# 4

# 第四讲 SQL简单查询

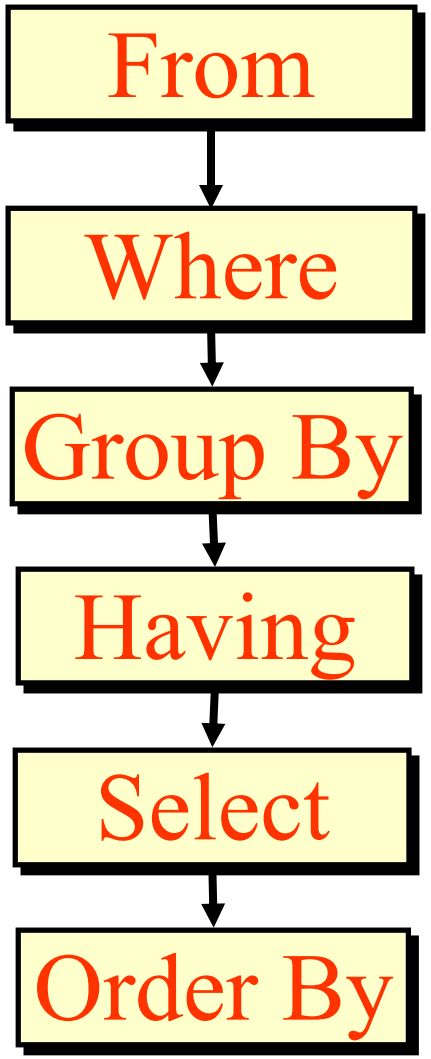
# 教学内容

- 4.1 Select查询语句结构
- 4.2 SELECT子句
  - Distinct 、AS
- 4.3 FROM子句
- 4.4 WHERE子句
  - =、IN、Between、Like、IS NULL
- 4.5 GROUP BY子句
  - Sum、Avg、Max、Min、Count
- 4.6 HAVING子句
- 4.7 ORDER BY子句
  - ASC、DESC

# 4.1 Select查询语句结构

```
SELECT select_list
FROM table_list
[WHERE search_conditions]
[GROUP BY group_by_list]
[HAVING search_conditions]
[ORDER BY order_list [ASC | DESC]]
[LIMIT [offset, ] count];
```

# （1）Select语句的执行过程



# 4.2 SELECT子句

- SELECT [ ALL | DISTINCT ]
  < select_list >
- < select_list > ::=

{ * | { table_name | view_name | table_alias }.*
| { column_name | expression}
[ [ AS ] column_alias ]
} [ ,...n ]

# 简单查询示例

- 查询学生信息

```
Select *
From StudInfo;
```

```
Select StudNo,StudName,StudGender,
       StudBirthDay,ClassID
From StudInfo;
```

- 查询学生部分信息

```
Select StudNo,StudName,ClassID
From StudInfo;
```

# 简单查询示例

- 查询学生信息

```
Select concat(StudNo,StudName), StudGender,
       StudName,ClassID
From StudInfo;
```

- 查询字段计算成绩

```
Select StudNo,CourseID,
       StudScore+5,StudScore*0.8
From StudScoreInfo;
```

- 查询常量计算

```
Select StudNo,concat('姓名:',StudName), 2.65*0.7
From StudInfo;
```

# DISTINCT

- 查询学生性别

```
Select Distinct StudGender
From StudInfo;
```

- 查询学生姓名

```
Select Distinct StudName
From StudInfo;
```

- 查询姓名性别

```
Select Distinct StudName,StudGender
From StudInfo;
```

# 使用别名示例

- 查询学生信息，别名运算

```
Select StudNo As 学号,
       StudName,
       ClassID 班级编号
From StudInfo ;
```

```
Select concat(StudNo, StudName) As 学号姓名,
       concat('学生性别:', StudGender) 性别
From StudInfo;
```

```
Select StudInfo.StudNo 学号,
       StudInfo.StudName as 姓名
From StudInfo;
```

# 4.3 FROM子句

- 语法
  - **[ FROM { < table_source > } [ ,...n ] ]**
- 功能
  - **FROM**子句主要用来指定检索数据的来源，指定数据来源的数据表和视图的列表
  - 列表中的数据表名和视图名之间使用逗号分隔

# 使用表别名

- 使用表别名查询学生信息

```
Select StudInfo.StudNo ,StudInfo.StudName
From StudInfo
```

```
Select S.StudNo,S.StudName
From StudInfo AS S;
```

```
Select S.StudNo 学号,
       S.StudName As 姓名,
       ClassID
From StudInfo S;
```

# 4.4 WHERE子句

- 语法
  - [ WHERE *[binary]* < search_condition >]
- 功能
  - WHERE子句用于限制返回的行的搜索条件，binary 区别大小写。
- 查询或限定条件可以是：
  - 比较运算符（如=、<>、<和>）
  - 范围说明（BETWEEN和NOT BETWEEN）
  - 可选值列表（IN、NOT IN）
  - 模式匹配（LIKE和NOT LIKE）
  - 是否为空值（IS NULL和IS NOT NULL）
  - 上述条件的逻辑组合（AND、OR、NOT）

# 4.4.2 比较查询条件

- 由表达式的双方和比较运算符组成
- 根据查询条件的真假查询某一条记录

| 运算符 | 含义 |
| --- | --- |
| = | 等于 |
| > | 大于 |
| < | 小于 |
| >= | 大于等于 |
| <= | 小于等于 |
| <>, != | 不等于 |

# 比较运算符示例

- 查询90以上的成绩信息

```
Select *
From StudScoreInfo
Where StudScore>=90;
```

- 查询学号为20191152010的学生信息

```
Select *
From StudInfo
Where StudNo='20191152010' ;
```

```
Select *
From StudInfo
Where StudNo>'20191152010' ;
```

# 比较运算符示例

- 按生日查询学生信息

```
Select *
From StudInfo
Where StudBirthDay='2001/10/08' ;
```

```
Select *
From StudInfo
Where StudBirthDay>'2001-10-08' ;
```

- 查询性别不为男的学生信息

```
Select *
From StudInfo
Where StudGender<>'男' ;
```

# 区分大小写

- 查询课程编号为yyj2003的学生成绩

```
select *
from studscoreinfo
where CourseID='yyj2003';
```

- Binary区分大小写

```
select *
from studscoreinfo
Where binary CourseID='yyj2003';
```

# 逻辑运算符（AND、OR和NOT）

- **AND**
  - 连结两个布尔型表达式
  - 当两个表达式都为真时返回真
- **OR**
  - 将两个条件结合起来
  - 结果为两个条件的并集
- **NOT**
  - 用于反转查询条件的结果
- 括号改变逻辑运算符优先级别

# And示例

- 使用And运算符查询学生成绩

```
Select *
From StudScoreInfo
Where StudScore>=60 and StudScore<=70 ;
```

```
Select *
From StudScoreInfo
Where StudNo='20191152010' And StudScore>=80 ;
```

```
Select *
From StudScoreInfo
Where StudNo='20191152010' And
      StudScore>=90 And StudScore<=100 ;
```

# OR示例

- 使用Or运算符查询学生信息

```
Select *
From StudInfo
Where StudNo='20191152010' Or StudName='赵晨' ;
```

- 使用Or运算符查询学生成绩信息

```
Select *
From StudScoreInfo
Where StudScore>80 or StudScore<70 ;
```

# NOT示例

- 使用NOT运算符查询成绩

```
Select *
From StudScoreInfo
Where not StudScore>90 ;
```

- NOT与AND优先级别

```
Select *
From StudScoreInfo
Where NOT StudScore>80 AND StudScore<90 ;
```

# 运算符示例

- 使用NOT、OR运算符查询成绩

```
Select *
From StudScoreInfo
Where not StudScore>80 or StudScore<90;
```

- 使用括号改变优先级别

```
Select *
From StudScoreInfo
Where not (StudScore>80 or StudScore<90);
```

# 4.4.3 范围查询条件

- 内含范围条件(BETWEEN…AND…)
  - 返回某个字段的值在两个指定值范围内的记录，同时包括这两个指定的值。
  - 
```
SELECT * FROM StudScoreInfo
Where StudScore Between 70 And 80;
```
- 排除范围条件(NOT BETWEEN…AND…)
  - 返回某个字段的值在两个指定值范围以外的记录，并不包括这两个指定的值
  - 
```
SELECT * FROM StudScoreInfo
Where StudScore Not Between 70 And 80 ;
```

# 范围查询条件示例

- 查询学生成绩在70到80之间的学生成绩

```
SELECT *
FROM StudScoreInfo
Where StudScore Between 70 And 80 ;
```

```
SELECT *
FROM StudScoreInfo
Where StudScore>=70 And StudScore<=80 ;
```

# 范围查询条件示例

- 查询学生成绩不在70到80之间的学生成绩

```
SELECT *
FROM StudScoreInfo
Where StudScore NOT Between 70 And 80 ;
```

```
SELECT *
FROM StudScoreInfo
Where StudScore<70 Or StudScore>80 ;
```

# 范围查询示例

- 查询学号为20191152010成绩在[80,90]之间成绩信息

```
Select *
From StudScoreInfo
Where StudNo='20191152010' And
      StudScore Between 80 And 90 ;
```

```
Select *
From StudScoreInfo
Where StudScore>=80 And StudScore<=90
      And StudNo= '20191152010' ;
```

# 范围查询示例

- 查询成绩在[60,70]和[80,90]之间的成绩信息

```
Select *
From StudScoreInfo
Where StudScore Between 60 And 70 Or
      StudScore Between 80 And 90 ;
```

```
Select *
From StudScoreInfo
Where StudScore>=60 And StudScore<=70 Or
      StudScore>=80 And StudScore<=90 ;
```

# 4.4.4 列表查询条件（IN)

- 格式
  - IN(列表值1,列表值2,…)
- 功能
  - 返回所有与列表中的任意一个值匹配的记录。
- 示例

```
SELECT *
FROM StudInfo
Where StudNo IN('20191152010','20191152041') ;
```

```
SELECT *
FROM StudInfo
Where StudNo='20191152010' Or StudNo='20191152041' ;
```

# 4.4.5 模式查询条件

- 模式查询条件常用来返回符合某种格式的所有记录。
- 通常使用LIKE或NOT LIKE关键字来指定模式查询条件。
- LIKE关键字使用通配符来表示字符串需要匹配的模式。

| 通配符 | 描述 | 示例 |
| --- | --- | --- |
| % | 零个或更多字符的任意字符串 | WHERE StudName LIKE '%丽%' |
| _ | 任何单个字符 | WHERE StudName LIKE '_丽' |

# LIKE示例

- 查询姓“李”的学生信息

```
Select *
From StudInfo
Where StudName Like '李%';
```

- 查询姓名中有“丽”字的学生信息

```
Select *
From StudInfo
Where StudName Like '%丽%' ;
```

```
Select *
From StudInfo
Where StudName Like '_丽' ;
```

# 4.4.6 空值判断查询条件

- 空值判断查询条件
  - 关键字：IS NULL、IS NOT NULL
  - 用来搜索某一字段值为空值的记录
- 示例

```
Select *
From ClassInfo
Where ClassDesc is null ;
```

```
Select *
From ClassInfo
Where ClassDesc is Not null ;
```

# 4.4.7 正则表达式

- **REGEXP 操作符正则表达式匹配**

| 模式 | 示例 | 描述 |
| --- | --- | --- |
| ^ | ‘^st’ | 匹配输入字符串的开始位置。设置RegExp 对象的 Multiline 属性，^ 也匹配 ‘\n’ 或 ‘\r’ 之后位置。 |
| $ | 'ok$' | 匹配输入字符串的结束位置。设置RegExp 对象的 Multiline 属性，$ 也匹配 '\n' 或 '\r' 之前位置。 |
| . | | 匹配除 "\n" 之外的任何单个字符。要匹配包括 '\n' 在内的任何字符，请使用像 '[.\n]' 的模式。 |
| [...] | '[abc]' | 字符集合。匹配所包含的任意一个字符。可以匹配 "plain" 中的 'a'。 |
| [^...] | '[^abc]' | 负值字符集合。匹配未包含的任意字符。可以匹配 "plain" 中的'p'。 |
| p1\|p2\|p3 | 'z\|food' | 匹配 p1 或 p2 或 p3。匹配 "z" 或 "food"。'(z\|f)ood' 则匹配 "zood" 或 "food"。 |
| * | ‘zo*’ | 匹配零次或多次。匹配 "z" 以及 "zoo"。* 等价于{0,}。 |
| + | 'zo+' | 匹配一次或多次。匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。 |
| {n} | 'o{2}' | 匹配 n 次。匹配 “food” 中的两个 o，不能匹配 "Bob" 中的 'o'。 |
| {n,m} | 'o{2，5}' | m 和 n 均为非负整数，其中n <= m。最少匹配 n 次且最多匹配 m 次。 |

# REGEXP示例

- select *
- from studscoreinfo
- where courseid regexp 'DX[PS]'

- select *
- from studinfo
- where studname regexp '丽$'

# 算术运算符

| 运算符 | 作用 |
| --- | --- |
| + | 加法 |
| - | 减法 |
| * | 乘法 |
| / 或 DIV | 除法 |
| % 或 MOD | 取余 |

- 除：select 2/3;
- 商：select 10 DIV 4;
- 取余：select 10 MOD 4;

# 比较运算符

| 符号 | 描述 |
| --- | --- |
| = | 等于 |
| <>, != | 不等于 |
| > | 大于 |
| < | 小于 |
| <= | 小于等于 |
| >= | 大于等于 |
| BETWEEN | 在两值之间>=min&&<=max |
| NOT BETWEEN | 不在两值之间 |
| IN | 在集合中 |
| NOT IN | 不在集合中 |
| <=> | 严格比较两个NULL值是否相等，均为NULL，值为1，否则为0 |
| LIKE | 模糊匹配 |
| REGEXP 或 RLIKE | 正则式匹配 |
| IS NULL | 为空 |
| IS NOT NULL | 不为空 |

# 比较运算符示例

- select 2=3;
- select NULL = NULL;
- select null<=>null;
- select 2<>3;
- select 2<=>3;
- select 5 between 1 and 10;
- select 5 in (1,2,3,4,5);
- select null is NULL;
- select '12345' like '12%';
- select 'beijing' REGEXP 'jing';

# 逻辑运算符

| 运算符号 | 作用 |
| --- | --- |
| NOT 或 ! | 逻辑非 |
| AND | 逻辑与 |
| OR | 逻辑或 |
| XOR | 逻辑异或 |

- 与：select 2 and 0;
- 或：select 2 or 0;
- 非：select not 1;
- 异或：select 1 xor 0;

# 位运算符

| 运算符号 | 作用 |
| --- | --- |
| & | 按位与 |
| \| | 按位或 |
| ^ | 按位异或 |
| ! | 取反 |
| << | 左移 |
| >> | 右移 |

- 按位与：select 3&5;
- 按位或：select 3|5;
- 按位异或：select 3^5;
- 按位取反：select !15;
- 按位右移：select 3>>1;
- 按位左移：select 3<<1;

# 运算符优先级

| 优先级顺序 | 运算符 |
| --- | --- |
| 1 | := |
| 2 | \|\|, OR, XOR |
| 3 | &&, AND |
| 4 | NOT |
| 5 | BETWEEN, CASE, WHEN, THEN, ELSE |
| 6 | =, <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN |
| 7 | \| |
| 8 | & |
| 9 | <<, >> |
| 10 | -, + |
| 11 | *, /, DIV, %, MOD |
| 12 | ^ |
| 13 | - (一元减号), ~ (一元比特反转) |
| 14 | ! |

- 最低优先级为： := 最高优先级为： !、BINARY、 COLLATE

# 4.5 GROUP BY子句

- 语法
  - **[ GROUP BY *group_by_expression* [ ,...*n* ] WITH ROLLUP]**
- 功能
  - 按指定的条件进行分类汇总。
  - 如果**SELECT**子句包含聚合函数，则计算每组的汇总值。
- 注意
  - **group_by_expression**是对其执行分组的表达式或列，且分组列不能使用列别名。
  - 在使用**Group By**子句，只有聚合函数和分组字段才能出现在**Select**子句中。

# 汇总函数

| 聚合函数 | 结果 |
| --- | --- |
| SUM([ALL \| DISTINCT] expression) | 数字表达式中所有值的和 |
| AVG([ALL \| DISTINCT] expression) | 数字表达式中所有值的平均值 |
| COUNT([ALL \| DISTINCT] expression) | 表达式中值的个数 |
| COUNT(*) | 选定的行数 |
| MAX(expression) | 表达式中的最高值 |
| MIN(expression) | 表达式中的最低值 |

# 聚合函数示例

- 统计所有成绩平均分

```
Select Avg(StudScore)
From StudScoreInfo ;
```

- 统计学号为20191152010的平均分、课程门数

```
Select Avg(StudScore)
From StudScoreInfo
Where StudNo='20191152010' ;
```

```
Select Count(*)
From StudScoreInfo
Where StudNo='20191152010' ;
```

# 聚合函数示例

- 统计学号为20191152010的成绩信息

```
Select Max(StudScore),
       Min(StudScore),
       Sum(StudScore),
       Avg(StudScore) As AvgScore,
       Count(*) CourseCount
From StudScoreInfo
Where StudNo='20191152010' ;
```

# 聚合函数示例

- 统计学号为20191152010的成绩信息

```
Select Max(StudScore),
       Min(StudScore),
       Sum(StudScore),
       Avg(StudScore),
       Count(*),
       Sum(StudScore)/Count(*) AvgScore
From StudScoreInfo
Where StudNo='20191152010' ;
```

# Group By和聚合函数示例

- 统计各学生平均分

```
Select StudNo,
        Avg(StudScore) AvgScore
From StudScoreInfo
Group By StudNo ;
```

- 统计各学生平均分，保留1位小数

```
Select StudNo,
        Cast(Avg(StudScore) As Decimal(4,1)) AvgScore
From StudScoreInfo
Group By StudNo ;
```

# Group By和聚合函数示例

- 统计各学生所学课程门数

```
Select StudNo,
        Count(*) CourseCount
From StudScoreInfo
Group by StudNo ;
```

```
Select StudNo,
        Count(*) CourseCount,
        Cast(Avg(StudScore) As Decimal(4,1))
            AS  AvgScore
From StudScoreInfo
Group By StudNo ;
```

# Group By和聚合函数示例

- 统计各学生总分、平均分、课程门数

```
Select StudNo,
        Sum(StudScore) As SumScore,
        Count(*) CourseCount,
        Avg(StudScore)  As AvgScore1,
        Sum(StudScore)/Count(*) As AvgScore2
From StudScoreInfo
Group By StudNo ;
```

# WITH ROLLUP

- 在分组统计数据基础上再进行相同的统计（SUM,AVG,COUNT…）
- 统计各学生总分、平均分、课程门数

```
Select StudNo, Count(*) CourseCount, Avg(StudScore) AvgScore
From StudScoreInfo
Group By StudNo
with rollup;
```

- COALESCE更改NULL统计字段值名

```
Select COALESCE(StudNo,'总计'),
        Count(*) CourseCount, Avg(StudScore) AvgScore
From StudScoreInfo Group By StudNo with rollup;
```

# 4.6 HAVING子句

- **HAVING**
  - 对分组统计后的结果再次筛选
  - **HAVING语法与WHERE语法类似**
- **WHERE和HAVING**
  - **WHERE在分组之前进行筛选。**
  - **HAVING在分组之后进行筛选。**
  - **HAVING可以包含聚合函数。**
  - **HAVING可以引用选择列表中出现的任意项。**

# Having示例

- 统计学生平均分在80以上的成绩信息

```
Select StudNo,
        Sum(StudScore) As SumScore,
        Count(*) CourseCount,
        Cast(Avg(StudScore) As Decimal(4,1))
            AS AvgScore
From StudScoreInfo
Group By StudNo
Having Avg(StudScore)>=80;
```

# Where示例

- 统计课程成绩在80分以上的成绩信息

```
Select StudNo,
        Sum(StudScore) As SumScore,
        Count(*) CourseCount,
        Cast(Avg(StudScore) As Decimal(4,1))
            AS  AvgScore
From StudScoreInfo
Where StudScore>=80
Group By StudNo;
```

# 4.7 ORDER BY子句

- 语法
  - **[ORDER BY order_list [ASC | DESC]]**
- 功能
  - 将根据查询结果中的一个字段或多个字段对查询结果进行排序
  - **ASC为升序， DESC为降序**
- 注意
  - 在**ORDER BY** 列表中不允许使用子查询、聚合表达式或常量表达式
  - **ORDER BY**子句可以使用列的别名

# ORDER BY示例

- 将学号为20191152010成绩升序排列

```
SELECT *
FROM StudScoreInfo
Where StudNo='20191152010'
Order by StudScore;
```

- 将学号为20191152010成绩按高低排序

```
SELECT *
FROM StudScoreInfo
Where StudNo='20191152010'
Order by StudScore Desc;
```

# ORDER BY示例

- 查询学号为20191152010成绩信息

```
SELECT *
FROM StudScoreInfo
Where StudNo='20191152010'
Order by StudScore Desc,CourseID ASC;
```

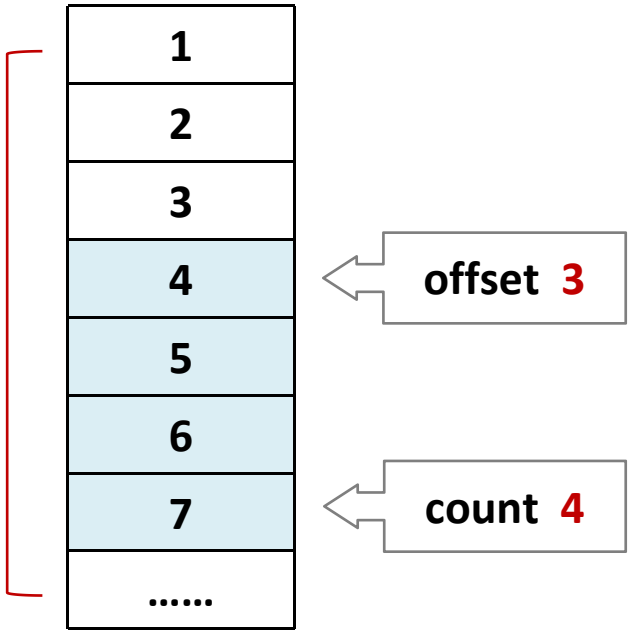
- 将平均分按高低排序

```
Select StudNo,
        Avg(StudScore) As  AvgScore
From StudScoreInfo
Where StudScore>=80
Group By StudNo
Order By AvgScore Desc;
```

# Limit子句

- **SELECT** 列名
- **FROM** 表名
- **LIMIT** [起始行号], 返回行数

```
SELECT *
FROM table;
```




```
SELECT *
FROM table
LIMIT 3, 4;
```

# Limit示例

```
SELECT *
FROM StudInfo
LIMIT 5
```


```
SELECT *
FROM StudInfo
LIMIT 0,5
```

- **SELECT ***
- **FROM StudScoreInfo**
- **WHERE StudNo='20180712001'**
- **ORDER BY StudScore DESC**
- **LIMIT 5;**

```
SELECT StudNo,Avg(StudScore) AvgScore
FROM StudScoreInfo
GROUP BY StudNo
ORDER BY AvgScore DESC
LIMIT 5;
```

# 下次课内容

- **Where关联表**
  - 关联表查询
  - 关联表统计
- **Union查询**
- 子查询
  - **In、Some/Any、ALL、Exists**
- 连接查询
  - 内联接、外联接（左、右联接）
  - 自联接
- 经典**SQL**语句
  - 批量插入、关联表更新