数据库原理

The Theory of Database System

第三章 关系数据库标准语言SQL

中国矿业大学计算机学院

本讲主要内容

数据查询——单表查询

SELECT语句

- 语法格式

SELECT 指定要显示的属性列 >
[，<目标列表达式>] …
FROM < 指定查询对象(基本表或视图) >] …
[WHERE 指定查询条件
[GROUP BY 对查询结果按指定列的值分组，该属性列值相等的元组为一个组。
[HAVING 筛选出只有满足指定条件的组
[ORDER BY <列名2> [ASC|DESC]];
对查询结果表按指定列值的升序或降序排序

学生-课程数据库

- 学生(学号，姓名，性别，籍贯，出生年份，学院)
- 课程(课程号，课程名，学时，先修课程号，课程性质)
- 学习(学号，课程号，成绩)

投影

[例] 查询全体学生的学号与姓名。

```
SELECT 学号，姓名
FROM 学生；
```

[例] 查询全体学生的姓名、学号和学院。

```
SELECT 姓名，学号，学院
FROM 学生；
```

一、投影

[例] 查询全体学生的详细记录。

```
SELECT 学号,姓名,性别,籍贯,出生年份,学院
FROM 学生;
```

或

```
SELECT  *
FROM 学生;
```

投影项

SELECT子句的<目标列表达式>

- 算术表达式
- 字符串常量
- 函数

投影项

【例】查询学生的姓名和年龄。

```
SELECT 姓名, year(now())-出生年份
FROM  学生;
```

查询结果：

姓名	Expr1001
王英	21
王小梅	18
张小飞	22
孙志鹏	20
徐颖	21
….	…

投影项

使用列别名改变查询结果的列标题：

```
SELECT 姓名, year(now())-出生年份 as 年龄
FROM  学生;
```

查询结果：

姓名	年龄
王英	21
王小梅	18
张小飞	22
孙志鹏	20
徐颖	21
….	…

二、选择

- 查询满足条件的元组

【例】查询有不及格门次的学生的学号。

```
SELECT  学号
FROM 学习
WHERE 成绩 < 60;
```

查询结果：

学号
091504
091504
…

选择（续）

说明：

- SQL查询语句的结果也是一个关系。
- 关系代数中：关系是一个集合。
- 实际的商用数据库产品中：不删除重复元组。

选择（续）

- 在SELECT子句中使用DISTINCT短语去除重复的元组

```
SELECT  DISTINCT 学号
FROM 学习
WHERE 成绩 < 60;
```

查询结果：

学号
091504
091505
…

选择（续）

常用的查询条件

查询条件	谓 词
比较	=、<>、>、<、>=、<=
算术运算	+、-、*、/
确定范围	BETWEEN AND, NOT BETWEEN AND
确定集合	IN, NOT IN
字符匹配	LIKE , NOT LIKE
空值	IS NULL , IS NOT NULL
多重条件	AND, OR

选择条件：比较运算符

在WHERE子句的<比较条件>中使用比较运算符

- =，>，<，>=，<=，!=（或 <>）

[例]查询信电学院中2000年以后出生的学生记录。

```
SELECT *
FROM 学生
WHERE 学院='信电‘ AND  出生年份>=2000
```

选择条件：确定范围

- 使用谓词：BETWEEN … AND …

  NOT BETWEEN … AND …

[例]查询出生年份在1996到1998年之间（包括1996和1998年）的学生的姓名、性别、学院和出生年份。

```
SELECT 姓名, 性别, 学院, 出生年份
FROM 学生
WHERE 出生年份 BETWEEN 1996 AND 1998;
```

该查询等价于：

```
SELECT 姓名, 性别, 学院, 出生年份
FROM   学生
WHERE 出生年份 >= 1996  AND
          出生年份 <= 1998;
```

【例】查询出生年份不在1996到1998年之间（包括1996和1998年）的学生的姓名、性别、学院和出生年份。

```
SELECT 姓名, 性别, 学院, 出生年份
FROM   学生
WHERE 出生年份 NOT BETWEEN 1996 AND 1998;
```

选择条件：确定集合

- 使用谓词：IN <值表>，NOT IN <值表>

<值表>：用逗号分隔的一组取值

[例]查询信电学院 、理学院和计算机学院的学生的学号，姓名和学院。

```
SELECT 学号, 姓名, 学院
FROM  学生
WHERE 学院 IN ('信电', '理学院', '计算机');
```

[例]查询不在信电学院 、理学院和计算机学院的学生的学号、姓名和学院。

```
SELECT 学号, 姓名, 学院
FROM  学生
WHERE 学院 NOT  IN ('信电', '理学院', '计算机');
```

选择条件：字符串匹配

思考：查询所有姓王的学生的姓名、学号和性别。

- LIKE ‘<匹配串>’ [ESCAPE ‘ <转义字符>’]

  <匹配串>：指定匹配模板

  匹配模板：固定字符串或含通配符的字符串

通配符

- % (百分号) 代表任意长度（长度可以为0）的字符串
  - 例：a%b表示以a开头，以b结尾的任意长度的字符串。如acb，addgb，ab 等都满足该匹配串。
- _ (下横线) 代表任意单个字符
  - 例：a_b表示以a开头，以b结尾的长度为3的任意字符串。如acb，afb等都满足该匹配串。

【例】查询所有姓王的学生的姓名、学号和性别。

```
SELECT 姓名, 学号, 性别
FROM 学生
WHERE 姓名 LIKE  '王%';
```

【例】查询名字中第二字为“小”字的学生的姓名和学号。

```
SELECT 姓名, 学号
FROM 学生
WHERE 姓名 LIKE '_小%';
```

ESCAPE 短语

- 当用户要查询的字符串本身就含有 % 或 _ 时，要使用‘\’或者ESCAPE ‘<转义字符>’短语对通配符进行转义。

- 【例】查找课程名是DB_Design课程的课程号，课程性质。

【例】查找课程名是DB_Design课程的课程号，课程性质。

```
SELECT 课程号, 课程性质
FROM 课程
WHERE 课程名 LIKE 'DB\_Design';
```

或者

```
SELECT 课程号, 课程性质
FROM 课程
WHERE 课程名 LIKE 'DB*_Design' ESCAPE '*';
```

选择条件：判断空值

- 使用谓词 IS NULL 或 IS NOT NULL
- “IS NULL” 不能用 “= NULL” 代替

[例] 查询缺少成绩的学生的学号和相应的课程号。

```
SELECT 学号, 课程号, 成绩
FROM  学习
WHERE 成绩 IS  NULL;
```

三、使用集函数

5类主要集函数

- 计数

  COUNT（[DISTINCT|ALL] *）

  COUNT（[DISTINCT|ALL] <列名>）

- 计算总和

  SUM（[DISTINCT|ALL] <列名>）

- 计算平均值

  AVG（[DISTINCT|ALL] <列名>）

使用集函数（续）

- 求最大值

  MAX（[DISTINCT|ALL] <列名>

- 求最小值

  MIN（[DISTINCT|ALL] <列名>

- DISTINCT短语：在计算时要取消指定列中的重复值
- ALL短语：不取消重复值
- ALL为缺省值

【例】查询学生总人数。

```
SELECT COUNT(*)  AS  总人数
FROM 学生;
```

【例】查询计算机学院学生的平均年龄。

```
SELECT AVG(year(now())-出生年份)  AS 平均年龄
 FROM 学生
WHERE 学院 ='计算机';
```

【例】查询学习180101号课程的学生最高分数。

```
SELECT MAX(成绩)  AS  最高分
FROM 学习
WHERE 课程号='180101';
```

四、对查询结果分组

使用GROUP BY子句分组

细化集函数的作用对象

- 未对查询结果分组，集函数将作用于整个查询结果
- 对查询结果分组后，集函数将分别作用于每个组

【例】查询各个课程号相应的选课人数。

```
SELECT 课程号, COUNT(学号) AS 选课人数
FROM 学习
GROUP BY 课程号;
```

查询结果：

课程号	选课人数
180101	3
180102	3
180103	3
180104	1
…	…

【例】查询各个学号对应的选课数量。

```
SELECT 学号, COUNT(课程号) as 选课数量
FROM 学习
GROUP BY 学号;
```

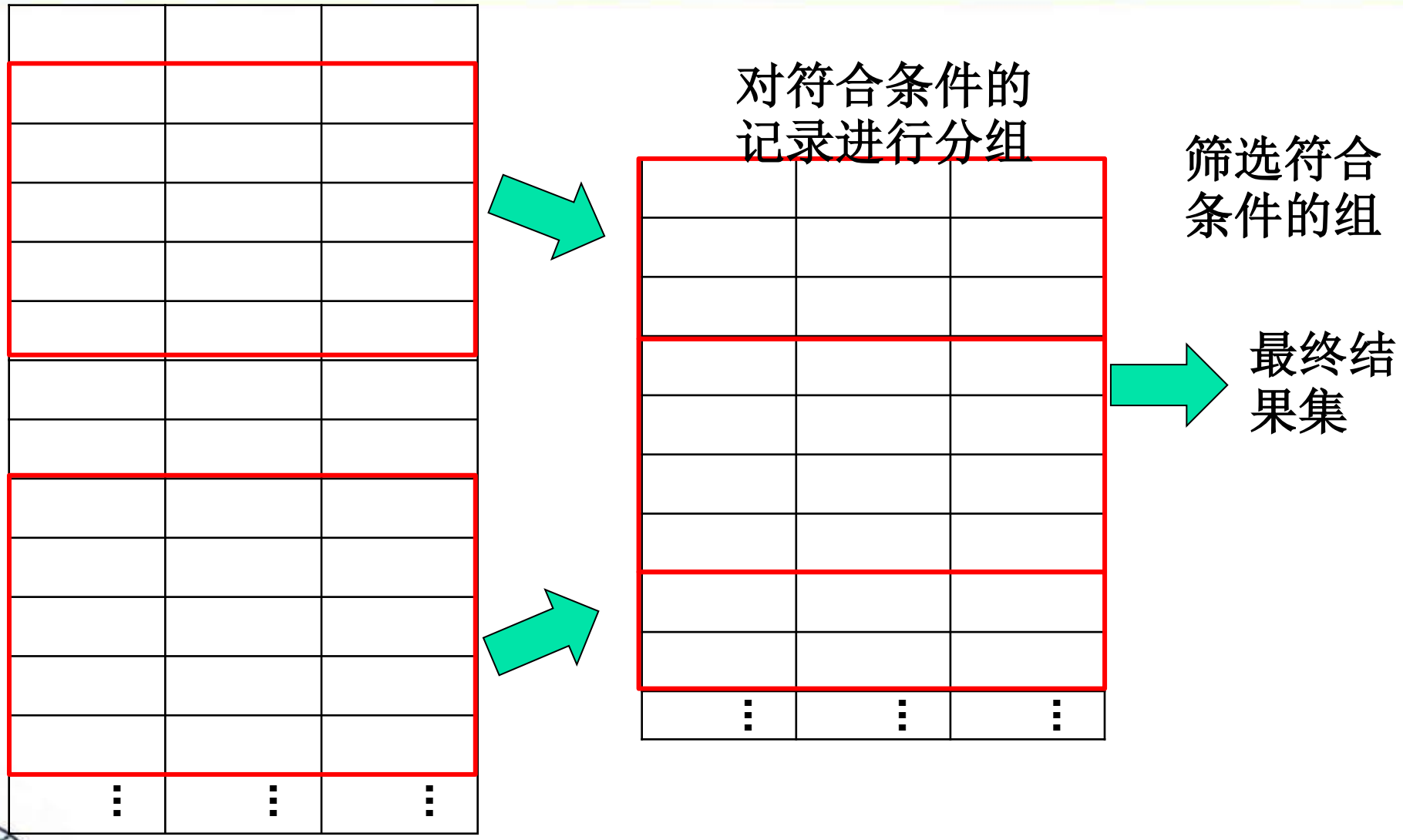
五、HAVING子句

- 使用HAVING子句筛选最终输出结果

[例]查询学号在091501~091506之间至少选修了3门课程的学生的学号和选修课程的课程数。

```
SELECT 学号,COUNT（课程号）AS 选课数
FROM 学习
WHERE 学号 BETWEEN ‘091501’ AND ‘091506’
GROUP BY 学号
HAVING  COUNT(课程号)>=3;
```

对表中记录进行选择



HAVING子句（续）

- 只有满足HAVING短语指定条件的组才输出
- HAVING短语与WHERE子句的区别：作用对象不同
  - WHERE子句作用于基表或视图，从中选择满足条件的元组。
  - HAVING短语作用于组，从中选择满足条件的组。

[例]查询不及格门次超过5门的学生的学号。

```
SELECT 学号
FROM 学习
WHERE 成绩<60
GROUP BY  学号
HAVING COUNT(*)>5;
```

集函数的作用对象

- 如果没有where子句，集函数作用对象是整张表；
- 如果有where子句，集函数的作用对象是满足条件的那些行；
- 如果有分组，那么集函数的作用对象是每个组。

六、排序

使用ORDER BY子句

- 可以按一个或多个属性列排序
- 升序：ASC；降序：DESC；缺省值为升序

当排序列含空值时

- ASC：排序列为空值的元组最先显示
- DESC：排序列为空值的元组最后显示

对查询结果排序（续）

[例]查询选修了180102号课程的学生学号和成绩，
查询结果按成绩从高到低排列。

```
SELECT  学号,成绩
FROM  学习
WHERE 课程号='180102'
ORDER BY 成绩 DESC;
```

对查询结果排序（续）

[例]查询全体学生情况，查询结果按所在学院的名称升序排列，对同一学院中的学生按年龄降序排列。

```
SELECT *
FROM 学生
ORDER BY 学院 ASC , year(now())-出生年份DESC;
```

LIMIT子句

- LIMIT 子句用于限制 SELECT 语句返回指定的记录数
- LIMIT [offset,] count
  - 第一个参数指定返回记录行的偏移量
  - 第二个参数指定返回记录行的最大数目
  - 初始记录行的偏移量是 0（而不是 1）

LIMIT子句

- select * from table limit 5;

  #等价于select * from table limit 0,5;

- select * from table limit 5,10;

  #返回第6-15行数据

- select * from table limit 1,1;

LIMIT子句

[例]查询课程平均分位于前五名的课程，罗列出课程号和平均分。

```
SELECT 课程号, avg(成绩) as 平均分
FROM 学习
GROUP BY  课程号
ORDER BY 平均分 DESC
LIMIT 5;
```

条件分支

形式一：用于等值判断

```
CASE expression
    WHEN   value1   THEN   result1
    WHEN   value2   THEN   result2
    ...
    ELSE   default_result
END
```

条件分支

形式二：用于范围判断

```
CASE
    WHEN  condition1  THEN  result1
    WHEN  condition2  THEN  result2
    ...
    ELSE  default_result
END
```

条件分支（续）

【例】设教师表结构如下：

教师（职工号，姓名，职称，出生年份，学院）

查询每个教师的姓名以及年龄等级。

教师按照年龄划分为三个年龄等级：

- 50岁以上为 “老教师”
- 50岁以下，35 岁以上为 “中年教师”
- 35岁及以下信息为 “年轻教师”

条件分支（续）

```
SELECT 姓名,
   CASE
       WHEN  year(now())-出生年份>=50  THEN  '老教师'
       WHEN  year(now())-出生年份> 35    THEN  '中年教师'
       ELSE    '年轻教师'
   END as 年龄等级
FROM  教师;
```

条件分支（续）

【例】假设有一个员工表：employees(id, name, salary, department)
现在为每个员工计算奖金，规则如下：如果员工的工资大于 10000且部门为Sales，奖金为工资的20% 。如果员工的工资大于 10000 且部门为 HR，奖金为工资的 15%。其他情况奖金为工资的 10%。

```
SELECT   id, name, salary, department,
   CASE
      WHEN salary>10000 AND department='Sales' THEN salary * 0.20
      WHEN salary>10000 AND department = 'HR'  THEN salary * 0.15
       ELSE salary * 0.10
   END AS bonus
FROM employees;
```