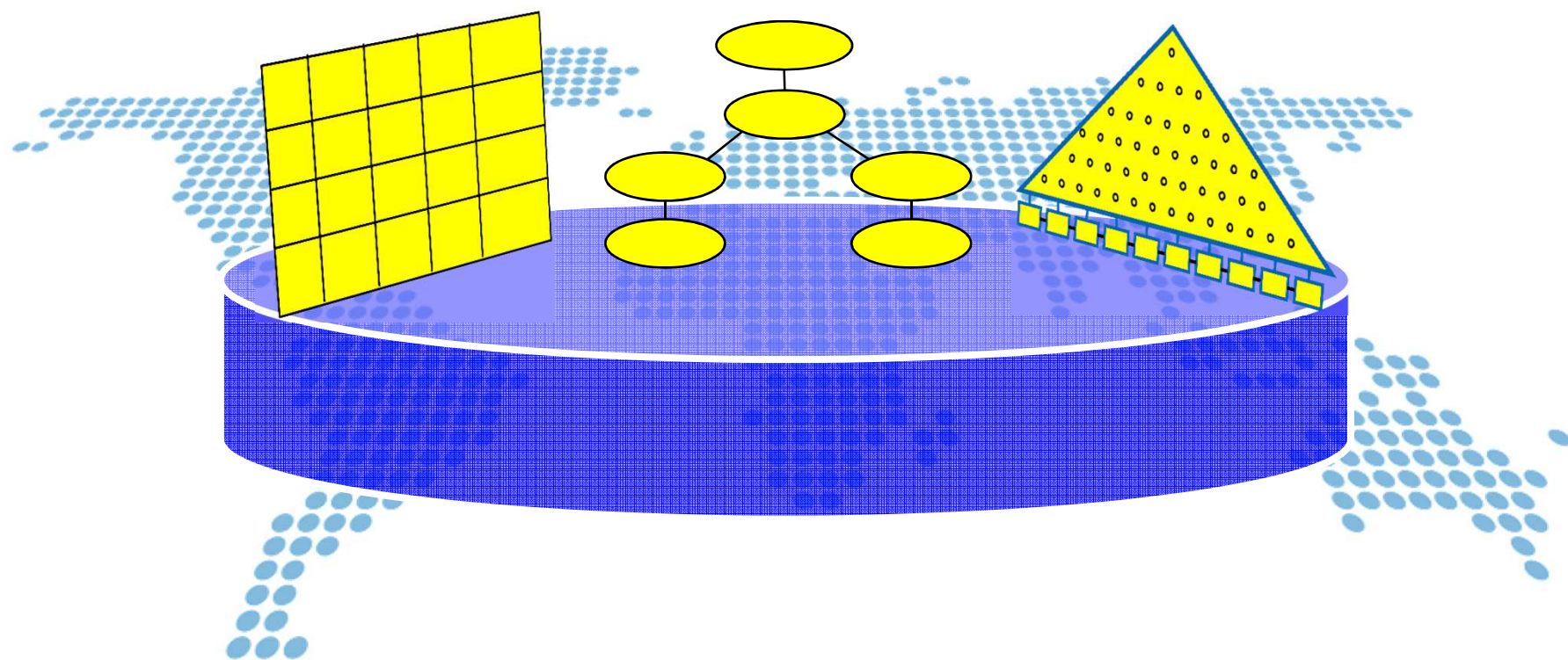


数据库系统

# 关系模型与SQL (1)

陈世敏

(中科院计算所)



# Outline

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

# 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

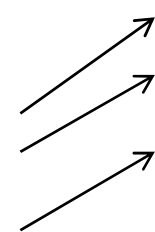
列数称为度(arity 或 degree)

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  
(~10<sup>4</sup>)




记录的条数称为  
基(cardinality)

# Key (键)

- Candidate key (候选键)
  - 唯一确定一个记录
  - 最少的列组合
- Primary key (主键)
  - 选一个候选键为主键
- Foreign key (外键)
  - 是另一个表的Primary key
  - 唯一确定另一个表的一个记录

# 举例：学生信息表


Primary Key (主键): 唯一确定一个记录



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
...	...	...	...	...	...	
...	...	...	...	...	...	
...	...	...	...	...	...	
...	...	...	...	...	...	
...	...	...	...	...	...	
...	...	...	...	...	...	
...	...	...	...	...	...	

# 举例：选课信息表

Foreign Key (外键)      Foreign Key (外键)



Couse ID	Student ID	Year	Semester	Grade
7001	131234	2014	春季	85
7012	145678	2014	夏季	80
7005	129012	2013	秋季	95
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...



# 举例：Primary Key 与 Foreign Key

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			

实际上，可以把外键理解为“指针”或“引用”

# 举例：选课信息表

什么列是Primary key? (CourseID, StudentID) 的组合

Couse ID	Student ID	Year	Semester	Grade
7001	131234	2014	春季	85
7012	145678	2014	夏季	80
7005	129012	2013	秋季	95
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

# Outline

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

# SQL

- 1970s, system R的SEQUEL (Structured English QUery Language)
- 1980s, 成为ANSI和ISO标准
  - SQL-86: 是IBM SQL实现的一个子集
  - SQL-89: 替代了SQL-86, 定义了基础常用的SQL功能
  - SQL-92: 更多数据类型、多种连接操作、断言、临时表、动态SQL等
  - SQL-99: 触发器、存储过程、用户定义函数、数据仓库扩展等
  - 之后的标准增加了对XML的支持等
  - 大部分数据库系统实现了SQL-92, 并在此基础上进行了扩展, 对于SQL-99和之后的标准大多只实现了标准的一部分
- 主流的关系型数据库语言
  - Declarative language (宣告式编程)
  - 4GL (第四代语言)
    - 3GL: 例如 C, C++, Java, C#, Javascript等

# SQL语句的分类

- 数据定义（Data Definition Language, DDL）
  - 表的增删改、视图的操作等
- 数据操纵（Data Manipulation Language, DML）
  - 记录的增删改查
- 数据控制（Data Control Language, DCL）
  - 访问控制

# Create Database

- 创建一个数据库

- RDBMS可以管理多个数据库
- 每个数据库可以包含多个表

- 语法

create database *数据库名*;

- 举例

create database *my\_example\_db*;

# 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

# 名字

- 名字（表名、列名等）
- 最长128个字符
- 通常情况不分大小写
  - 如果要区分，那么需要用“”把名字括起来
  - 例如：“HowMany”



# 类型

蓝色的是标准类型，其它类型在很多系统中都支持

## • 数值类型

### □ 整数

- tinyint: 8位
- smallint: 16位
- integer: 32位
- bigint: 64位

### □ 浮点数

- real: 32位浮点数
- double或float: 64位浮点数

### □ 精确小数

- decimal(size, d)或 numeric(size, d): size位十进制数，其中小数点右侧有d位小数
  - 所以，小数点左侧有size-d位数字

# 类型

- 字符串类型

- `char(n)`: 定长字符串, 长度为n
- `varchar(n)`: 变长字符串, 长度最大为n

- 日期和时间

- `date`: 日期, 例如 `date '2016-09-08'`
- `time`: 时间, 例如 `time '13:50:12.5'`
- `timestamp`: 时间戳, 包含日期和时间

- 其它

- `BLOB`: Binary Large Object, 例如最长4GB
- `CLOB`: Character Large Object, 例如最长4GB
- `enum(val1, val2, val3...)`: 枚举类型

# 默认值

- 当一个列的值缺失时默认的值： NULL
- 什么时候会缺失？
  - 当插入一条新记录时，某列没有给具体的值
  - 当修改表的定义，增加一个新的列时

☞ 可以定义非NULL的默认值

# 定义默认值：Default

## Student

ID	Name	Birthday	Gender	Major	Year	GPA
...	...	...	...	...	...	...

```
create table 表名 (  
    列名 类型 default 默认值,  
);
```

```
create table Student (  
    ID integer, 字符串形成日期  
    Name varchar(20),  
    Birthday date default date '0000-00-00',  
    Gender enum('M', 'F'),  
    Major varchar(20) default 'Computer Science'  
);
```

# 声明主键Primary Key: 附加声明

## Student

ID	Name	Birthday	Gender	Major	Year	GPA
...	...	...	...	...	...	...

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

← 只适用于主键是一个属性的情况

# 声明主键Primary Key：独立声明

## 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,  
    primary key (ID)  
);
```

独立声明：适用于任何主键，主键可以包含多个属性

# 声明候选键：附加声明

## Student

ID	Name	Birthday	Gender	Major	Year	GPA
...	...	...	...	...	...	...

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

只适用于键是一个属性的情况



# 声明候选键：独立声明

## Student

ID	Name	Birthday	Gender	Major	Year	GPA
...	...	...	...	...	...	...

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

独立声明：适用于任何候选键，可以包含多个属性





# Primary Key和Unique的区别

- Primary Key

- 声明主键
- 唯一决定一个记录
- 组成主键的属性不为NULL

- Unique

- 声明候选键
- 唯一决定一个记录
- 候选键的属性可以为NULL

# 特别要求一列不为NULL

## Student

ID	Name	Birthday	Gender	Major	Year	GPA
...	...	...	...	...	...	...

```
create table Student (  
  ID integer not null,  
  Name varchar(20),  
  Birthday date,  
  Gender enum('M', 'F'),  
  Major varchar(20),  
  Year year,  
  GPA float  
);
```

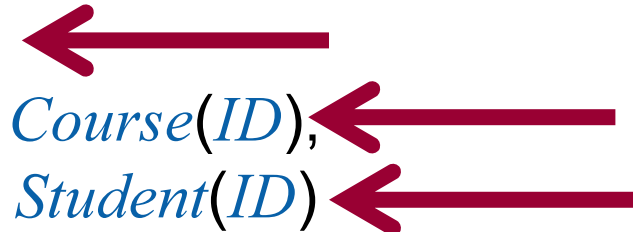


# 外键Foreign Key（又称为引用完整性）

## TakeCourse

Couse ID	Student ID	Year	Semester	Grade
...	...	...	...	...

```
create table TakeCourse (  
    CourseID integer,  
    StudentID integer,  
    Year year,  
    Semester enum('Spring', 'Summer', 'Fall'),  
    Grade float,  
    primary key (CourseID, StudentID),  
    foreign key (CourseID) references Course(ID),  
    foreign key (StudentID) references Student(ID)  
);
```



# 删除表： Drop Table

- 删除一个表

drop table 表名;

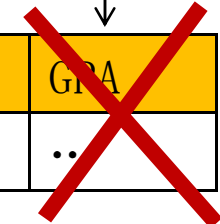
- 举例

drop table *Student*;

# 修改表：Alter Table

Student

ID	Name	Birthday	Gender	Major	Year	GPA
...	...	...	...	...	...	..



- 删除列

alter table 表名 drop 列名;

- 举例

alter table *Student* drop *GPA*;

# 修改表：Alter Table

Student

ID	Name	Birthday	Gender	Major	Year	GPA
...	...	...	...	...	...	...

增加列



- 增加列

alter table 表名 add 列名 类型;

alter table 表名 add 列名 类型 default 默认值;

- 举例

alter table *Student* add *GPA* float;

alter table *Student* add *GPA* float default 100;

# 表的增删改

- DDL

- 增: create table

- 删: drop table

- 改: alter table

- 注意: alter table可能是一个很昂贵的操作

- 下面我们介绍DML中表记录的增删改

- 增: insert

- 删: delete

- 改: update

# 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；
```

# 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
  - 表的增删改
  - 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
- “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

- 特点

- 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)  $\sigma$
- Projection (投影)  $\pi$
- 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

$\sigma_{\text{条件}}(\text{表名})$


- 返回给定表中符合给定条件的行

# Selection (选择)

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

# SQL表达选择

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

## 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

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

```
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)

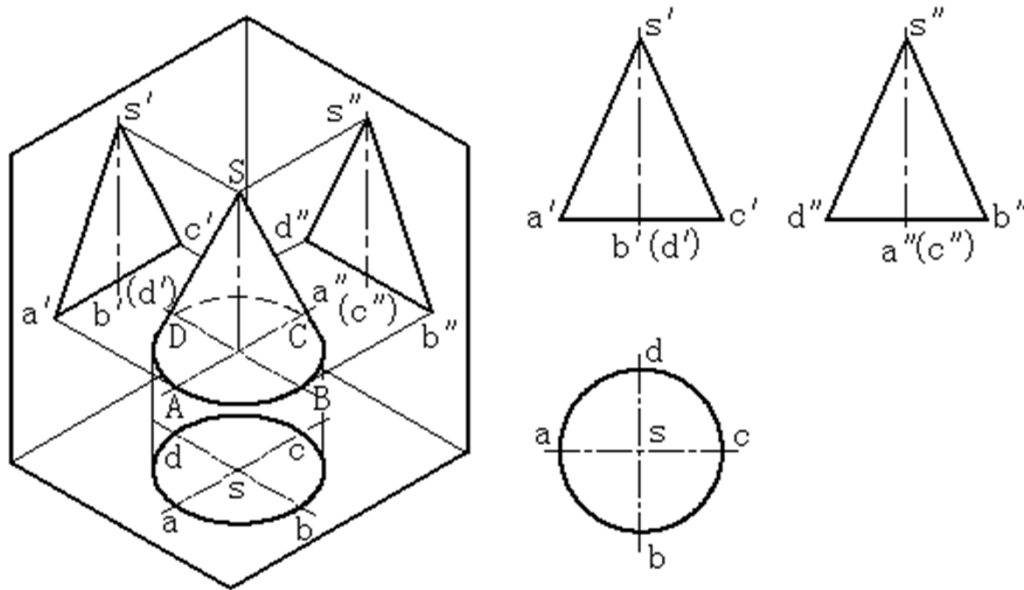
- 从一个表中提取一些列

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

# 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)

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

- 从一个表中提取一些列

提取学生姓名和平均分



The diagram illustrates the projection operation. Two large red arrows point downwards from the text '提取学生姓名和平均分' to the 'Name' and 'GPA' columns of the table below, indicating that only these two columns are being selected from the original data.

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
...	...	...	...	...	...	
...	...	...	...	...	...	

# SQL表达投影

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

## 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

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

```
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：重复？

$\pi_{\text{GPA}}(\text{Student})$

```
select GPA  
from Student;
```

输出结果

GPA
85
90
80
90
95

**注意：**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

$\pi_{\text{GPA}}(\text{Student})$

```
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

$\pi_{\text{Name, GPA}}(\sigma_{\text{Major}=\text{'计算机'}}(\text{Student}))$

## 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表达式选择+投影

$\pi_{\text{Name, GPA}}(\sigma_{\text{Major}=\text{'计算机'}}(\text{Student}))$

```
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

# 条件：比较操作

- 数值或字符串比较

- $>$ ,  $<$ ,  $>=$ ,  $<=$

- 相等 =

- 不等  $\neq$  或者  $<>$



# 条件： between和in

- 列名 between 值1 and 值2

- 例如： *GPA* between 90 and 100

- 包含90和100

- 列名 in (值1, 值2, ...)

- 例如： *Major* 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

□ 例如：

- *Name* like '%Alice%'  
名字中包含Alice

- *Name* 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

- ( )

# 空值NULL的问题

- NULL参与的算术运算：结果为NULL
- 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

# 判断空值的条件

- 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

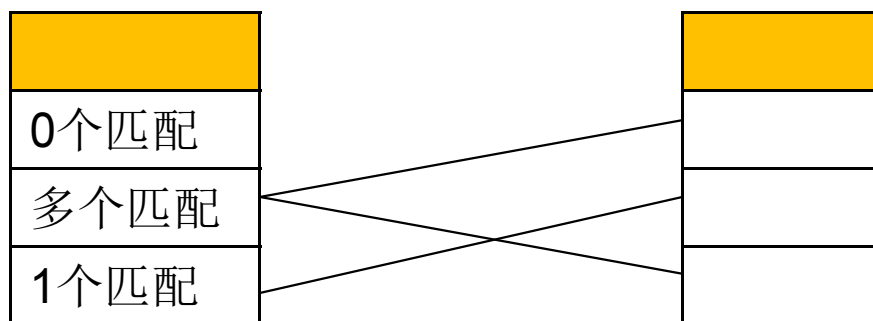
ID	Name				
131234	张飞				
145678	貂蝉				
129012	孙权				

- 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

- 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 多个表

where 连接条件

# 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

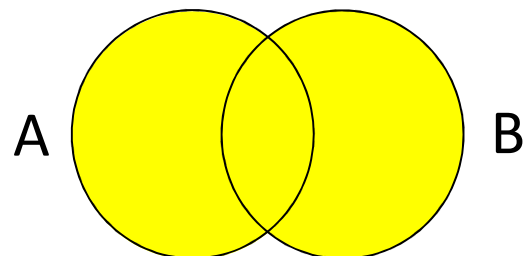
- 集合操作：并、交、差
- 选择行或列：选择 $\sigma$ ✓、投影 $\pi$ ✓
- 两个关系元组之间的操作：笛卡尔积、连接 $\bowtie$ ✓
- 其它：重命名、除

☞ 下面继续介绍其它运算

# 集合操作

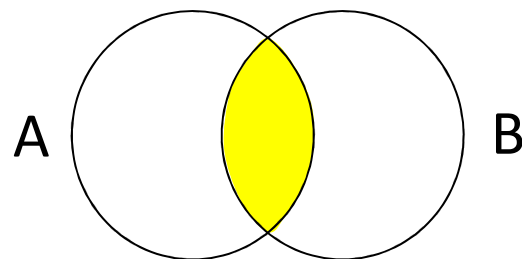
- 并 (Union)

- $\square A \cup B$



- 交 (Intersection)

- $\square A \cap B$



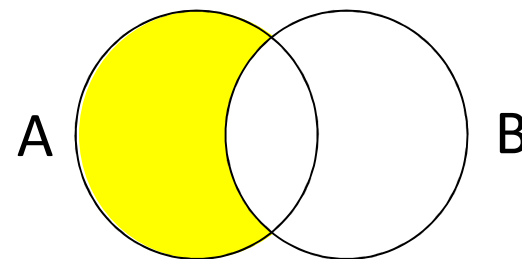
- 差 (Difference)

- $\square A - B$

- $\square$  注意差是非对称的

- $\blacksquare A - B \neq B - A$

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



# 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
- 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

$$\square A \cup B$$

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

- 交：保留字Intersect

$$\square A \cap B$$

集合操作保留字

- 差：保留字Except

$$\square A - B$$

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

# 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			

- 找到所有选了课的学生

(select *ID* from *Student*)

**intersect**

(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				

- 找到所有选了课的学生

(select *ID* from *Student*)

**intersect**

(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）

- $\rho_{S(A1,A2,\dots,A_n)}(R)$

□ 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*)

**intersect**

(select *StudentID* as *ID* from *TakeCourse*)

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

# 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  $A \times B = ?$

		B			
		1	2	3	4
A	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)

$A \times B$

# SQL表达笛卡尔积

select \*

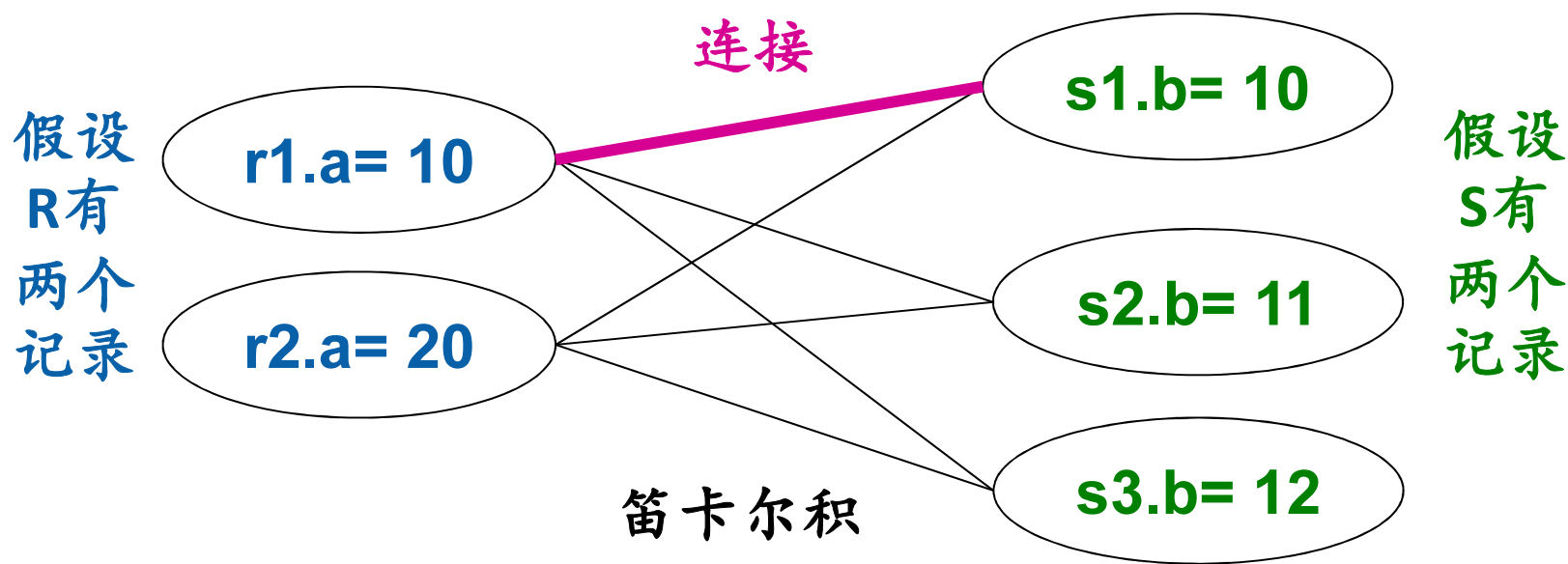
from  $R, S$ ;

# 再次考虑连接Join

- 等值连接可以用笛卡尔积和选择来表达

$$\square R \bowtie_{R.a=S.b} S = \sigma_{R.a=S.b} (R \times S)$$

- 实际上应该还有一次投影，把R.a和S.b中只保留一列



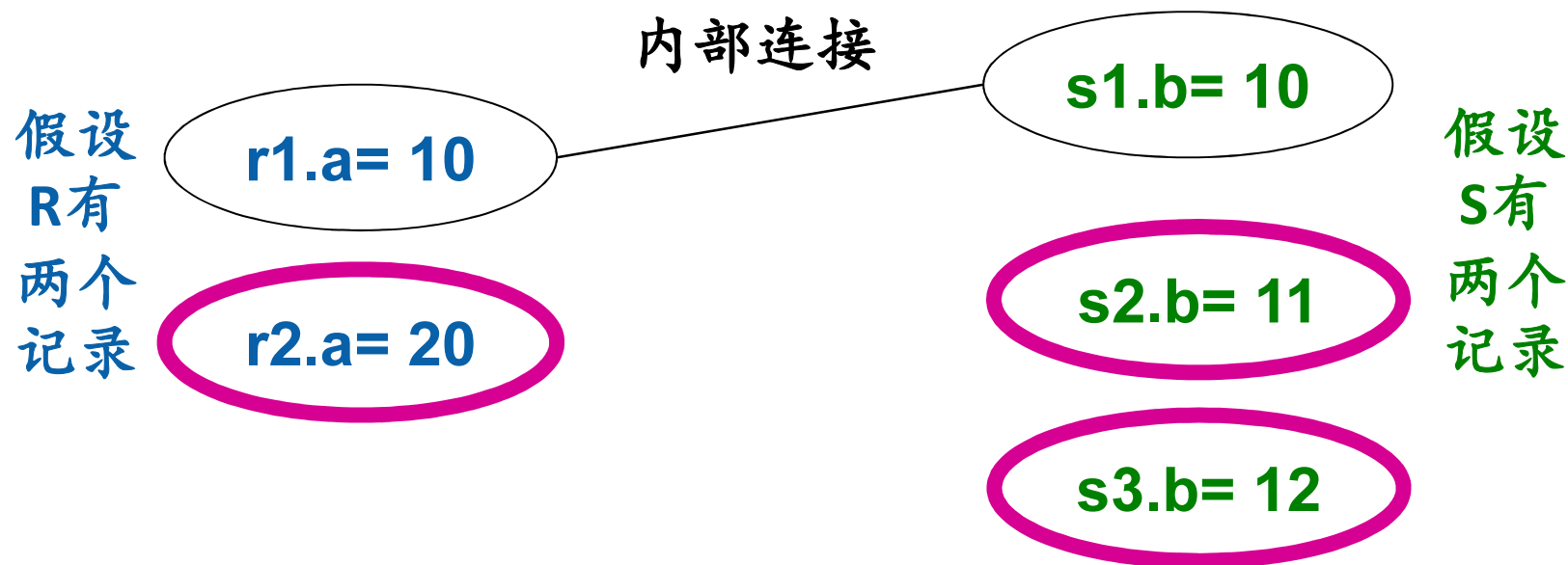
# SQL-92 Join

- 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？

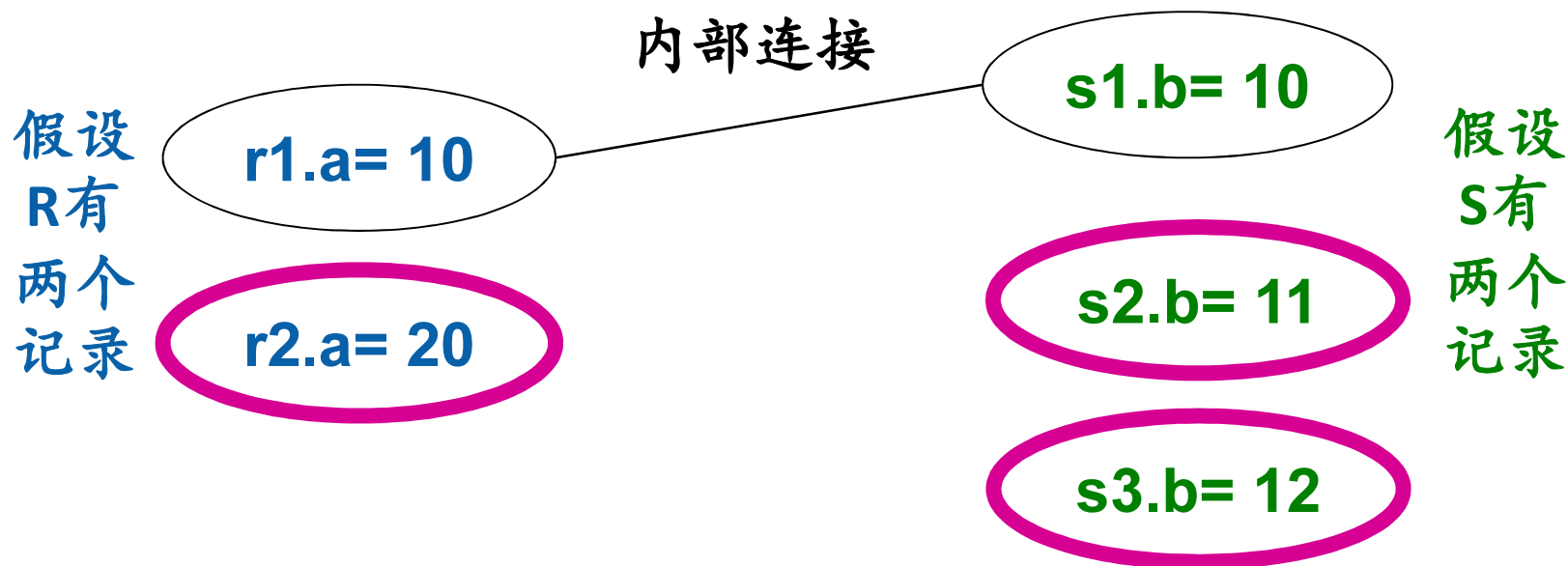
□ 实际上是想把没有匹配的记录也输出





# 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语法效果一样

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  $R.a = S.b$ ;

$R$

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'
'r2c'	null

# SQL-92 Join语法

- Right Outer Join

select  $R.R\_other$ ,  $S.S\_other$   
from  $R$  right outer join  $S$  on  $R.a = S.b$ ;

$R$

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'
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  $R.a = S.b$ ;

**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'
'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 $R \bowtie_{R.id = S.id} S$

- 注意：这个我个人不推荐自然连接，因为这种隐式定义的内容有可能错。

- 例如，新增了一列的列名和另一个表中某个列恰好相同？尤其是当初开发人员和现在不一样时

- 发生了错误，需要花很多精力才能找到！

# 其它关于连接： $\theta$ 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 $\theta$ 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

- 关系模型复习
- SQL 初步
  - 表的增删改
  - 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) );
- 
- 书上的例子，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;
```

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

### 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;
```

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

## 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*;

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

## 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';
```

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

## 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 '2016-09-08';
```

$\pi_{\text{color}}(\sigma_{\text{sname}='Bob'}(\text{Sailors}))$

$\bowtie_{\text{Sailors.sid=Reserves.sid}} \sigma_{\text{day}=2016-09-08}(\text{Reserves})$

$\bowtie_{\text{Reserves.bid=Boats.bid}} \text{Boats})$

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

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

## 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  
);
```

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

# 小结

- 关系模型复习

- SQL 初步

- 表的增删改

- 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