

哈工大DB-第3讲关系模型--基本概念

哈工大DB-第3讲关系模型--基本概念

0.本讲学什么

1.关系模型概述

1.1关系与关系模型

1.2关系模型的三要素

1.3关系模型的运算

关系代数--集合的运算

元组演算

域演算

2.什么是关系

2.1表的基本构成

2.2表格的构成的定义

2.2.1域(Domain)--列的取值范围

2.2.2元组(tuple)--笛卡尔积的一个元素

2.2.3关系与关系模式

2.3关系的特性

2.4关系的一些重要概念

2.4.1候选码/候选键--能唯一标识一个元组

2.4.2主码/主键--多个候选码中选一个作为主码

2.4.3主属性与非主属性--包含在候选码中的属性是主属性

2.4.4 外码/外键--某个关系非主属性中的别的关系的候选码

3.关模型中的三个完整性约束

3.1实体完整性--主码属性不能为空

3.2参照完整性--外键的值域必须与被参照的关系的主码的值域相同

3.3用户自定义完整性

0.本讲学什么

- 关系模型概述?
- 2. 什么是关系?
- 3. 关系模型中的完整性约束

重点与难点

- 一组概念的区分：围绕关系的相关概念
 - 域、笛卡尔积
 - 关系，关系模式
 - 关键字/键/码，外码/外键，主码/主键，主属性与非主属性。
- 三个完整性：实体完整性，参照完整性和用户自定义的完整性；

1.关系模型概述

1.1 关系与关系模型

- 一个关系(relation)就是一个Table
- 关系模型就是处理Table的，它由三个部分组成：
 - 描述DB各种数据的基本结构形式(Table/Relation)
 - 描述Table与Table之间所可能发生各种操作(关系运算)
 - 描述这些操作所应遵循的约束条件(完整性约束)
- 就是要学习: Table如何描述，有哪些操作、结果是什么、有哪些约束等?

1.2 关系模型的三要素

- 基本结构--关系/表格
- 基本操作

基本的: \cup (并, UNION)、 $-$ (差, DIFFERENCE)、 \times (广义积, PRODUCT)、 σ (选择, SELECTION)、 π (投影, PROJECTION)。
扩展的: \cap (交, INTERSECTION)、 \bowtie (连接, JOIN)、 \div (除, DIVISION) 运算

- 完整性约束
 - 实体完整性、参照完整性和用户自定义的完整性

1.3 关系模型的运算

关系代数--集合的运算

- 操作的对象和结构都是集合

➤ 即：操作的对象及结果都是集合，是一次一集合(Set-at-a-time)的操作。而非关系型的数据操作通常是一次一记录(Record-at-a-time)的操作

$$\pi_{\text{姓名, 课程名}} (\sigma_{\text{课程号}=\text{c2}} (R \bowtie S))$$

➤ 基于关系代数设计的数据库语言(ISBL): 用计算机可识别的符号表征关系代数的运算符号

$((R * S) : \text{课程号}=\text{c2}) \% \text{姓名, 课程名}$

R:F表示选择运算，R%表示投影运算

元组演算

➤ 元组演算示例：基于逻辑的运算

$$\{ t \mid (\exists u)(R(t) \wedge W(u) \wedge t[3] < u[1]) \}$$

➤ 基于元组演算设计的数据库语言(Ingres系统的QUEL): 用计算机可识别的符号表征元组演算的运算符号

range of t is R
range of u is W
retrieve t
where t.sage < u.sage

域演算

➤ 域演算示例：基于示例的运算

$\{ t_1, t_2, t_3 \mid S(t_1, t_2, t_3) \wedge R(t_1, t_2, t_3) \wedge t_1 < 20 \wedge t_2 > 30 \}$

➤ 基于域演算设计的数据库语言示例：(QBE: Query By Example)

S	S#	Sname	Sage	Sclass	Sex
	<u>S1</u>	P.X			

C	C#	Cname	Cformat	Cteacher	Coffice
	<u>C1</u>	计算机原理			

SC	S#	C#	Semester	Scgrade
	<u>S1</u>	<u>C1</u>		P.Y

2.什么是关系

2.1表的基本构成

家庭		
丈夫	妻子	子女
李基	王方	李键
张鹏	刘玉	张睿
张鹏	刘玉	张峰

2.2表格的构成的定义

2.2.1域(Domain)--列的取值范围

域：

- 一组值的集合，这组值具有相同的数据类型
- 集合中元素的个数称为域的基数(Cardinality)
- 如整数的集合、字符串的集合、全体学生的集合;再如, 由8位数字组成的数字串的集合，由0到100组成的整数集合

2.2.2元组(tuple)--笛卡尔积的一个元素

- 一组域 D_1, D_2, \dots, D_n 的笛卡尔积为: $D_1 \times D_2 \times \dots \times D_n = \{ (d_1, d_2, \dots, d_n) \mid d_i \in D_i, i=1, \dots, n \}$
- 笛卡尔积的每个元素 (d_1, d_2, \dots, d_n) 称作一个n-元组 (n-tuple)
- 元组 (d_1, d_2, \dots, d_n) 的每一个值 d_i 叫做一个分量(component)
- 元组 (d_1, d_2, \dots, d_n) 是从每一个域任取一个值所形成的一种组合，笛卡尔积是所有这种可能组合的集合，即：笛卡尔积是由n个域形成的所有可能的n-元组的集合
- 若 D_i 的基数为 m_i ，则笛卡尔积的基数，即元组个数为 $m_1 * m_2 * \dots * m_n$

2.2.3 关系与关系模式

关系(Relation)

- 一组域 D_1, D_2, \dots, D_n 的笛卡尔积的子集
- 笛卡尔积中具有某一方面意义的那些元组被称作一个关系(Relation)
- 由于关系的不同列可能来自同一个域，为区分，需要为每一列起一个名字，该名字即为属性名。
- 关系可用 $R(A_1 : D_1, A_2 : D_2, \dots, A_n : D_n)$ 表示，可简记为 $R(A_1, A_2, \dots, A_n)$ ，这种描述又被称为关系模式(Schema)或表标题(head)
 - R 是关系的名字, A_i 是属性, D_i 是属性所对应的域, n 是关系的度或目(degree), 关系中元组的数目称为关系的基数(Cardinality)
- 例如下图的关系为一3目关系，描述为家庭(丈夫:男人, 妻子:女人, 子女:儿童)或家庭(丈夫, 妻子, 子女)

◦

丈夫	妻子	子女
李基	王方	李健
张鹏	刘玉	张睿
张鹏	刘玉	张峰

- 关系模式 $R(A_1:D_1, A_2:D_2, \dots, A_n:D_n)$ 中属性向域的映象在很多DBMS中一般直接说明为属性的类型、长度等
 - 例如：
 - Student(S# char(8), Sname char(10), Ssex char(2), Sage integer, D# char(2), Sclass char(6))
 - Course (C# char(3), Cname char(12), Chours integer, Credit float(1), T# char(3))
 - SC(S# char(8), C# char(3), Grade float(1))

关系模式与关系

- 同一关系模式下，可有很多的关系
- 关系模式是关系的结构, 关系是关系模式在某一时刻的数据
- 关系模式是稳定的；而关系是某一时刻的值，是随时间可能变化的
- 可以理解为，关系是关系模式的一个实例

2.3 关系的特性

- **列的同质性**：即每一列中的分量来自同一域，是同一类型的数据
- **属性名的唯一性**：不同的列可来自同一个域，称其中的每一列为一个属性，不同的属性要给予不同的属性名。关系模式 $R(A_1:D_1, A_2:D_2, \dots, A_n:D_n)$ 中， $A_i (i = 1, \dots, n)$ 必须是不同的, 而 $D_i (i = 1, \dots, n)$ 可以是相同的
- **列位置互换性**：区分哪一列是靠列名
- **行位置互换性**：区分哪一行是靠某一或某几列的值(关键字/键字/码字)
- **位置无关性**：关系是以内容(名字或值)来区分的，而不是属性在关系的位置来区分
 - 如下面两个关系是完全相同的关系

家庭		
丈夫	妻子	子女
李基	王方	李键
张鹏	刘玉	张睿
张鹏	刘玉	张峰

家庭		
丈夫	子女	妻子
李基	李键	王方
张鹏	张峰	刘玉
张鹏	张睿	刘玉

- **元组可能相同**：理论上，关系的任意两个元组不能完全相同。(集合的要求：集合内不能有相同的两个元素)；现实应用中，表(Table)可能并不完全遵守此特性。
 - 元组相同是指两个元组的每个分量都相同。
- **关系第一范式-属性不可再分性**：即每一列不能再拆成小列，而每一行也不能在同一个属性中拥有多个值。

Students						
sid	name		class	telephone	enrollment	
	lname	fname			cno	major
1	Jones	Allan	2	555-1234	101	No
					108	Yes
2	Smith	John	3	555-4321	105	No
3	Borwn	Harry	2	555-1122	101	Yes
					108	No
4	White	Edward	3	555-3344	102	No
					105	No

不符合第一范式
(Not Table)

Head: structured type

Value: structured value
collection of values

2.4关系的一些重要概念

2.4.1候选码/候选键--能唯一标识一个元组

候选码(Candidate Key)/候选键

- **关系中的一个属性组，其值能唯一标识一个元组**，若从该属性组中去掉任何一个属性，它就不具有这一性质了，这样的属性组称作候选码。
 - 例如：“学生(S#(学号), Sname, Sage, Sclass)”，S#就是一个候选码，在此关系中，任何两个元组的S#是一定不同的，而这两个元组的Sname, Sage, Sclass都可能相同(同名、同龄、同班)，所以S#是候选码。
 - 再如：“选课(S#(学号), C#(课程号), Sname, Cname, Grade)”，(S#,C#)联合起来是一个候选码
- 因此，**候选码可以是一个属性，也可以是一组属性**
- **关系中可以同时存在多组候选码**，比如员工号，或者员工名字+员工电话号码

2.4.2主码/主键--多个候选码中选一个作为主码

主码(Primary Key)/主键

- **当有多个候选码时，可以选定一个作为主码。**
- DBMS以主码为主要线索管理关系中的各个元组。
 - 例如可选定属性S#作为“学生”表的主码，也可以选定属性组(Sname, Saddress)作为“学生”表的主码。
 - 选定EmpID为Employee的主码。

2.4.3主属性与非主属性--包含在候选码中的属性是主属性

主属性与非主属性

- **包含在任何一个候选码中的属性被称作主属性**，而其他属性被称作非主属性
 - 如“选课”中的S#，C#为主属性，而Sname, Cname, Grade则为主属性；

- 最简单的，候选码只包含一个属性
- 最极端的，所有属性构成这个关系的候选码，称为全码(All-Key)。
 - 比如：关系“教师授课”(T#,C#)中的候选码(T#,C#)就是全码。

2.4.4 外码/外键--某个关系非主属性中的别的关系的候选码

外码(Foreign Key)/外键

- 关系R中的一个属性组，它不是R的候选码，但它与另一个关系S的候选码相对应，则称这个属性组为R的外码或外键。
 - 例如“合同”关系中的客户号不是候选码，但却是外码。因它与“客户”关系中的候选码“客户号”相对应。
- 两个关系通常是靠外码连接起来的。

3.关模型中的三个完整性约束

3.1实体完整性--主码属性不能为空

实体完整性

- 关系的主码中的属性值不能为空值；
- 空值：不知道或无意义的值；
- 意义：关系中的元组对应到现实世界相互之间可区分的一个个个体，这些个体是通过主码来唯一标识的；若主码为空，则出现不可标识的个体，这是不容许的。

空值的额外说明：

空值的含义

- 空值：不知道、不存在或无意义的值；
- 在进行关系操作时，有时关系中的某属性值在当前是填不上的，比如档案中有“生日不详”、“下落不明”、“日程尚待公布”等，这时就需要空值来代表这种情况。
- 关系模型中用‘?’表征
- 数据库中有空值，会影响许多方面，如影响聚集函数运算的正确性，不能参与算术、比较或逻辑运算等
- 有空值的时候是需要特殊处理的，要特别注意。

3.2参照完整性--外键的值域必须与被参照的关系的主码的值域相同

参照完整性

- 如果关系R1的外码Fk与关系R2的主码Pk相对应，则R1中的每一个元组的Fk值或者等于R2 中某个元组的Pk 值，或者为空值
- 意义：如果关系R1的某个元组t1参照了关系R2的某个元组t2，则t2必须存在
- 例如关系Student在D#上的取值有两种可能：
 - 空值，表示该学生尚未分到任何系中
 - 若非空值，则必须是Dept关系中某个元组的D#值，表示该学生不可能分到一个不存在的系中
- 即，外键的值要么处在被参照的关系的主码的值域中，要么就是为空值，否则参照就是无意义的

3.3用户自定义完整性

用户自定义完整性

- 用户针对具体的应用环境定义的完整性约束条件
 - 如S#要求是10位整数，其中前四位为年度，当前年度与他们的差必须在4以内

DBMS对关系完整性的支持

- 实体完整性和参照完整性由DBMS系统自动支持
- DBMS系统通常提供了如下机制：
 - (1)它使用户可以自行定义有关的完整性约束条件
 - (2)当有更新操作发生时，DBMS将自动按照完整性约束条件检验更新操作的正确性，即是否符合用户自定义的完整性