link: null
title: 珠峰架构师成长计划
description: 对一组值进行计算，并返回计算后的值，一般用来统计数据
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=92 sentences=85, words=756

1. 什么是聚合函数

对一组值进行计算，并返回计算后的值，一般用来统计数据

1.1 SUM

累加所有行的值

```
计算ID=1的学生的的总分
select SUM(grade)  as '总分' from score where student_id = 1;
```

1.2 AVG

计算所有行的平均值

```
计算ID=1的学生的的平均分
select AVG(grade) as '平均分' from score where student_id = 1;
```

1.3 MAX、MIN

- 计算所有行的最大值和最小值

```
select MAX(grade) 最高分,MIN(grade) 最低分 from score where student_id = 1;
```

1.4 AVG

- 计算所有行的平均值

```
select AVG(grade) as '平均分' 最低分 from score where student_id = 1;
```

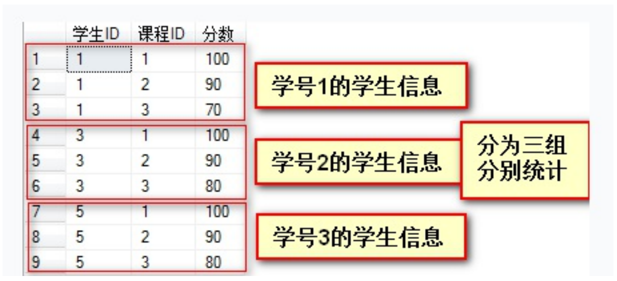
1.5 COUNT

- 计算值不为NULL的行

```
select COUNT(*) from student;
select COUNT(1) from student;
select COUNT(name) from student;
select COUNT(NULL) from student;
```

2. 分组

分组查询就是按某列的值进行分组，相同的值分成一组，然后可以对此组内进行求平均、求和等计算



	学生ID	课程ID	分数
1	1	1	100
2	1	2	90
3	1	3	70
4	3	1	100
5	3	2	90
6	3	3	80
7	5	1	100
8	5	2	90
9	5	3	80

学号1的学生信息
学号2的学生信息
分为三组
分别统计
学号3的学生信息

2.1 语法

```
SELECT 列名,查询表达式
FROM
WHERE
GROUP BY
HAVING 分组后的过滤条件
ORDER BY 列名 [ASC,DESC]
LIMIT 偏移量,条数
```

SELECT列表中只能包含：

- 被分组的列
- 为每个分组返回一个值的表达式，如聚合函数

2.2 练习

- 统计每位同学的平均成绩-单列分组

```
select student_id,avg(grade) from score group by student_id;
```

- 统计每门课程的最高分，并按分数从高到低排列

```
select course_id,max(grade) 平均分 from score group by course_id order by max(grade) desc
```

- 统计各省的男女同学人数-多列分组

```
select province,gender,COUNT(*) from student group by province,gender
```

2.3 分组筛选

2.3.1 语法

```
SELECT  FROM
WHERE
GROUP BY {col_name|expr|position}
HAVING  {col_name|expr|position}
ORDER BY {col_name|expr|position} [ASC|DESC]
LIMIT offset,row_count
```

1. WHERE用于过滤掉不符合条件的记录
2. HAVING 用于过滤分组后的记录
3. GROUP BY用于对筛选后的结果进行分组

2.3.2 练习

- 统计学生人数超过1人的省份

```
select province,COUNT(*) from student group by province having COUNT(*)>1
```

- 不及格次数大于1次的学生

```
select student_id,COUNT(*) 不及格次数 from score where grade <60 group by student_id having COUNT(*)>1
```

3. 子查询

- 子查询就是指出现在其它SQL语句中的SELECT语句,必须始终出现在圆括号中
- 子查询可以包含多个关键字或条件
- 子查询的外层查询可以是: SELECT、INSERT、UPDATE、SET等
- 子查询可以返回常量、一行数据、一列数据或其它子查询

3.1 比较运算符的子查询

- = 等于
- > 大于
- < 小于
- >= 大于等于

3.2 查询年龄大于平均年龄的学生

```
SELECT ROUND(AVG(age),2) FROM student;

SELECT * from student WHERE age > (SELECT ROUND(AVG(age),2) FROM student)
```

3.2 ANY SOME ALL

-

```
= <
```

- ANY 任何一个
- SOME 某些
- ALL 全部

```
年龄大于陕西省任何一位同学
SELECT * from student WHERE age > ANY (SELECT age  FROM student WHERE province = '陕西省');
年龄大于陕西省某些同学
SELECT * from student WHERE age > SOME (SELECT age  FROM student WHERE province = '陕西省');
年龄大于陕西省所有同学
SELECT * from student WHERE age > ALL (SELECT age  FROM student WHERE province = '陕西省');
```

3.3 查询一下有考试成绩的学生信息

- [IN]
- [NOT IN]

```
SELECT * FROM student where id in (SELECT distinct student_id from score);
```

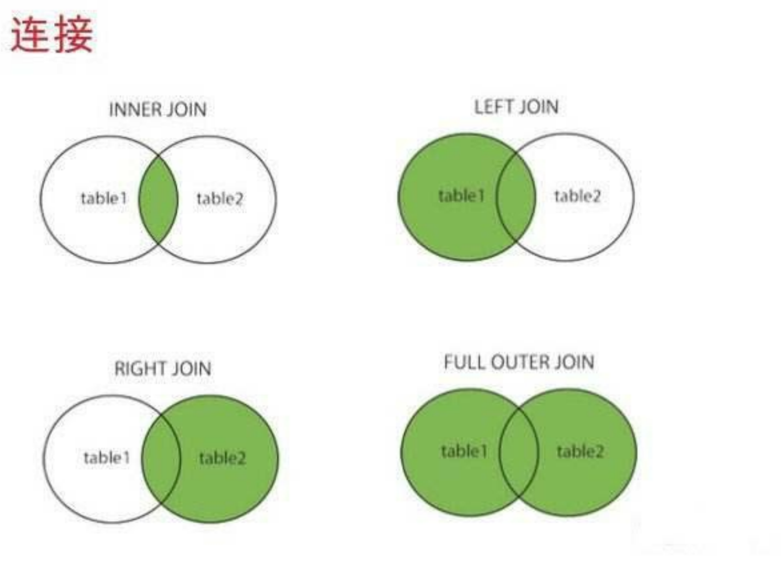
- [EXISTS]
- [NOTEXISTS]

```
SELECT * FROM student where EXISTS (SELECT distinct student_id from score where student.id = score.student_id  )
```

4. 表连接

4.1 连接类型

- INNER JOIN 内连接
- LEFT JOIN 左外连接
- RIGHT JOIN 右外连接
- ON 连接条件



4.2 连接条件

使用ON关键字来设定连接条件，也可以使用WHERE来代替

- ON来设定连接条件
- 也可以使用WHERE来对结果进行过滤

4.3 内连接

显示左表和右表中符合条件的

```
SELECT * FROM student INNER JOIN score ON student.id = score.student_id;
```

4.4 左外连接

显示左表的全部和右表符合条件的

```
SELECT * FROM student LEFT JOIN score ON student.id = score.student_id;
```

4.5 右外连接

显示右表的全部和左表符合条件的

```
SELECT * FROM student RIGHT JOIN score ON student.id = score.student_id;
```

4.6 多表连接

```
SELECT student.name,course.name,score.grade FROM score
INNER JOIN student ON student.id = score.student_id
INNER JOIN course ON course.id = score.course_id;
```

4.7 无限分类[自身连接]

4.7.1 建表

```
CREATE table category(
 id int(11) PRIMARY KEY AUTO_INCREMENT NOT NULL,
 name varchar(50),
 parent_id int(11)
)
```

4.7.2 插入语句

```
INSERT INTO category(id,name,parent_id)
VALUES (1,'数码产品',0),(2,'服装',0),(3,'食品',0),
(4,'iPad',1),(5,'李宁',2),(6,'康师傅',3);
```

4.7.3 查询所有的顶级分类下面分类的数量

```
SELECT c1.id,c1.name,COUNT(1)
FROM category c1 INNER JOIN category c2 ON c1.id = c2.parent_id
WHERE c1.parent_id = 0
GROUP BY c1.id;
```

4.7.4 父类变成名称

```
SELECT c1.id,c1.name,p.name
FROM category c1 LEFT JOIN category p ON c1.parent_id = p.id
```

4.8 删除重复记录[多表删除]

```
INSERT INTO category(id,name,parent_id)
VALUES
(7,'iPad',1),
(8,'李宁',2),
(9,'康师傅',3);
```

4.8.1 子查询找要删除的ID

```
SELECT * FROM category c1 LEFT JOIN
(SELECT id,name from category GROUP BY name HAVING COUNT(1)>1) c2
ON c1.name = c2.name WHERE c1.id != c2.id
```

4.8.2 通过IN找要删除的ID

```
SELECT * FROM category c1
WHERE c1.name IN
(SELECT name from category GROUP BY name HAVING COUNT(1)>1)
AND c1.id NOT IN
(SELECT MIN(id) from category GROUP BY name HAVING COUNT(1)>1)
```

4.8.3 删除重复记录

```
DELETE FROM category
WHERE name IN
(SELECT NAME FROM ( SELECT name from category GROUP BY name HAVING COUNT(1)>1) AS T1 )
AND id NOT IN
(SELECT id FROM (SELECT MIN(id) id from category GROUP BY name HAVING COUNT(1)>1) AS T2)
```

4.9 多表更新

4.9.1 (插入省份)INSERT SELECT#

```
CREATE TABLE province(id int PRIMARY KEY AUTO_INCREMENT,name varchar(50))
INSERT INTO province(name) SELECT DISTINCT province FROM student;
```

4.9.2 更新省份

```
UPDATE student INNER JOIN province ON student.province=province.name
 SET student.province=province.id
```

4.9.3 修改字段

```
ALTER TABLE student
CHANGE COLUMN `province` `province_id` int(11);
```