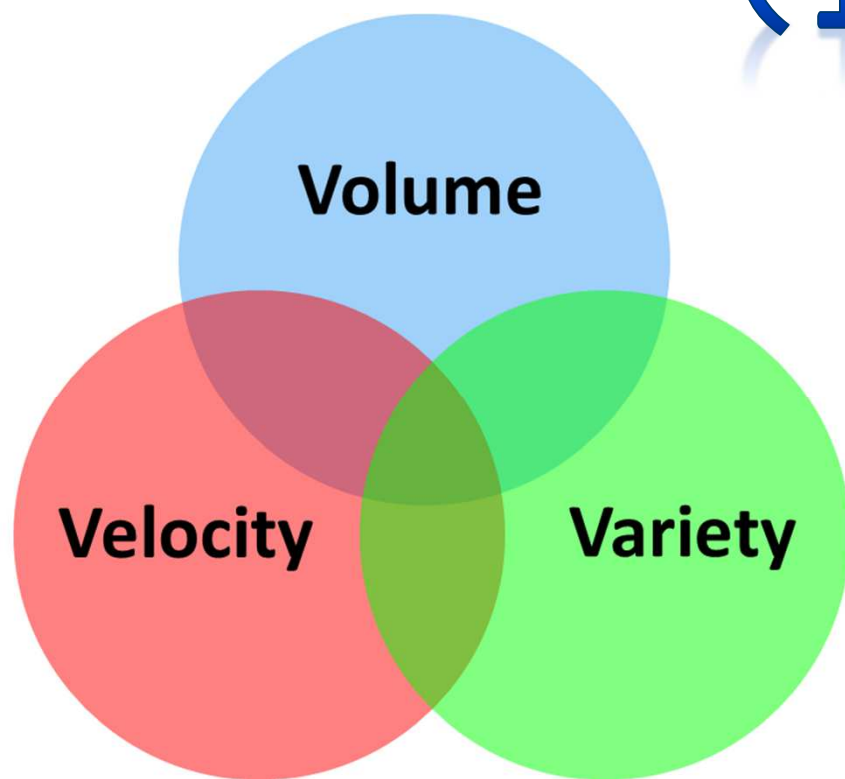


大数据系统与大规模数据分析

关系型数据管理系统 (1)



陈世敏

中科院计算所
计算机体系结构
国家重点实验室

©2015-2020 陈世敏

Outline

- 关系型数据模型
- 关系型运算
- SQL语言

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

举例：学生信息表

每一列是一个属性 (~10)

每一行是一个学生记录 (~10⁴)

| 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：学生信息表

原子
类型

数值

字符串

日期

类别

字符串

年

数值

每一行是
一个学生
记录
(~10⁴)

| 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 |
| ... | ... | ... | ... | ... | ... | |
| ... | ... | ... | ... | ... | ... | |
| ... | ... | ... | ... | ... | ... | |
| ... | ... | ... | ... | ... | ... | |
| ... | ... | ... | ... | ... | ... | |
| ... | ... | ... | ... | ... | ... | |

什么是原子类型？无内部嵌套结构

√ Int

√ Double

√ Char string

√ Int 基础上表达的类型：Date, Enum, ...

✗ 编程语言中的struct

✗ class

✗ array

✗ list, set, map ...

举例：学生信息表

每一列是一个属性 (~10)

每一行是一个学生记录 (~10⁴)

记录
无序

| 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：学生信息表

每一列是一个属性 (~10)

每一行是一个学生记录 (~10⁴)

非常瘦长

| 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 |
| ... | ... | ... | ... | ... | ... | |
| ... | ... | ... | ... | ... | ... | |
| ... | ... | ... | ... | ... | ... | |
| ... | ... | ... | ... | ... | ... | |
| ... | ... | ... | ... | ... | ... | |
| ... | ... | ... | ... | ... | ... | |

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）

Schema vs. Instance

- Schema: 类型

- 一个表的类型是由每个列的类型决定的

- Instance: 具体取值

- 具体存储哪些记录，每个列的具体值

- 由具体应用决定的

- 这样区分的意义

- Schema只需要定义一次


- 可以对应多个instance

Key (键)

- 特殊的列
- 有什么用?
 - 取值是唯一的
 - 唯一确定一个记录
- Primary key (主键)
 - 唯一确定本表中的一个记录
- 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 |
| ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... |

关系数据模型

- Relation/Table, 表
- Column, 列
- Row, record, tuple, 行
- Schema vs. instance
- Primary key vs. foreign key (主键与外键)
- 问题?

SQL

- 1970s, system R的SEQUEL (Structured English QUery Language)语言
- 1980s, 成为ANSI和ISO标准
- 主流的关系型数据库语言
 - 定义表
 - 插入、删除、修改
 - 关系型运算

SQL 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

声明主键Primary Key

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)  
);
```

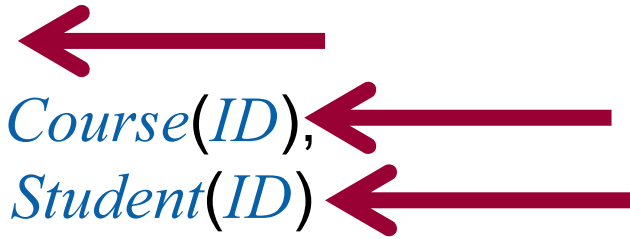
主键可以包含
多个属性

SQL Create Table + Foreign Key

TakeCourse

| Couse ID | Student ID | Year | Semester | Grade |
|----------|------------|------|----------|-------|
| ... | ... | ... | ... | ... |

```
create table TakeCourse (  
  CourseID integer NOT NULL,  
  StudentID integer NOT NULL,  
  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)  
);
```



SQL 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 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：

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

SQL Delete

Student

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

删除上述记录：

```
delete from Student  
where ID = 131234;
```

删除所有计算机系的学生记录：

```
delete from Student  
where Major = '计算机';
```

SQL Update

Student

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

86

用新的GPA代替到上个学期为止的GPA:

```
update Student
set GPA = 86
where ID = 131234;
```

主要关系运算

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

Selection (选择)

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

- 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 |
| ... | ... | ... | ... | ... | ... | |
| ... | ... | ... | ... | ... | ... | |
| ... | ... | ... | ... | ... | ... | |
| ... | ... | ... | ... | ... | ... | |
| ... | ... | ... | ... | ... | ... | |

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;

多个条件可以用and, or, ()等组合

SQL表达选择

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

```
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)

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

- 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 these two columns are the result of the projection.

| 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表达式选择+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

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
- 我们这里只介绍equi-join

- 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

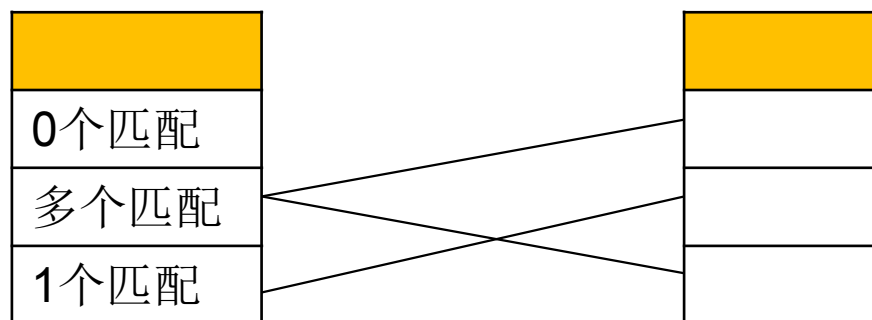
| ID | Name | | | | |
|--------|------|--|--|--|--|
| 131234 | 张飞 | | | | |
| 145678 | 貂蝉 | | | | |
| 129012 | 孙权 | | | | |

- 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：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 |

SQL Select

select 列名,...,列名
from 表,..., 表
where 条件

group by 列名,...,列名
having 条件
order by 列名,...,列名

投影
选择, 连接
选择, 连接

Group by: 分组统计

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 |

统计各系2013-2014年入学的学生人数

```
select Major, count(*)  
from Student  
where Year >= 2013 and Year <= 2014  
group by Major;
```

输出结果

| Major | Count |
|-------|-------|
| 经管 | 1 |
| 计算机 | 2 |

统计函数

- SQL定义的统计函数包括sum, count, avg, max, min
- 在一些系统中可以扩展

Having: 在group by 基础上选择

统计各系2013-2014年入学的学生人数，过滤掉人数<2的系

```
select Major, count(*) as Cnt
from Student
where Year >= 2013 and Year <= 2014
group by Major
having Cnt >= 2;
```

输出结果

| Major | Cnt |
|-------|-----|
| 计算机 | 2 |

Order by: 排序

统计各系2013-2014年入学的学生人数，并按照人数从大到小排序输出

```
select Major, count(*) as Cnt
from Student
where Year >= 2013 and Year <= 2014
group by Major
order by Cnt desc;
```

输出结果

| Major | Cnt |
|-------|-----|
| 计算机 | 2 |
| 经管 | 1 |

desc (descending 减少)表示从大到小排序
asc (ascending 增加) 表示从小到大排序

SQL Select

select 列名,...,列名

from 表,..., 表

where 条件

group by 列名,...,列名

having 条件

order by 列名,...,列名

投影

选择, 连接

选择, 连接

分组统计

分组后选择

结果排序

问题?

思考：如何计算平均成绩？

Student

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

TakeCourse

| Couse ID | Student ID | Year | Semester | Grade |
|----------|------------|------|----------|-------|
| ... | ... | ... | ... | ... |

- 已知 Student.GPA 为截止上学期的平均成绩
- 需要重新计算每位学生的GPA
- 写一个SQL语句输出

| Student. Name | NewGPA |
|---------------|--------|
| ... | ... |

部分解答：如何计算平均成绩？

Student

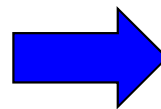
| ID | Name | Birthday | Gender | Major | Year | GPA |
|-----|------|----------|--------|-------|------|-----|
| ... | ... | ... | ... | ... | ... | ... |

TakeCourse

| Couse ID | Student ID | Year | Semester | Grade |
|----------|------------|------|----------|-------|
| ... | ... | ... | ... | ... |

```
select TakeCourse.StudentID, avg(Grade) as NewGPA
from TakeCourse
group by TakeCourse.StudentID;
```

| ID | NewGPA |
|-----|--------|
| ... | ... |



| Student.Name | NewGPA |
|--------------|--------|
| ... | ... |

解答：如何计算平均成绩？

Student

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

TakeCourse

| Couse ID | Student ID | Year | Semester | Grade |
|----------|------------|------|----------|-------|
| ... | ... | ... | ... | ... |

```
select Student.Name, avg(Grade) as NewGPA
from TakeCourse, Student
where TakeCourse.StudentID = Student.ID
group by Student.ID, Student.Name;
```

课后问题

- 请大家回答下面关于第1周内容的问题
 - 请问机械硬盘的随机读性能和顺序读性能是什么样的？
 - 请问什么是选择、投影和连接？
- 请在“作业>远程课堂问答”中上传答案
 - 回答每堂课后提问，每堂上传1个txt文件，名称为第xx周
 - 例如，第1周.txt