

课程安排

周次	内容	
第1周(9月6日)	概述	
第2周(9月13日)	ER模型: 关系模型与SQL (1)	
第3周(9月20日)	关系模型与SQL (2)	
第4周(9月27日)	关系模型与SQL (3)	
第5周(9月29日)	关系模型与SQL (4)	
第6周(10月11日)	实验1: 数据库安装与使用 (独立完成)	5%
第7周(10月18日)	实验2: 数据库应用设计 (每组3人)	15%
第8周(10月25日)	数据存储与访问路径 (1)	
第9周(11月1日)	数据存储与访问路径 (2)	
第10周(11月8日)	数据存储与访问路径(3); 查询处理(1)	

前面的内容

- ER图
- 关系模型
- SQL 初步
 - 表的定义、增删改
- 逻辑设计: ER图到关系模型

关系模型复习：Table/Relation（表）

- 列(Column): 一个属性, 有明确的数据类型
 - 例如: 数值类型 (e.g., int, double), 字符串类型(varchar), 类别类型(有些像程序语言中的enum)
 - 必须是原子类型, 不能够再进一步分割, 没有内部结构
- 行(Row): 一个记录 (tuple, record)
 - 表是一个记录的集合
 - 记录之间是无序的
- 通常是一个很瘦长的表
 - 几列到几十列
 - 成千上万行, 很大的表可以有亿/兆行

表的数学定义

- K列的表: $\{ \langle t_1, \dots, t_k \rangle \mid t_1 \in D_1, \dots, t_k \in D_k \}$
 - 整个表是一个集合 $\{ \langle t_1, \dots, t_k \rangle \}$
 - 集合的每个元素有这样的形式 $\langle t_1, \dots, t_k \rangle$
 - 第j个部分 t_j
 - D_j 是第j列的类型所对应的所有可能取值的集合 (域)

举例：学生信息表

列数称为度 (arity 或 degree)
每一列是一个属性 (~10)

ID	Name	Birthday	Gender	Major	Year	GPA
131234	张飞	1995/1/1	男	计算机	2013	85
145678	貂蝉	1996/3/3	女	经管	2014	90
129012	孙权	1994/5/5	男	法律	2012	80
...
...
...
...
...
...
...

每一行是一个学生记录 (~10⁴)
记录的条数称为基(cardinality)

创建表：Create Table

Student

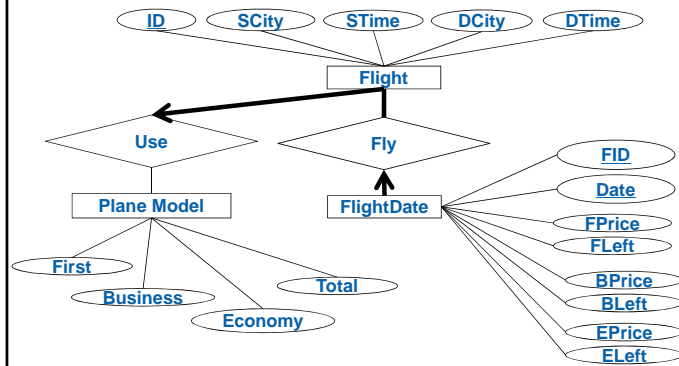
ID	Name	Birthday	Gender	Major	Year	GPA
...

```
create table Student (  
  ID integer,  
  Name varchar(20),  
  Birthday date,  
  Gender enum(M, F),  
  Major varchar(20),  
  Year year,  
  GPA float  
);
```

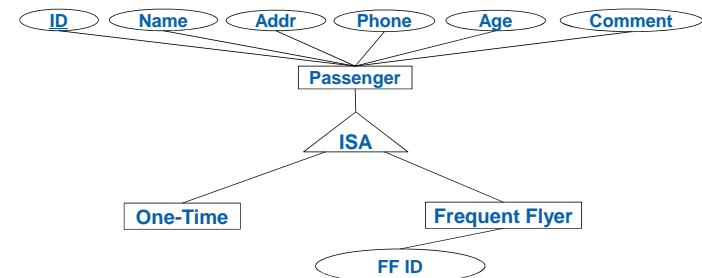
```
create table 表名 (  
  列名 类型,  
  列名 类型,  
  列名 类型,  
  .....  
);
```

有些像程序语言中的函数定义

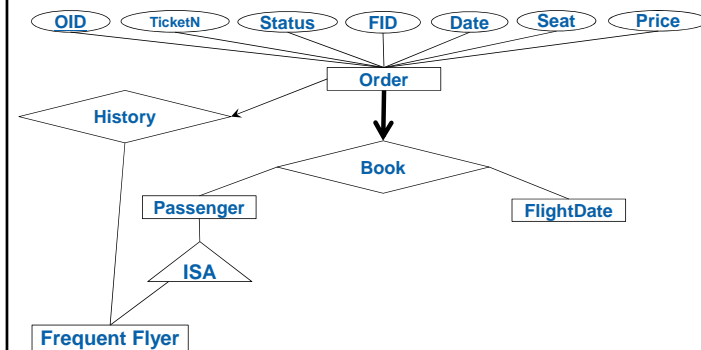
航班



乘客



订单



Outline

- ER图和关系模型复习
- SQL 初步
 - 记录的增删改
 - 简单的查询
- 关系代数
- 举例

表的定义

Student

ID	Name	Birthday	Gender	Major	Year	GPA
...

```
create table Student (  
    ID integer,  
    Name varchar(20),  
    Birthday date,  
    Gender enum(M, F),  
    Major varchar(20),  
    Year year,  
    GPA float  
);
```

```
create table 表名 (  
    列名 类型,  
    列名 类型,  
    列名 类型,  
    .....  
);  
  
有些像程序语言中的函数定义
```

Insert

Student

ID	Name	Birthday	Gender	Major	Year	GPA
131234	张飞	1995/1/1	M	计算机	2013	85

插入完整记录:

insert into 表名 values (值以逗号隔开, 按create table顺序);

insert into Student
values (131234, '张飞', 1995/1/1, M, '计算机', 2013, 85);

插入记录特定的列, 其它列为空NULL或默认值:

insert into 表名(列名1, 列名2, ...) values (列1值, 列2值, ...);

insert into Student(ID, Name) values (131234, '张飞');

Insert

把查询的结果插入一个表:

```
insert into 表名  
select 语句;
```

Select语句是查询语句, 本节后面将仔细讲解

Delete

Student

ID	Name	Birthday	Gender	Major	Year	GPA
131234	张飞	1995/1/1	M	计算机	2013	85

delete from 表名 where 条件;

删除上述记录:

delete from Student where ID = 131234;

删除所有计算机系且GPA小于60的学生记录:

delete from Student where Major = '计算机' and GPA < 60;

Update

Student

ID	Name	Birthday	Gender	Major	Year	GPA
131234	张飞	1995/1/1	M	计算机	2013	85

86

update 表名
set 列名 = 值
where 条件;

用新的GPA代替到上个学期为止的GPA:

update Student
set GPA = 86
where ID = 131234;

Outline

- 关系模型复习
- SQL 初步
 - 记录的增删改
 - 简单的查询
- 关系代数
- 举例

关系代数 (Relational Algebra)

• 什么是代数?

- $a+b+c*d$
- 用变量符号代替具体的数
- 用运算符连接表达运算

• 什么是关系代数?

- 在关系上定义的代数
- “变量”: 关系变量
- “数”: 关系实例, 即数据表
- “运算”: 关系运算

传统关系代数

• 我们在本节课中介绍传统关系代数

- 这是最早最经典的关系代数
- 可以解释很多查询的功能

• 传统关系代数的特征

- 关系: 看作是集合(Set)
 - 没有重复的记录

• 我们后面的课还会讲扩展关系代数

- 关系看作是多重集 (Multi-set), 允许重复的元素
- 可以解释更多的查询功能

传统关系代数的关系运算

- 集合操作：并、交、差
- 选择行或列：选择、投影
- 两个关系元组之间的操作：笛卡尔积、连接
- 其它：重命名、除

我们先讲最重要的三个关系运算

- Selection (选择) σ
- Projection (投影) π
- Join (连接) \bowtie

对应于SQL查询中

select 列名, ..., 列名

from 表名, ..., 表名

where 条件;

Selection (选择)

- 从一个表中提取一些行
- 关系代数形式：

$\sigma_{\text{条件}}(\text{表名})$

- 返回给定表中符合给定条件的行

Selection (选择)

$\sigma_{\text{Major}='计算机'}(\text{Student})$

- 从一个表中提取一些行

选择所有计算机系学生记录

ID	Name	Birthday	Gender	Major	Year	GPA
131234	张飞	1995/1/1	男	计算机	2013	85
145678	貂蝉	1996/3/3	女	经管	2014	90
129012	孙权	1994/5/5	男	法律	2012	80
121101	关羽	1994/6/6	男	计算机	2012	90
142233	赵云	1996/7/7	男	计算机	2014	95
...
...
...
...
...

用SQL语句来表达选择

Major='计算机' (Student)

Student

ID	Name	Birthday	Gender	Major	Year	GPA
131234	张飞	1995/1/1	男	计算机	2013	85
145678	貂蝉	1996/3/3	女	经管	2014	90
129012	孙权	1994/5/5	男	法律	2012	80
121101	关羽	1994/6/6	男	计算机	2012	90
142233	赵云	1996/7/7	男	计算机	2014	95

```
select *
from Student
where Major = '计算机';
```

```
select *
from 表名
where 条件;
```

*表示返回整个记录
多个条件可以用and, or, ()等组合

SQL表达选择

Major='计算机' (Student)

```
select *
from Student
where Major = '计算机';
```

输出结果

ID	Name	Birthday	Gender	Major	Year	GPA
131234	张飞	1995/1/1	男	计算机	2013	85
121101	关羽	1994/6/6	男	计算机	2012	90
142233	赵云	1996/7/7	男	计算机	2014	95

把不满足where条件的记录过滤掉了

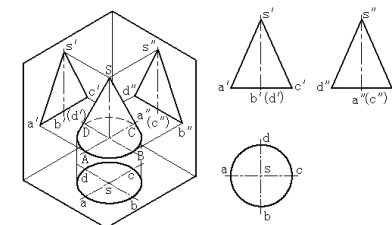
Projection (投影)

- 从一个表中提取一些列
- 关系代数形式:

$\pi_{\text{列名}, \dots, \text{列名}}(\text{表名})$

Projection (投影)

- 为什么叫投影?
 - 比照立体几何中, 立体向平面的投影
 - 投影是从高维向低维的映射, 是一种降维操作
 - 如果把每一列看作是一维, 那么关系代数的投影也是降维




立体向平面投影

Projection (投影)

$\pi_{Name, GPA}(Student)$

- 从一个表中提取一些列

提取学生姓名和平均分



ID	Name	Birthday	Gender	Major	Year	GPA
131234	张飞	1995/1/1	男	计算机	2013	85
145678	貂蝉	1996/3/3	女	经管	2014	90
129012	孙权	1994/5/5	男	法律	2012	80
121101	关羽	1994/6/6	男	计算机	2012	90
142233	赵云	1996/7/7	男	计算机	2014	95
...
...

用SQL语句来表达投影

$\pi_{Name, GPA}(Student)$

Student

ID	Name	Birthday	Gender	Major	Year	GPA
131234	张飞	1995/1/1	男	计算机	2013	85
145678	貂蝉	1996/3/3	女	经管	2014	90
129012	孙权	1994/5/5	男	法律	2012	80
121101	关羽	1994/6/6	男	计算机	2012	90
142233	赵云	1996/7/7	男	计算机	2014	95

select *Name, GPA*
from *Student*;

select 列名, ..., 列名
from 表名;

SQL表达投影

$\pi_{Name, GPA}(Student)$

select *Name, GPA*
from *Student*;

输出结果

Name	GPA
张飞	85
貂蝉	90
孙权	80
关羽	90
赵云	95

把没有在select分句中的列去掉了

SQL表达投影：结果有重复？

$\pi_{GPA}(Student)$

select *GPA*
from *Student*;

输出结果

GPA
85
90
80
90
95

注意：在数据库系统的实现中，select不会去重，这时的输出不再是一个集合，与传统关系代数的语义有差异

SQL表达投影+去重

$\pi_{GPA}(Student)$

```
select distinct GPA
from Student;
```

输出结果

GPA
85
90
80
95

注意: select默认不去重, 用distinct去重

为什么默认不去重?

- 去重有代价
- 重复数量可能有意义

Selection (选择)+ Projection (投影)

- 选择: 从一个表中提取一些行

$\sigma_{\text{条件}}(\text{表名})$

- 投影: 从一个表中提取一些列

$\pi_{\text{列名}, \dots, \text{列名}}(\text{表名})$

- 选择+投影

$\pi_{\text{列名}, \dots, \text{列名}}(\sigma_{\text{条件}}(\text{表名}))$

SQL表达选择+投影 $\pi_{\text{Name, GPA}}(\sigma_{\text{Major}='计算机'}(Student))$

Student

ID	Name	Birthday	Gender	Major	Year	GPA
131234	张飞	1995/1/1	男	计算机	2013	85
145678	貂蝉	1996/3/3	女	经管	2014	90
129012	孙权	1994/5/5	男	法律	2012	80
121101	关羽	1994/6/6	男	计算机	2012	90
142233	赵云	1996/7/7	男	计算机	2014	95

```
select Name, GPA
from Student
where Major = '计算机';
```

```
select 列名, ..., 列名
from 表名
where 条件;
```

SQL表达选择+投影 $\pi_{Name, GPA}(\sigma_{Major='计算机'}(Student))$

```
select Name, GPA
from Student
where Major = '计算机';
```

输出结果

Name	GPA
张飞	85
关羽	90
赵云	95

既过滤了记录，又提取了列

仔细讲一下where的条件：比较操作

• 数值或字符串比较

- >, <, >=, <=
- 相等 =
- 不等 != 或者 <>

条件：between和in

• 列名 between 值1 and 值2

- 例如： $GPA \text{ between } 90 \text{ and } 100$
- 包含90和100

• 列名 in (值1, 值2, ...)

- 例如： $Major \text{ in } ('计算机', '经管', '法律')$

条件：字符串的模式匹配

□ 匹配：字符串 like 模式

□ 不匹配：字符串 not like 模式

□ 模式：

- 普通字符：匹配这个字符
- %：百分号匹配0到多个任意字符
- _：下划线匹配任意一个字符
- [字符列表]：出现其中一个字符
- 与unix中的regex表达式有些不同

□ 例如：

- $Name \text{ like } \%Alice\%$
名字中包含Alice

- $Name \text{ like } [A-H]__\%$
名字以A到H中的一个字母开头，然后至少再有两个字符

条件：多个比较操作

- 多个比较操作可以用逻辑运算等连接起来
 - AND
 - OR
 - NOT
- 用()可以规定比较操作的优先级
 - ()
- 例如：
 - (a > 1) AND ((b < 1) OR (C > 100))

空值NULL的问题

- 我们认为NULL代表了取值未知
- NULL参与的算术运算：结果为NULL
 - 运算输入未知，那么输出也自然未知
- NULL参与的比较操作：结果为UNKNOWN

x	y	x AND y	x OR y	NOT x
T	T	T	T	F
T	U	U	T	F
T	F	F	T	F
U	U	U	U	U
U	F	F	U	U
F	F	F	F	T

3种真值
T, F, U

条件：判断列的取值是否为NULL

- 列名 is null
- 列名 is not null

Join (连接)

- Equi-join (等值连接)
 - 最简单、最广泛使用的连接操作
 - 理解和实现其它种类join的基础和精华部分
- 概念
 - 已知两个表R和S，R表的a列和S表的b列
 - 以R.a = S.b为条件的连接
 - 找到两个表中互相匹配的记录

$R \bowtie_{R.a=S.b} S$

R.a与S.b被称为join key

Join举例

TakeCourse.StudentID = Student.ID

TakeCourse

Couse ID	Student ID		
7001	131234		
7012	145678		
7005	129012		

Student

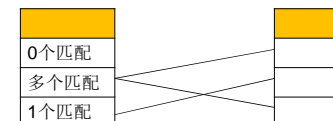
ID	Name			
131234	张飞			
145678	貂蝉			
129012	孙权			

- 这是一个特殊的例子
- Join 发生在Foreign Key与Primary Key之间
- 每一个TakeCourse记录有一个且仅有一个Student记录与之对应

通常情况

- 实际上，一个记录可以有

- 0个匹配
- 1个匹配
- 多个匹配



SQL表达连接 (SQL-89语法)

select ...
from 逗号分开的2个或多个表
where 连接条件
AND 其它选择条件

我们后面还会介绍SQL-92语法表达Join

SQL表达连接：输出每个学生所选的课程

Course

ID	Name		
7001	体系结构		
7005	数据结构		
7012	大数据处理		

Student

ID	Name			
131234	张飞			
145678	貂蝉			
129012	孙权			

TakeCourse

Couse ID	Student ID			
7001	131234			
7012	145678			
7005	129012			

select Student.Name, Course.Name
from Student, Course, TakeCourse
where TakeCourse.CourseID = Course.ID
and TakeCourse.StudentID = Student.ID;

← 多个表

← 连接条件

SQL表达连接：输出每个学生所选的课程

```
select Student.Name, Course.Name
from Student, Course, TakeCourse
where TakeCourse.CourseID = Course.ID
and TakeCourse.StudentID = Student.ID;
```

输出结果

Student.Name	Course.Name
张飞	体系结构
貂蝉	大数据处理
孙权	数据结构

传统关系代数的关系运算

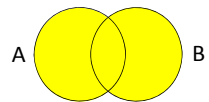
- 集合操作：并、交、差
- 选择行或列：选择 σ ✓、投影 π ✓
- 两个关系元组之间的操作：笛卡尔积、连接 \bowtie ✓
- 其它：重命名、除

☞下面继续介绍其它运算

集合操作

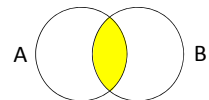
• 并 (Union)

$$\square A \cup B$$



• 交 (Intersection)

$$\square A \cap B$$



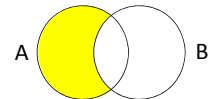
• 差 (Difference)

$$\square A - B$$

□ 注意差是非对称的

$$\square A - B \neq B - A$$

$$\square A \cap B = A - (A - B)$$



集合操作应用于关系代数

• 要求

- A和B的关系模式Schema必须相同（列数相同，每列类型也相同，列名也相同）
- 这样产生的结果的关系模式与输入相同

SQL中的集合操作

- 并：保留字Union

$A \cup B$

(select ...
from ...
where ...)

- 交：保留字Intersect

$A \cap B$

集合操作保留字

(select ...
from ...
where ...)

- 差：保留字Except

$A - B$

SQL中的集合操作

TakeCourse

Couse ID	Student ID				
7001	131234				
7012	145678				
7005	129012				

Student

ID	Name				
131234	张飞				
145678	貂蝉				
129012	孙权				

- 找到所有没选课的学生

(select ID from Student)

except

(select StudentID as ID from TakeCourse)

注意：as 用于取别名，这里使第2个
select的输出与第1个有同样的Schema

SQL中的集合操作

TakeCourse

Couse ID	Student ID				
7001	131234				
7012	145678				
7005	129012				

Student

ID	Name				
131234	张飞				
145678	貂蝉				
129012	孙权				

- 找到所有没选课的学生

(select ID from Student)

except

(select StudentID as ID from TakeCourse)

注意：SQL 中集合操作的结果默认去重！

- 可以用intersect all, union all, except all来阻止去重

重命名（对应于SQL AS功能）

- $\rho_S(A_1, A_2, \dots, A_n)(R)$

□ 把关系R重命名为关系S，各列的名字分别是A1, ..., An

(select ID from Student)

except

(select StudentID as ID from TakeCourse)

$\pi_{ID}(Student) - \rho_{Temp(ID)}(\pi_{StudentID}(TakeCourse))$

AS的使用

- 大多是在select和from子句中
- From中对表重命名的目的可能是为了self join
- 例如：Student表中增加了一列class，那么找‘Alice’的同学的名字：

```
select S2.name  
from Student as S1, Student as S2  
where S1.class=S2.class and S1.name='Alice'
```

笛卡尔积 (Cartesian Product)

- 又称为叉积或者积
- $A = \{a, b, c\}$
- $B = \{1, 2, 3, 4\}$
- 那么 $A \times B = ?$

The diagram illustrates the Cartesian product of sets A and B. Set A is represented by a vertical list of elements {a, b, c} on the left, with a bracket labeled 'A' next to it. Set B is represented by a horizontal list of elements {1, 2, 3, 4} at the top, with a bracket labeled 'B' above it. The resulting Cartesian product is shown as a 3x4 grid of pairs (a,1), (a,2), (a,3), (a,4) in the first row, (b,1), (b,2), (b,3), (b,4) in the second row, and (c,1), (c,2), (c,3), (c,4) in the third row. The entire grid is highlighted in yellow and labeled 'A x B' with an arrow pointing to it from the bottom right.

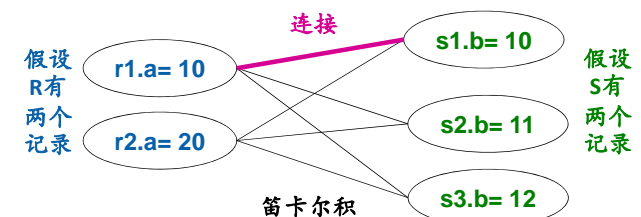
	1	2	3	4
a	(a,1)	(a,2)	(a,3)	(a,4)
b	(b,1)	(b,2)	(b,3)	(b,4)
c	(c,1)	(c,2)	(c,3)	(c,4)

SQL表达笛卡尔积

```
select *  
from R, S;
```

再次考虑连接Join

- 等值连接可以用笛卡尔积和选择来表达
- $R \bowtie_{R.a=S.b} S = \sigma_{R.a=S.b} (R \times S)$
- 实际上应该还有一次投影，把R.a和S.b中只保留一列

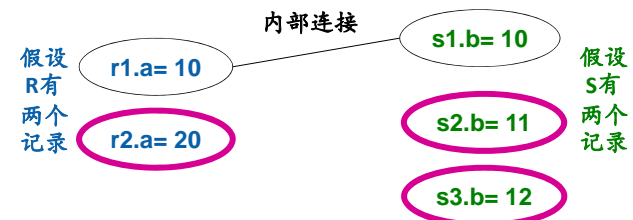


SQL-92 Join

- 上述等值连接是在SQL-89中就已经定义了
- 在SQL-92中对Join进一步丰富
 - 内部连接 (Inner Join) : 上述连接
 - 外部连接 (Outer Join)
 - Left Outer Join
 - Right Outer Join
 - Full Outer Join

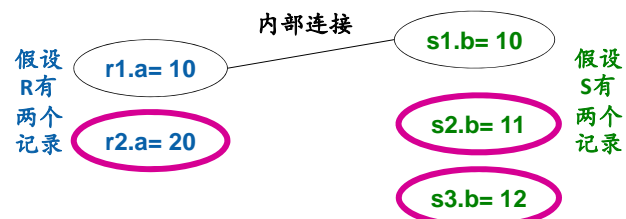
外部连接

- 什么意思呢?
 - 实际上是想把没有匹配的记录也输出



外部连接

- Left outer join: 输出左边关系中没有匹配的记录
- Right outer join: 输出右边关系中没有匹配的记录
- Full outer join: 输出左边关系和右边关系中没有匹配的记录



SQL-92 Join 语法

- Inner Join

```
select R.R_other, S.S_other
from R inner join S on R.a = S.b;
```

- 这个与SQL-89语法效果是一样的

```
select R.R_other, S.S_other
from R, S
where R.a = S.b;
```

R	
a	R_other
10	'r1c'
20	'r2c'

S	
b	S_other
10	's1c'
11	's2c'
12	's3c'

结果

R_other	S_other
'r1c'	's1c'

SQL-92 Join语法

• Left Outer Join

select *R.R_other, S.S_other*
from *R left outer join S on R.a = S.b;*

a	R_other
10	'r1c'
20	'r2c'

b	S_other
10	's1c'
11	's2c'
12	's3c'

结果

R_other	S_other
'r1c'	's1c'
'r2c'	null

SQL-92 Join语法

• Right Outer Join

select *R.R_other, S.S_other*
from *R right outer join S on R.a = S.b;*

a	R_other
10	'r1c'
20	'r2c'

b	S_other
10	's1c'
11	's2c'
12	's3c'

结果

R_other	S_other
'r1c'	's1c'
null	's2c'
null	's3c'

SQL-92 Join语法

• Full Outer Join

select *R.R_other, S.S_other*
from *R full outer join S on R.a = S.b;*

a	R_other
10	'r1c'
20	'r2c'

b	S_other
10	's1c'
11	's2c'
12	's3c'

结果

R_other	S_other
'r1c'	's1c'
'r2c'	null
null	's2c'
null	's3c'

SQL-92 Join 多个表

select *Student.Name, Course.Name*
from *Student, Course, TakeCourse*
where *TakeCourse.CourseID = Course.ID*
and *TakeCourse.StudentID = Student.ID;*

Select *Student.Name, Course.Name*
from *TakeCourse*
inner join *Student* on *TakeCourse.StudentID = Student.ID*
inner join *Course* on *TakeCourse.CourseID = Course.ID ;*

其它关于连接：自然连接

- 自然连接

- 不显式地给出join key
- Join key是两个关系中相同名字的列
- 例如：
 - $R(id, R_other), S(id, S_other)$
 - 那么自然连接 $R \bowtie S$
 - 也就是 $R \bowtie_{R.id = S.id} S$

其它关于连接： θ 连接

- 也就是非等值连接

```
select  $R.c, S.d$ 
from  $R, S$ 
where  $R.a > S.b$ ;
```

- 这里 θ 是指比较操作

除 (Division)

- A/B

- 假设 $A(x, y), B(y)$
- 那么 A/B 是 $\{x | \forall y \in B, (x, y) \in A\}$

- 注意：这个操作在SQL中没有直接支持！

A

x	y
Bob	10
Bob	12
Mary	10
Lucy	12

B

y
10
12

结果

x
Bob

Outline

- ER图和关系模型复习
- SQL 初步
 - 记录的增删改
 - 简单的查询
- 关系代数
- 举例

假设下面的关系模式

- create table *Sailors*(sid integer primary key, sname varchar(20) unique, rating integer, age real);
- create table *Boats*(bid integer primary key, bname varchar(20), color varchar(10));
- create table *Reserves*(sid integer, bid integer, day date primary key (sid, bid, day), foreign key (sid) references *Sailors*(sid), foreign key (bid) references *Boats*(bid));

- 水手、船、预订三个表

例1：找出评价高于8的所有水手？

- create table *Sailors*(sid integer primary key, sname varchar(20) unique, rating integer, age real);
- create table *Boats*(bid integer primary key, bname varchar(20), color varchar(10));
- create table *Reserves*(sid integer, bid integer, day date primary key (sid, bid, day), foreign key (sid) references *Sailors*(sid), foreign key (bid) references *Boats*(bid));

- SQL? 关系代数?

例1：解答

```
select *  
from Sailors  
where rating > 8;
```

$\sigma_{\text{rating} > 8}(\text{Sailors})$

例2：找出所有水手的名字和评价？

- create table *Sailors*(sid integer primary key, sname varchar(20) unique, rating integer, age real);
- create table *Boats*(bid integer primary key, bname varchar(20), color varchar(10));
- create table *Reserves*(sid integer, bid integer, day date primary key (sid, bid, day), foreign key (sid) references *Sailors*(sid), foreign key (bid) references *Boats*(bid));

- SQL? 关系代数?

例2：解答

```
select sname, rating
from Sailors;
```

$\pi_{sname, rating}(Sailors)$

例3：找出评价高于8的所有水手的名字？

- create table *Sailors*(*sid* integer primary key, *sname* varchar(20) unique, *rating* integer, *age* real);
- create table *Boats*(*bid* integer primary key, *bname* varchar(20), *color* varchar(10));
- create table *Reserves*(*sid* integer, *bid* integer, *day* date primary key (*sid*, *bid*, *day*), foreign key (*sid*) references *Sailors*(*sid*), foreign key (*bid*) references *Boats*(*bid*));

• SQL? 关系代数?

例3：解答

```
select sname
from Sailors
where rating > 8;
```

$\pi_{sname}(\sigma_{rating > 8}(Sailors))$

例4：预订了103号船水手的名字？

- create table *Sailors*(*sid* integer primary key, *sname* varchar(20) unique, *rating* integer, *age* real);
- create table *Boats*(*bid* integer primary key, *bname* varchar(20), *color* varchar(10));
- create table *Reserves*(*sid* integer, *bid* integer, *day* date primary key (*sid*, *bid*, *day*), foreign key (*sid*) references *Sailors*(*sid*), foreign key (*bid*) references *Boats*(*bid*));

• SQL? 关系代数?

例4：解答

```
select sname
from Sailors, Reserves
where Sailors.sid=Reserves.sid and Reserves.bid=103;
```

$\pi_{sname}(Sailors \bowtie_{Sailors.sid=Reserves.sid} \sigma_{bid=103}(Reserves))$

例5：找出预订了红色船的水手的名字？

- create table *Sailors*(*sid* integer primary key, *sname* varchar(20) unique, *rating* integer, *age* real);
- create table *Boats*(*bid* integer primary key, *bname* varchar(20), *color* varchar(10));
- create table *Reserves*(*sid* integer, *bid* integer, *day* date primary key (*sid*, *bid*, *day*), foreign key (*sid*) references *Sailors*(*sid*), foreign key (*bid*) references *Boats*(*bid*));

• SQL? 关系代数?

例5：解答

```
select sname
from Sailors, Reserves, Boats
where Sailors.sid=Reserves.sid
and Reserves.bid=Boats.bid
and color='red';
```

$\pi_{sname}(Sailors \bowtie_{Sailors.sid=Reserves.sid} Reserves \bowtie_{Reserves.bid=Boats.bid} \sigma_{color='red'}(Boats))$

例6：找出Bob在今天预订的船的颜色？

- create table *Sailors*(*sid* integer primary key, *sname* varchar(20) unique, *rating* integer, *age* real);
- create table *Boats*(*bid* integer primary key, *bname* varchar(20), *color* varchar(10));
- create table *Reserves*(*sid* integer, *bid* integer, *day* date primary key (*sid*, *bid*, *day*), foreign key (*sid*) references *Sailors*(*sid*), foreign key (*bid*) references *Boats*(*bid*));

• SQL? 关系代数?

例6: 解答

```
select color
from Sailors, Reserves, Boats
where Sailors.sid=Reserves.sid
and Reserves.bid=Boats.bid
and sname='Bob'
and day=date '2018-09-20';
```

```
 $\pi_{\text{color}}(\sigma_{\text{sname}='Bob'}(\text{Sailors}))$   
 $\bowtie_{\text{Sailors.sid=Reserves.sid} \wedge \sigma_{\text{day}=2018-09-20}(\text{Reserves})}$   
 $\bowtie_{\text{Reserves.bid=Boats.bid}}(\text{Boats})$ 
```

例7: 找出至少预订了两只船的水手的名字?

- create table Sailors(sid integer primary key, sname varchar(20) unique, rating integer, age real);
- create table Boats(bid integer primary key, bname varchar(20), color varchar(10));
- create table Reserves(sid integer, bid integer, day date primary key (sid, bid, day), foreign key (sid) references Sailors(sid), foreign key (bid) references Boats(bid));

• SQL? 关系代数?

例7: 解答

```
select distinct sname
from Sailors, Reserves as R1, Reserves as R2
where Sailors.sid=R1.sid
and Sailors.sid=R2.sid
and R1.bid!=R2.bid;
```

```
 $\pi_{\text{sname}}(\sigma_{\text{R1.bid} \neq \text{R2.bid}}(\text{Sailors}$   
 $\bowtie_{\text{Sailors.sid=R1.sid}} \rho_{\text{R1}}(\text{sid, bid, day})(\text{Reserves})$   
 $\bowtie_{\text{Sailors.sid=R2.sid}} \rho_{\text{R2}}(\text{sid, bid, day})(\text{Reserves})))$ 
```

例8: 找出年龄超过20但没有预订任何船的水手的名字?

- create table Sailors(sid integer primary key, sname varchar(20) unique, rating integer, age real);
- create table Boats(bid integer primary key, bname varchar(20), color varchar(10));
- create table Reserves(sid integer, bid integer, day date primary key (sid, bid, day), foreign key (sid) references Sailors(sid), foreign key (bid) references Boats(bid));

• SQL? 关系代数?

例8：解答

```
(select sname
from Sailors
where age > 20)
```

except

```
(select sname
from Sailors, Reserves
where Sailors.sid=Reserves.sid
);
```

```
 $\pi_{sname}(\sigma_{age>20}(Sailors))$ 
-  $\pi_{sname}(Sailors \bowtie_{Sailors.sid=Reserves.sid} Reserves)$ 
```

小结

- ER图和关系模型复习

- SQL 初步

- 记录的增删改
- 简单的查询
 - 集合操作：并、交、差
 - 选择行或列：选择、投影
 - 两个关系元组之间的操作：笛卡尔积、连接
 - 其它：重命名、除

- 关系代数