第2章 关系数据库授课

重难点回顾

关系数据模型

关系数据模型是关系数据库管理系统的模型基础——是DBMS管理数据的框架，它基于数学的关系理论构建的，用来表示和管理数据及其关系

关系数据结构

关系数据结构：**关系数据模型** 中用于表示和组织数据的核心元素

关系就是一张二维表

- 行 为 元组，一行就是一条记录
- 列 为 属性，表示对象的性质

举个例子：

Example (关系)

Student

PH-001	Nick	M	20	Physics
CS-001	Elsa	F	19	CS
CS-002	Ed	M	19	CS
MA-001	Abby	F	18	Math
MA-002	Cindy	F	19	Math

重点：关系的键

键：关系的某些属性集合具有区分不同元组的作用，称为键

- 超键：关系的某一组属性的值能唯一标识每个元组，则称该组属性为超键

  举个例子：

  Example (超键)

  SC

Sno	Cno	Grade
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88
CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
MA-001	1002	

  属性集合{Sno, Cno}和{Sno, Cno, Grade}都是关系SC的超键

- 候选键：极小的超键。如果一个超键的任意真子集都不是超键，则称该超键为候选键

  候选键中去除其中的一个属性，则无法唯一标识每个元组

举个例子：

Example (候选键)

SC

Sno	Cno	Grade
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88
CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
MA-001	1002	

{Sno}和{Cno}都不是SC的超键，故{Sno, Cno}是SC的候选键

这节课对候选键没有什么讲述，但是你将会在第11章，第三范式的定义中再次看到它，不要忘了它

- **主键**：指定一个候选键作为主键，用于区分元组，避免插入重复的元组。
- **外键**：不同关系中的元组可以存在联系，这种联系是“参照与被参照”

  举个例子：从表Student中获得学生学号Sno，然后到表SC中找到学号为Sno的学生的课程和成绩信息。此时建立了关系SC和关系Student的参照联系——**SC是参照关系，Student是被参照关系，SC.Sno参照了Student.Sno**

Example (外键)

SC

Sno	Cno	Grade
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88
CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
MA-001	1002	

Student

Sno	Sname	Ssex	Sage	Sdept
PH-001	Nick	M	20	Physics
CS-001	Elsa	F	19	CS
CS-002	Ed	M	19	CS
MA-001	Abby	F	18	Math
MA-002	Cindy	F	19	

外键的定义：外键是一个表中的一个或多个字段，它参照另一个表中的主键（或候选键），建立起两者之间的联系。实际上，**外键是不同关系之间联系的纽带**

这个例子中，SC.Sno是SC的外键，它参照Student.Sno

关系数据结构就是这么简单清晰，这样我们就搭建出了关系数据库管理系统的基本框架，能够组织和存储数据了

关系完整性约束

关系完整性约束：关系数据库中的所有数据必须满足的约束条件，它实际上为我们定义了一组规范，保证了数据的准确性和一致性

- 实体完整性约束——限制在一个表上的约束

  关系中任意元组的主键值必须唯一

  引申：关系中任意元组在主键中的属性值非空，因为空值无法判断是否重复。在DBMS中，空值表示值不存在，它既不是0，也不是空串

举个例子：

Example (实体完整性约束)

SC

Sno	Cno	Grade
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88
CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
MA-001	1002	

- {Sno, Cno}是SC的主键
- 所有元组的Sno和Cno属性值组合必须唯一 演示
- 任意元组的Sno和Cno属性值必须非空 演示

Sno和Cno属性值组合必须唯一，且所有元组中这两个属性也必须均非空，对属性Grade没有这个要求（它不是主键），上面的表是正确的

- 参照完整性约束——保证两个有关系的表之间的参照关系不出错

  外键的值必须来自其参照的属性的值（可以用两个表演示，这里说的是 外键的属性值集合 包含于 外键参照的属性的属性值集合）

  外键的值可以为空

  更规范的描述：

参照完整性约束规则

设F是关系R的外键，F参照关系S的主键，则R中任意元组的F属性值必须满足以下两个条件之一：

1. F的值为空
2. 若F的值不为空，则F的值必须在S中存在

举个例子：

Example (参照完整性约束)

SC 主键为Sno+Cno

Sno	Cno	Grade
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88
CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
MA-001	1002	

Student

Sno	Sname	Ssex	Sage	Sdept
PH-001	Nick	M	20	Physics
CS-001	Elsa	F	19	CS
CS-002	Ed	M	19	CS
MA-001	Abby	F	18	Math
MA-002	Cindy	F	19	

- SC.Sno和Student.Sno分别表示SC和Student的属性Sno
- SC.Sno是SC的外键，它参照Student.Sno
- SC.Sno的属性值集合必须是Student.Sno属性值集合的子集 演示

这个例子中，SC.Sno的值 一定能在 Student.Sno中找到，也就是说SC.Sno的属性值集合 必须是Student.Sno属性值集合的子集

Example (参照完整性约束)

Student

Sno	Sname	Ssex	Sage	Sdept
PH-001	Nick	M	20	Physics
CS-001	Elsa	F	19	CS
CS-002	Ed	M	19	CS
MA-001	Abby	F	18	Math
MA-002	Cindy	F	19	

Department

Dept	Addr
Physics	B1
CS	B2
Math	B3

- Sdept是Student的外键，它参照Department的Dept属性
- Student中元组的Sdept属性值可以为空，表示该学生的院系未知
- 如果Student中元组的Sdept属性值非空，则该Sdept属性值必须属于Department中Dept的属性值集合

这个例子说明了外键可以为空：Student.Sdept属性值为空，表示该学生的院系未知

- 用户定义完整性约束——应用需求的约束，落实到具体的应用情景

  比如性别只能是男或女，年龄>0，这种约束来自于应用，与关系模型无关

  举个例子：

Example (用户定义完整性约束)

Student

Sno	Sname	Ssex	Sage	Sdept
PH-001	Nick	M	20	Physics
CS-001	Elsa	F	19	CS
CS-002	Ed	M	19	CS
MA-001	Abby	F	18	Math
MA-002	Cindy	F	19	Math

- Student.Sname不可以为空
- Student.Ssex的值只能是'M'或'F'
- Student.Sage的值必须大于0 演示

关系操作

关系代数表达式明确给出了查询的执行过程，操作数是关系，结果也是关系。DBMS在执行SQL语句时，会将它们转换为关系代数表达式，然后根据这些表达式来执行查询

由于后面有题目的讲解，这部分就大致复习一遍

基本关系代数操作

- 选择σ

  语法：$\sigma_{条件}(R)$

  从一个关系中选出满足给定条件的元组

- 投影Π

语法：$\Pi_{投影属性列表}(R)$

从一个关系中选出指定的列，并去掉重复元组（注意，它是去重的）

- 并$\cup$

  语法：$R \cup S$

  计算关系R和S的并集

  要求R和S必须相容：

  - R、S的属性个数一样多（对齐）
  - 对应的属性必须有相容（不必相同）的类型（整型和浮点型可以，整型和字符型不可以）
  - 有相似的语义（即使体重和身高是相同的数据类型，但是二者语义不同，所以也不是相同的）

- 差$-$

  语法：$R - S$

  计算关系R和S的差集

  同样的，要求R和S相容

- 笛卡尔积$\times$

  语法：$R \times S$

  计算两个关系的笛卡尔积，实际上是将R和S中的元组无条件地连接起来，最终的结果是一个$m \times n$的关系（R的元组数为m，S的元组数为n）

- 重命名ρ

  语法：

  - 修改单个属性名：$\rho_{新属性名<-原属性名}(R)$
  - 修改关系名：$\rho_S(R)$，将关系R更名为S
  - 修改关系名和它的所有属性名：$\rho_{S(A_1,A_2,...,A_n)}(R)$，将关系R更名为S，并将R的全部属性名更名为A1,...,An

  重命名用于修改关系名和属性名，当需要把一个关系和它自身进行连接，也就是自连接时，重命名操作可以区分这个关系的两个副本

  举个例子：现有Student表，记录学号，姓名，性别，年龄，所在的院系

Student

Sno	Sname	Ssex	Sage	Sdept
PH-001	Nick	M	20	Physics
CS-001	Elsa	F	19	CS
CS-002	Ed	M	19	CS
MA-001	Abby	F	18	Math
MA-002	Cindy	F	19	Math

  找出和Elsa在同一个系学习的学生的学号和姓名（使用自连接）：

  - 将这个表“复制”一份，命名为S2，原表命名为S1，用来区分两个副本：$\rho_{S1}(Student)$，$\rho_{S2}(Student)$
  - 对S1和S2做笛卡尔积进行连接：$\rho_{S1}(Student) \times \rho_{S2}(Student)$

    获得这样的表：（这里别忘了插图）

- 但这样的表是无序的，没有任何意义。自连接中，一个表作为参照，通过条件从另一个表中筛选出结果。在本题中，我们需要找到和Elsa所在系相同的学生，则S1作为参照，首先选出所有$S1.name =' Elsa'$的元组：$\sigma_{S_1.name='Elsa'}$，此时S1中的所有记录都是'Elsa'的了，则在S2中寻找与'Elsa'所在系相同的学生，等价于 从S2中寻找$S2.Sdept = S1.Sdept$的元组：$\sigma_{S1.Sdept=S2.Sdept}$。因此，在表S1和S2做笛卡尔积后，再做选择操作：$\sigma_{S_1.name='Elsa' \wedge S1.Sdept=S2.Sdept}(\rho_{S1}(Student) \times \rho_{S2}(Student))$.这样，我们就在S2中获得了和'Elsa'在同一个系的学生的元组
- 题目要求获得学生的学号的姓名，则进行投影：$\Pi_{S2.Sno,S2.Sname}$

整理一下，这个语句为：

$$\Pi_{S2.Sno,S2.Sname}(\sigma_{S_1.name='Elsa' \wedge S1.Sdept=S2.Sdept}(\rho_{S1}(Student) \times \rho_{S2}(Student)))$$

这就是基本关系代数操作，这些操作经过组合，可以实现复杂的功能。比如笛卡尔积和选择结合，可以实现连接。但是只用基本关系代数操作来编写复杂查询是非常繁琐的，因此要引入**派生关系代数操作**来简化查询编写

派生关系代数操作

任何一项派生关系代数操作都可以用基本关系代数操作来表示。派生关系不仅仅简化了查询编写，它们还反映了数据库查询中一些常见的，高效的模式，而且这些派生关系代数操作 在查询执行时有特定的实现和优化机制

- 交∩

  语法：$R \cap S$

  计算关系R和S的交集

  要求R和S相容

- 内连接：结果中只包含R和S中满足连接条件的元组，R和S不满足连接条件的元组均不会出现在连接结果中
  - θ连接$\bowtie_\theta$

    语法：$R \bowtie_\theta S$

    将关系R和S中满足连接条件θ的元组进行连接，连接结果中包含R和S中的全部属性

    举例：连接关系Student和SC：$Student \bowtie_{Student.Sno=SC.Sno} SC$

Example (θ连接)

Student

Sno	Sname	Ssex	Sage	Sdept
PH-001	Nick	M	20	Physics
CS-001	Elsa	F	19	CS
CS-002	Ed	M	19	CS
MA-001	Abby	F	18	Math
MA-002	Cindy	F	19	Math

SC

Sno	Cno	Grade
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88
CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
MA-001	1002	

$Student \bowtie_{Student.Sno=SC.Sno} SC$ 演示

Student.Sno	Sname	Ssex	Sage	Sdept	SC.Sno	Cno	Grade
PH-001	Nick	M	20	Physics	PH-001	1002	92
PH-001	Nick	M	20	Physics	PH-001	2003	85
PH-001	Nick	M	20	Physics	PH-001	3006	88
CS-001	Elsa	F	19	CS	CS-001	1002	95
CS-001	Elsa	F	19	CS	CS-001	3006	90
CS-002	Ed	M	19	CS	CS-002	3006	80
MA-001	Abby	F	18	Math	MA-001	1002	

实际上，结合笛卡尔积和选择操作可以实现θ连接

- 等值连接

  连接条件θ仅涉及相等比较的θ连接，上面的例子就是一个等值连接

- 自然连接$\bowtie$

  语法：$R \bowtie S$

  连接条件：R和S中**同名属性**进行等值连接。如果R和S有多个同名属性，则这些属性全部进行等值连接。在结果中，同名属性只保留一份

  举个例子：

  $Student \bowtie SC$

Example (自然连接)

Student

Sno	Sname	Ssex	Sage	Sdept
PH-001	Nick	M	20	Physics
CS-001	Elsa	F	19	CS
CS-002	Ed	M	19	CS
MA-001	Abby	F	18	Math
MA-002	Cindy	F	19	Math

SC

Sno	Cno	Grade
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88
CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
MA-001	1002	

$Student \bowtie SC$ 演示

Sno	Sname	Ssex	Sage	Sdept	Cno	Grade
PH-001	Nick	M	20	Physics	1002	92
PH-001	Nick	M	20	Physics	2003	85
PH-001	Nick	M	20	Physics	3006	88
CS-001	Elsa	F	19	CS	1002	95
CS-001	Elsa	F	19	CS	3006	90
CS-002	Ed	M	19	CS	3006	80
MA-001	Abby	F	18	Math	1002	

与等值连接做比较：

Example (θ连接)

Student

Sno	Sname	Ssex	Sage	Sdept
PH-001	Nick	M	20	Physics
CS-001	Elsa	F	19	CS
CS-002	Ed	M	19	CS
MA-001	Abby	F	18	Math
MA-002	Cindy	F	19	Math

SC

Sno	Cno	Grade
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88
CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
MA-001	1002	

$Student \bowtie_{Student.Sno=SC.Sno} SC$ 演示

Student.Sno	Sname	Ssex	Sage	Sdept	SC.Sno	Cno	Grade
PH-001	Nick	M	20	Physics	PH-001	1002	92
PH-001	Nick	M	20	Physics	PH-001	2003	85
PH-001	Nick	M	20	Physics	PH-001	3006	88
CS-001	Elsa	F	19	CS	CS-001	1002	95
CS-001	Elsa	F	19	CS	CS-001	3006	90
CS-002	Ed	M	19	CS	CS-002	3006	80
MA-001	Abby	F	18	Math	MA-001	1002	

可以看到，自然连接的结果中，同名属性只保留一份

一般来说，外键和它所参照的属性是同名的，在把两个表的信息进行拼接时，要将外键与它参照的属性进行等值连接。上面的例子就是这样的

对比一下自然连接与θ连接的区别：

	自然连接	θ连接
连接条件	隐含给出	明确给出
连接结果的属性	去除重复的同名属性	保留重复的同名属性

自然连接其实有一个坑，它会连接两个关系中的所有**同名属性**，也就是说，对于两个表中相同名称的属性，即使两个表中各自属性代表的含义不同，自然连接也会将这样的属性进行等值连接，获得的结果将是空的！

举个例子：Printer（打印机）和 Product（产品信息）

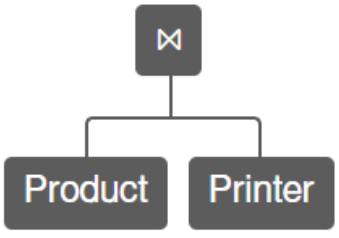
Printer

Printer.model	Printer.color	Printer.type	Printer.price
3001	true	ink-jet	99
3002	false	laser	239
3003	true	laser	899
3004	true	ink-jet	120
3005	false	laser	120
3006	true	ink-jet	100
3007	true	laser	200

Product

Product.maker	Product.model	Product.type
A	1001	pc
A	1002	pc
A	1003	pc
A	2004	laptop
A	2005	laptop
A	2006	laptop
B	1004	pc
B	1005	pc
B	1006	pc
B	2007	pc
C	1007	pc
D	1008	pc
D	1009	pc
D	1010	pc
D	3004	printer
D	3005	printer
E	1011	pc
E	1012	pc
E	1013	pc
E	2001	laptop

对这两个关系做自然连接：



Product ⋈ Printer

Product.maker	Product.model	Product.type	Printer.color	Printer.price

结果为空！这是因为Product.type和Printer.type在自然连接时做了等值连接，然而这两个属性的含义并不相通：Product.type是指产品的类型（Laptop, PC, Printer），而Printer.type是打印机的类型（ink-jet，laser），两个属性代表的含义不同，没有相同的属性值，而自然连接却对type属性做了等值连接，结果就是空的

自然连接写起来很方便，但是要注意上面这样的情况，考试中曾经挖过这样的坑

- 外连接：除了要在R和S的连接结果中保留满足连接条件的全部元组外，还需要在连接结果中保留R或S中的不满足连接条件的元组
  - 左外θ连接

    语法：$R⟕_{\theta}S$

    1. 将R和S中**满足连接条件**θ的元组进行连接，即计算$R \bowtie_{\theta} S$
    2. 保留R中不满足连接条件的元组，这些元组在右侧表中的列将填充为NULL

    Example (左外θ连接)

    ❶ 查询全体学生的选课情况(含未选课的学生)

    $Student ⟕_{Student.Sno=SC.Sno} SC$ 演示

    $Student ⟕_{Student.Sno=SC.Sno} SC$

Student.Sno	Sname	Ssex	Sage	Sdept	SC.Sno	Cno	Grade
PH-001	Nick	M	20	Physics	PH-001	1002	92
PH-001	Nick	M	20	Physics	PH-001	2003	85
PH-001	Nick	M	20	Physics	PH-001	3006	88
CS-001	Elsa	F	19	CS	CS-001	1002	95
CS-001	Elsa	F	19	CS	CS-001	3006	90
CS-002	Ed	M	19	CS	CS-002	3006	80
MA-001	Abby	F	18	Math	MA-001	1002	
MA-002	Cindy	F	19	Math			

  - 左外自然连接

    语法：$R⟕S$

    1. 将R和S中满足**自然连接条件**的元组进行连接，即计算$R \bowtie S$
    2. 保留R中不满足自然连接条件的元组，这些元组在右侧表中的列将填充为NULL

    可以看到，左外连接是在内连接的基础上，保留了左关系中不满足连接条件的元组。

    右外连接同理，在内连接的基础上保留了右关系中不满足连接条件的元组，这里就不再赘述了
  - 全外连接：左/右外连接的并集——全外连接的结果中包含左右关系中不满足连接条件的元组

扩展关系代数操作

扩展关系代数操作不能由基本关系代数操作组合实现，使用扩展关系代数操作是为了增强关系代数的查询表示能力。一个很关键的扩展关系代数操作是分组操作

分组操作γ

分组操作的功能（分组操作做了什么）：

- 分组：根据指定的分组属性，对一个关系中的元组进行分组，分组属性值相同的元组分到一个组中
- 聚集：对每个组中元组的非分组属性进行聚集，包括计数count，求最小值min，求最大值max，求和sum，有一系列聚集函数提供聚集操作

  注意：聚集函数只能对非空值，count(*)除外，它能计算分组内所有元组的数量

举个例子：

SC

Sno	Cno	Grade
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88
CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
MA-001	1002	

我们要统计SC中每名学生的选课数和平均分。明显地，SC这个表中同一个学生的信息是分散的，我们需要将同一个学生的信息汇聚在一起，因此要按照**Sno**属性进行分组，将Sno相同的元组归为一组，则每组都汇聚了同一个学生的信息。分组之后，使用count统计课程数量，使用average计算组内成绩的平均分. 最终的结果：

每名学生的选课数和平均分

Sno	Amount	AvgGrade
PH-001	3	88.3
CS-001	2	92.5
CS-002	1	80
MA-001	1	

聚集函数average计算非空值的平均值，如果所有值都是空的，则返回空。因此MA-001的平均成绩是空的

结合上面的描述，书写分组操作的语句：

语法：$\gamma_{L;agg}(R)$

- R：关系名
- L：分组属性列表，用逗号分隔
- agg：聚集函数表达式列表，用逗号分隔，每个聚集函数表达式形如sum(score) -> TotalScore（计算score属性值的和，并将结果命名为属性TotalScore，在分组操作的结果中你会看到这个属性）

在分组操作中，只有**分组属性**和**经过聚合的属性**才能在结果中出现，而非分组属性不能被直接投影

分组操作比较难理解，我们看几个例子：

1. 统计每个系的男生人数和女生人数，关系为Student

   首先，分组属性应该有两个：系 和 性别，第一个分组属性将同一个系的元组归为一组，第二个分组属性则将同一个系的同一个性别的元组归为一组。然后使用count(*)计算人数

$$\gamma_{Sdept,Ssex;count(*)->Amt}(Student)$$

SC

Sdept	Ssex	Amt
Physics	M	1
CS	F	1
CS	M	1
Math	F	2

2. 统计每名已选课学生的选课数和平均分

   分组属性为学生的学号Sno，将Sno相同的元组归为一组，这实际上是一个学号为Sno的学生的所有选课信息。然后用count(*)函数计算数量，用avg函数计算平均分：

$$\gamma_{Sno;count(*)->Amt,avg(Grade)->Score}(SC)$$

查询2的结果

Sno	Amt	Score
PH-001	3	88.3
CS-001	2	92.5
CS-002	1	80
MA-001	1	

至此，结束重难点回顾结束。详细讲述了关系完整性约束，派生关系代数操作和扩展关系代数操作，简略讲述了关系数据结构，基本关系代数操作。在授课时尽量控制在30min以内

下面是习题部分，摘取自HIT 2020年 数据库系统期末考试

习题讲解

题目1

已知如下关系数据库模式：

```
Student(sid, name, department, email)
Course(cid, title, credit)
Enroll(sid, cid, score)
```

- 关系Student记录学生信息，包括学号（sid）、姓名（name）、所在系（department）、电子邮件（email）。
- 关系Course记录课程信息，包括课号（cid）、课程名（title）、学分（credit）。
- 关系Enroll记录学生选课信息，包括学号（sid）、课号（cid）、成绩（score）。

问题1

查询“Elsa"选修过的课程的课号及名称

- 思路：
  - 连接：将Student和Enroll连接起来，能获得学生-选课信息，但是无法获得课程的信息（课程名称），所以还需要连接上Course获得课程名称，这样连接的结果是：学生-选课信息-选的这门课的课程信息
  - 选择：连接之后，用name = 'Elsa'进行选择，获得有关”Elsa“的元组
  - 投影：要求给出课号及名称，投影即可

```
Π cid, title (σ name = 'Elsa' (Student ⋈ Course ⋈ Enroll))
```

Student与Course没有同名属性，所以二者进行自然连接实际上是在做笛卡尔积，不过再和Enroll做自然连接时，对属性sid和cid进行了等值连接，最后的连接结果是正确的

问题2

查询所选课程的总学分低于120的学生的学号及所选课程的总学分

- 思路：
  - 连接：学分的信息在Course中，而选课信息在Enroll中，这又要进行连接，还是将三个关系进行连接
  - 分组：我们需要计算学生所选课程的总学分，这实际要求我们将一个学生的所有信息汇聚起来，然后计算总学分，也就是要按照sid进行分组，然后用sum计算总学分。分组后的关系有两个属性，即sid和sum的聚集属性
  - 选择：我们选出总学分<120的元组，作为最终结果

```
σ sum_credit < 120 (γ sid, sum(credit)->sum_credit (Student ⋈ Course ⋈ Enroll))
```

问题3

查询没有选修过"Database Systems"，却选修了"Data Mining"的学生的学号

- 思路：

  可以考虑用差操作，首先获得选修过“Database Systems”的学生的学号，然后获得选修过“Data Minig”的学生的学号，二者的差就是题目要找的

```
Π sid (σ title = 'Data Mining' (Student ⋈ Course ⋈ Enroll)) -
Π sid (σ title = 'Database Systems' (Student ⋈ Course ⋈ Enroll))
```

PPT上的题目答案在《数据库系统H1.md》中