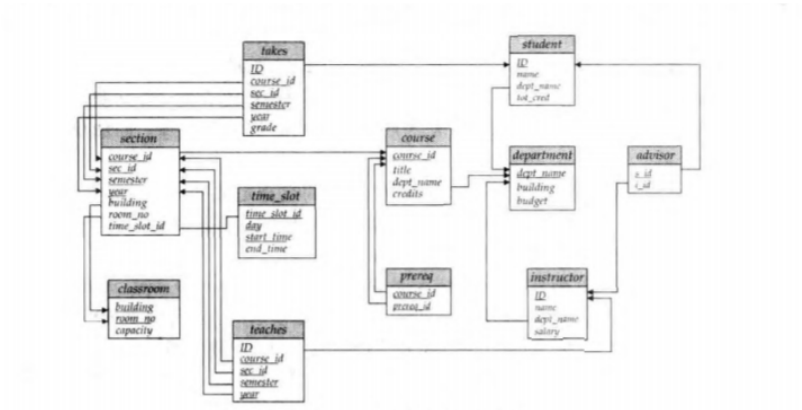

图 2-8　大学数据库的模式图

  - 主码选取要慎重，且习惯上把一个关系模式的主码属性列在其他属性前面，主码属性还加上了下划线。

在后面的章节中我们使用大学作为例子。图 2-9 给出了我们在例子中使用的关系模式，其中主码属性被标上了下划线。正如我们将在第 3 章中看到的一样，这对应于在 SQL 的数据定义语言中定义关系的方法。

classroom(*building*, *room_number*, *capacity*)
department(*dept_name*, *building*, *budget*)
course(*course_id*, *title*, *dept_name*, *credits*)
instructor(*ID*, *name*, *dept_name*, *salary*)
section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)
teaches(*ID*, *course_id*, *sec_id*, *semester*, *year*)
student(*ID*, *name*, *dept_name*, *tot_cred*)
takes(*ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*)
advisor(*s_ID*, *i_ID*)
time_slot(*time_slot_id*, *day*, *start_time*, *end_time*)
prereq(*course_id*, *prereq_id*)

图 2-9　大学数据库模式

- **2.5 关系查询语言**（详见第六章）
- **2.6 关系运算**（详见第六章）
  - 所有的过程化关系查询语言都提供了一组运算，这些运算要么施加于单个关系上，要么施加于一对关系上。这些运算具有一个很好的，并且也是所需的性质：运算结果总是单

个的关系。这个性质使得人们可以模块化的方式来组合几种这样的运算。特别是，由于关系查询的结果本身也是关系，所以关系运算可施加到查询结果上，正如施加到给定关系集上一样。

- 常见的两种运算：选出满足特定谓词的元组（行） or 选出特定属性（列）

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

图 2-1　*instructor* 关系

最常用的关系运算是从单个关系（如 *instructor*）中选出满足一些特定谓词（如 *salary* > 85 000 美元）的特殊元组。其结果是一个新关系，它是原始关系（*instructor*）的一个子集。例如，如果我们从图 2-1 的 *instructor* 关系中选择满足谓词“工资大于 85 000 美元”的元组，我们得到的结果如图 2-10 所示。

另一个常用的运算是从一个关系中选出特定的属性（列）。其结果是一个只包含那些被选择属性的新关系。例如，假设我们从图 2-1 的 *instructor* 关系中只希望列出教师的 *ID* 和工资，但不列出 *name* 和 *dept_name* 的值，那么其结果有 *ID* 和 *salary* 两个属性，如图 2-11 所示。结果中的每个元组都是从 *instructor* 关系中的某个元组导出的，不过只具有被选中的属性。

47 / 48

ID	*name*	*dept_name*	*salary*
12121	Wu	Finance	90000
22222	Einstein	Physics	95000
33456	Gold	Physics	87000
83821	Brandt	Comp. Sci.	92000

图 2-10　选择工资大于 85 000 美元的 *instructor* 元组的查询结果

ID	*salary*
10101	65000
12121	90000
15151	40000
22222	95000
32343	60000
33456	87000
45565	75000
58583	62000
76543	80000
76766	72000
83821	92000
98345	80000

图 2-11　从 *instructor* 关系中选取属性 *ID* 和 *salary* 的查询结果

- 连接运算可以通过下述方式来结合两个关系：把分别来自两个关系的元组对合并成单个元组。有几种不同的方式来对关系进行连接（正如我们将在第3章中看到的）。图2- 12显示了一个连接来自instructor和如artmem表中元组的例子，新元组给出了有关每个教师及

其工作所在系的信息。此结果是通过把instructor关系中的每个元组和department关系中对应于教师所在系的元组合并形成的。

ID	name	salary	dept_name	building	budget
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
12121	Wu	90000	Finance	Painter	120000
15151	Mozart	40000	Music	Packard	80000
22222	Einstein	95000	Physics	Watson	70000
32343	El Said	60000	History	Painter	50000
33456	Gold	87000	Physics	Watson	70000
45565	Katz	75000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
76543	Singh	80000	Finance	Painter	120000
76766	Crick	72000	Biology	Watson	90000
83821	Brandt	92000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000

图 2-12 *instructor* 关系和 *department* 关系的自然连接结果

- 笛卡儿积运算从两个关系中合并元组，但不同于连接运算的是，其结果包含来自两个关系元组的所有对，无论它们的属性值是否匹配。
-

关系代数

关系代数定义了在关系上的一组运算，对应于作用在数字上的普通代数运算，如加法、减法或乘法。正如作用在数字上的代数运算以一个或多个数字作为输入，返回一个数字作为输出，关系代数运算通常以一个或两个关系作为输入，返回一个关系作为输出。

第 6 章将详细介绍关系代数，下面我们给出几个运算的概述：

符号(名字)	使用示例
σ(选择)	$\sigma_{salary >= 85000}(instructor)$
	返回输入关系中满足谓词的行
Π(投影)	$\Pi_{ID, salary}(instructor)$
	对输入关系的所有行输出指定的属性。从输出中去除重复元组
⋈(自然连接)	*instructor* ⋈ *department*
	从两个输入关系中输出这样的元组对：它们在具有相同名字的所有属性上取值相同
×(笛卡儿积)	*instructor* × *department*
	从两个输入关系中输出所有的元组对(无论它们在共同属性上的取值是否相同)
∪(并)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	输出两个输入关系中元组的并

- **2.7 总结**
  - •关系数据模型（relational data model）建立在表的集合的基础上。数据库系统的用户可以对这些表进行查询，可以插入新元组、删除元组以及更新（修改）元组。表达这些操作的语言有几种.
  - •关系的模式（schema）是指它的逻辑设计，而关系的实例（instance）是指它在特定时刻的内容。数据库的模式和实例的定义是类似的。关系的模式包括它的属性，还可能包括属性类型和关系上的约束， 比如主码和外码约束。
  - •关系的超码(superkey)是一个或多个属性的集合，这些属性上的取值保证可以唯一识别出关系中的元组。候选码是一个最小的超码，也就是说，它是一组构成超码的属性集，但这组属性的任意子集都不是超码。关系的一个候选码被选作主码(primary key) 。
  - •在参照关系中的外码(foreign key)是这样的一个属性集合：对于参照关系中的每个元组来说，它在外码属性上的取值肯定等于被参照关系中某个元组在主码上的取值。
  - •模式图(schema diagram)是数据库中模式的图形化表示，它显示了数据库中的关系，关系的属性、主码和外码。
  - •关系查询语言(relational query language)定义了一组运算集，这些运算可作用于表上，并输出表作为结果。这些运算可以组合成表达式，表达所需的查询。

> - •关系代数(relational algebra)提供了一组运算.它们以一个或多个关系为输入，返回一个关系作为输出。诸如SQL这样的实际查询语言是基于关系代数的•但增加了一些有用的句法特征。