MySQL数据库第五部分讲义

重点掌握子查询

一 理解并口述（技术要点和面试题）

【口述1】什么是子查询?

【口述2】什么时候使用子查询？

【口述3】子查询的编写思路

【口述4】使用子查询的注意事项

【口述5】常规SQL语句优化

【口述6】什么是索引?

【口述7】索引的作用有哪些?

【口述8】索引的分类有哪些?

【口述9】创建索引的原则有哪些?

二 动手做（TODO)

(一) 子查询

1. 单行子查询 （TODO 4 个任务）

(1). 在WHERE子句中的单行子查询

(2). 在HAVING子句中的单行子查询

2. 多行子查询(TODO 8个任务)

2.1 多行子查询中使用IN操作符

2.2 多行子查询中使用ANY操作符

2.3 多行子查询中使用ALL操作符

2.4 子查询中使用EXISTS操作符

3. 多列子查询（TODO 1个任务）

4. 关联子查询 （TODO 2个任务）

5. 在DDL的建表语句中使用子查询 (TODO 2个任务)

6. 在SELECT语句中使用子查询 （TODO 1个任务）

7. 插入语句中使用子查询 （TODO 1个任务）

8. FROM子句中使用子查询 （TODO 2个任务）

9. 合并查询结果（UNION,UNION ALL) (TODO 2个任务)

(二) 索引 （TODO 2个任务）

1. 如何创建索引?

2. 如何删除索引?

三 课后任务 (TODO 9个任务)

四 拓展（了解MySQL数据库中的触发器，存储函数和存储过程）

4.1 什么是触发器?

4.2 如何创建一个触发器?

4.3 什么是存储过程?

4.4 什么是存储函数（自定义函数）？

4.5 存储过程与存储函数（自定义函数）的区别?

4.4 如何创建及调用一个存储过程?

4.5 如何创建及调用一个存储函数?

4.6 如何使用游标?

4.7 如何删除存储过程?

4.8 如何删除存储函数?

MySQL数据库第五部分讲义

重点掌握子查询

一 理解并口述（技术要点和面试题）

【口述1】什么是子查询?

- 子查询是指插入在其他SQL语句中的SELECT语句，也称为嵌套查询。 使用子查询主要是将结果作为外部主查询的查询条件来使用的查询。

【口述2】什么时候使用子查询？

1. 当要显示的数据在表里并不存在，但可以通过对已有数据的加工获得，可通过子查询实现。
2. 子查询可以出现在SELECT、WHERE子句、FROM子句、DML语句、DDL语句中。
3. 在SELECT、INSERT、UPDATE或DELETE命令中允许是一个表达式的地方都可以包含子查询，子查询甚至可以包含在另外一个子查询中。

【口述3】子查询的编写思路

- 子查询的编写思路：
    1. 仔细分析题目，确定要查询的表及字段（数据）
    2. 分析要查询的字段（数据）哪些在表里直接存在，哪些不存在
    3. 考虑如何把要显示的数据造出来（通过查询语句获得）
    4. 考虑子查询与表的连接点是什么(通常是主外键、共有字段)
    5. 考虑子查询放在什么位置
    6. 组合成完整的SQL语句

【口述4】使用子查询的注意事项

```
1．要将子查询放入圆括号中。
2．子查询可出现在WHERE子句、FROM子句、SELECT列表（此处只能是一个单行子查询）和HAVING子句,DDL,DML中。
3．子查询不能出现在主查询的GROUP BY语句中
4．子查询和主查询可以使用不同表，只要子查询返回的结果能够被主查询使用即可。
5．单行子查询只能使用单行操作符，多行子查询只能使用多行操作符。
6．在多行子查询中，ALL和ANY操作符不能单独使用，而只能与单行比较符（=、<、>、<=、>=、<>)结合使用。
7．要注意子查询中的空值问题。如果子查询返回了一个空值，则主查询将不会查询任何结果。
8．在WHERE子句中进行子查询的时候，不能带有GROUP BY子句。
9．子查询允许嵌套多层，但不能超过255层。
```

【口述5】常规SQL语句优化

```
1．建议不用"*"来代替所有列名
2．用TRUNCATE代替DELETE（删除数据表的所有数据时）
3．在确保完整性的情况下多用COMMIT（事务提交）
4．尽量减少表的查询次数
5．用[NOT] EXISTS代替[NOT] IN
```

【口述6】什么是索引?

- 索引：类似“目录”，给一张表添加了一个目录。

- 索引是一个单独的、物理的数据结构，是某个表中一列或若干列值的集合和相应的指向物理标识这些值的数据页的逻辑指针清单。索引依赖于表建立的，提供了编排表中数据的内部方法。
- 目的为了数据的查询效率。

【口述7】索引的作用有哪些?

- 索引的作用：
    1. 为了提高查询效率
    2. 通过快速定位数据的方法，减少磁盘I/O操作

【口述8】索引的分类有哪些?

> 1. 普通索引 不需要添加任何限制条件，可以创建在任何数据类型中，由字段本身的完整性约束决定。
> 2. 唯一索引 使用 unique 参数进行设置，该值必须是唯一的。（主键是一种特殊的唯一索引）
> 3. 全文索引 使用 fulltext 参数进行设置，只能创建在 char,varchar 或者 text 类型的字段 上（适用于查询数据量较大的字符串类型的字符时）
> 4. 单列索引 在表中单个字段上创建，只能根据该字段进行索引查询，只要保证该索引只对应一个字段即可。多列索引在表中多个字段上创建，可根据多个字段进行索引查询（注意：只有查询条件中使用了这些字段中的第一个字段时，索引才会被使用）。例如：id ， name， age，查询条件使用了 id 字段时该索引才会被使用.
> 5. 空间索引（用的比较少） 使用 spatial 参数进行设置，只能建立在空间数据类型上。（geometry、point、linestring 和 polygon 等）

【口述9】创建索引的原则有哪些?

> 1. 选择唯一索引 因为唯一索引的值是唯一的，可快速通过该索引来确定某条记录 例如：人-->身份证号 学生-->学号
> 2. 为经常需要排序、分组和联合操作的字段建立索引 频繁使用 order by、group by、distinct 和 union 等来操作字段时，
> 3. 经常作为查询条件的字段建立索引
>
> WHERE 条件经常使用的字段可以创建索引。可以给外键创建索引。
>
> 4. 限制索引的数目: 索引的数目并不是越多越好，每个索引都要占用磁盘空间，修改表时，对索引的重构和更新比较麻烦。
> 5. 尽量使用数据量少的索引
> 6. 尽量使用前缀来索引 检索值很长时，比如 text、blog，只检索前面的若干个字符
> 7. 不使用或使用频率低的，应尽快删除 **不适合建索引的情况**： (1) 表很小 (2) 字段不经常出现在 WHERE 子句中 (3) 每次访问的数据量大于记录总数的2%~4% (4) 字段经常更新

二 动手做（TODO)

(一) 子查询

1. 单行子查询 （TODO 4 个任务）

(1). 在WHERE子句中的单行子查询

```sql
-- (1)  子查询与主查询使用同一张表:
-- 任务一:查询与SCOTT在同一部门的员工的姓名,薪水
SELECT ename,sal FROM emp WHERE deptno=(SELECT deptno FROM emp WHERE ename='SCOTT');
-- 任务二:查询出工资低于所有员工平均工资的员工姓名和工资,按照工资高低降序排序。
SELECT ename,sal FROM emp WHERE sal < (SELECT AVG(sal) FROM emp) ORDER BY sal DESC;

-- (2)子查询与主查询不是使用的同一张表:
-- 任务三:  查询部门名称为"RESEARCH"的员工信息(显示员工号,姓名,职位)
SELECT empno,ename,job FROM emp WHERE deptno=(SELECT deptno FROM dept WHERE
dname='RESEARCH');
```

(2). 在HAVING子句中的单行子查询

```sql
-- 任务四:查询出各部门员工的平均工资低于各部门最高平均工资的部门号和部门的平均工资。
SELECT deptno,AVG(sal) avgsal FROM emp GROUP BY deptno
HAVING AVG(sal)<(SELECT MAX(e.avgsal)
                 FROM (
                 SELECT AVG(sal) AS avgsal
                 FROM emp GROUP BY deptno) e);
```

> **注意单行子查询中经常遇到的错误:
>
> - 因为WHERE条件限定不规范而返回多行,就会出现单行子查询返回多行的错误。
> - 子查询中不能包含ORDER BY 子句,相反任何排序都必须在外部查询中完成。
> - MySQL中不允许聚合函数直接嵌套:例如 : MAX(AVG(sal)) 会报错,可以写成子查询。必须给出表的别名。Oracle中允许聚合函数直接嵌套。**

2. 多行子查询(TODO 8个任务)

2.1 多行子查询中使用IN操作符

```sql
-- 任务一. 列出薪金与30号部门员工的薪金相同的所有员工的姓名和薪金。
SELECT ename,sal
FROM emp
WHERE sal IN (SELECT sal FROM emp WHERE deptno=30);

-- 标准嵌套子查询多层嵌套
-- 任务二. 列出薪水与销售部门在同部门的员工薪水相同的所有员工的姓名和薪金。
SELECT ename,sal
FROM emp
WHERE sal IN (
          SELECT sal
          FROM emp
          WHERE deptno=(
                        SELECT deptno
                        FROM dept
                        WHERE dname='SALES'));

-- 任务三. 列出至少有4个员工的所有部门信息
SELECT * FROM dept
WHERE deptno IN(SELECT deptno
                FROM emp
```

```
                GROUP BY deptno
                HAVING COUNT(*)>=4);
```

2.2 多行子查询中使用ANY操作符

```
-- 任务四 在emp表中，查询工资大于部门编号为10的任意一个员工工资的其他部门的员工信息。
SELECT * FROM emp WHERE sal>ANY(SELECT sal FROM emp WHERE deptno=10) AND deptno<>10;
```

2.3 多行子查询中使用ALL操作符

```
-- 任务五 显示工资大于所有部门平均工资的雇员姓名,工资。
SELECT ename,sal FROM emp
WHERE sal>ALL(SELECT AVG(sal) FROM emp GROUP BY deptno);

-- 任务六 查询工资大于部门编号为20的所有员工工资的员工信息
SELECT ename,sal FROM emp
WHERE sal>ALL(SELECT sal FROM emp WHERE deptno=20);
```

2.4 子查询中使用EXISTS操作符

```
-- 任务七 查询在“SALES"销售部门的所有员工信息。
SELECT  *
FROM emp e
WHERE  EXISTS(SELECT deptno
               FROM dept  d
               WHERE e.deptno = d.deptno
               AND d.dname='SALES');

-- 任务八 查询在NEW YORK工作的所有雇员的名字、职位、薪水、所在部门。
SELECT ename,job,sal,deptno
FROM emp e
WHERE EXISTS(SELECT deptno
             FROM dept d
             WHERE e.deptno = d.deptno
             AND d.loc='NEW YORK');
```

3. 多列子查询（TODO 1个任务）

```
-- 任务 查询显示和ALLEN同部门同职位的员工姓名、职位、部门编号
SELECT ename,job,deptno FROM emp
WHERE (deptno,job)=(SELECT deptno,job FROM emp WHERE ename='ALLEN');
```

> 说明：子查询结果返回多列。

4. 关联子查询 （TODO 2个任务）

> - 在单行子查询中和多行子查询中，内查询和外查询是分开执行的，也就是说内查询的执行与外查询的执行没有关系，外查询仅仅是使用内查询的最终结果。
> - 但是关联子查询中内查询与外查询是相互关联的。
> - 内查询的执行需要借助于外查询，而外查询的执行又离不开内查询的执行。

```
-- 任务一 在emp表中，使用“关联子查询”检索工资大于同职位的平均工资的员工信息(显示字段：员工编号，姓名，工资）。
SELECT empno,ename,sal FROM emp e
WHERE sal>(SELECT AVG(sal) FROM emp m WHERE m.job=e.job);
-- 任务二 查询所有大于本部门平均工资的员工信息
SELECT * FROM emp e
WHERE sal>(SELECT AVG(sal) FROM emp m WHERE m.deptno=e.deptno);
```

5. 在DDL的建表语句中使用子查询 (TODO 2个任务)

```
-- 任务一  创建一个和部门表一样的表，结构与数据都一样。
CREATE TABLE dept1 AS SELECT * FROM dept;
-- 任务二  创建一个和部门表结构一样的空表,没有数据。
CREATE TABLE dept2 AS SELECT * FROM dept WHERE 1=2;
```

6. 在SELECT语句中使用子查询 （TODO 1个任务）

```
-- 任务 统计出有奖金和没有奖金的人数
SELECT DISTINCT (SELECT COUNT(comm) FROM emp WHERE comm<>0) "有奖金人数",
                (SELECT COUNT(*) FROM emp WHERE comm=0 OR comm IS NULL)                  "没奖金人数"
FROM emp;
```

7. 插入语句中使用子查询 （TODO 1个任务）

```
-- 任务一  将20号部门的员工信息插入新的员工表emp1中 （emp1与emp结构一样，没有数据）
INSERT INTO emp1 SELECT * FROM emp WHERE deptno=20;
COMMIT;  -- 若开启事务，手动提交
```

8. FROM子句中使用子查询 （TODO 2个任务）

```
-- 任务一  查询高于部门平均工资的员工信息
SELECT e.*
FROM emp e,(SELECT deptno,AVG(sal) avgsal FROM emp GROUP BY deptno) d
WHERE e.deptno=d.deptno AND e.sal>d.avgsal;

-- 任务二  检索部门编号、部门名称、部门所在地及其每个部门的员工总数。
SELECT d.deptno,d.dname,d.loc,e.count FROM dept d,(SELECT deptno,COUNT(*) count FROM emp
GROUP BY deptno) e
WHERE d.deptno=e.deptno;
```

9. 合并查询结果（UNION,UNION ALL) (TODO 2个任务)

```
-- 任务一：查询工资大于2500或者职位为经理的员工的姓名，工资，职位。
SELECT ename,sal,job FROM emp WHERE sal>2500
UNION
SELECT ename,sal,job FROM emp WHERE job='MANAGER';
-- 任务二：查询工资大于2500或者职位为经理的员工的姓名，工资，职位。
SELECT ename,sal,job FROM emp WHERE sal>2500
UNION ALL
SELECT ename,sal,job FROM emp WHERE job='MANAGER';
```

> 注意： union（合并查询结果去除重复） union all（单纯的合并结果不处理重复数据）

(二) 索引 （TODO 2个任务）

1. 如何创建索引?

> 语法格式：CREATE INDEX 索引名称 ON 表名（字段1，字段2）；

```
-- 任务  给员工表的字段（部门号）创建索引。
CREATE INDEX index_emp ON emp(deptno);
```

2. 如何删除索引?

> 语法格式： DROP INDEX 索引名称 ON 表名;

```
--任务 删除索引index_emp。
DROP INDEX index_emp ON emp;
```

三 课后任务 (TODO 9个任务)

- [x] 任务1 查询与SMITH所在部门和职务相同的员工信息
- [x] 任务2 显示BLAKE同部门的所有员工姓名,职位和部门号，但不显示BLAKE
- [x] 任务3 查询与10号部门员工职位相同的员工信息。
- [x] 任务4 显示超过平均工资的所有员工名、工资和部门号
- [x] 任务5 显示超过所有部门平均工资的所有员工名、工资和部门号
- [x] 任务6 显示高于CLERK岗位所有雇员工资的所有雇员名、工资和岗位
- [x] 任务7 显示工资、奖金与SCOTT完全一致的所有雇员名、工资和奖金
- [x] 任务8 查询比10号部门员工工资高的员工信息（姓名，薪水，部门号）
- [x] 任务9 查询比30号部门所有员工工资高的员工信息（姓名，薪水，部门号）

四 拓展（了解MySQL数据库中的触发器，存储函数和存储过程）

4.1 什么是触发器?

> 触发器是对表进行插入、更新、删除的时候会自动执行的特殊存储过程。简单的说，就是一张表发生了某件事（插入、删除、更新操作），然后自动触发了预先编写好的若干条SQL语句的执行。 触发器一般用在check约束更加复杂的约束上面。例如在执行update、insert、delete这些操作的时候，系统会自动调用执行该表上对应的触发器。主要是为了保证数据的完整性，起到约束的作用。

> MySQL中触发器可以分为两类：DML触发器和DDL触发器，其中DDL触发器它们会影响多种数据定义语言语句而激发，这些语句有create、alter、drop语句。

> DML触发器分为： 1、 after触发器（之后触发） a、 insert触发器 b、 update触发器 c、 delete触发器 2、 before触发器 （之前触发）

> 其中after触发器要求只有执行某一操作insert、update、delete之后触发器才被触发，且只能定义在表上。

4.2 如何创建一个触发器?

> 语法格式：
>
> DELIMITER $$
>
> CREATE TRIGGER `数据库名称`.`触发器名称` AFTER|BEFORE INSERT|UPDATE|DELETE ON `数据库名称`.`对哪一张表进行监控` FOR EACH ROW
>
> BEGIN 这里写对应的SQL END$$
>
> DELIMITER ;

例如：创建一个触发器,当删除数据表t_student数据时，触发器被触发向数据表t_time里插入当前时间(一旦有满足条件的删除操作，就会执行BEGIN和END中的语句)

```
DELIMITER $$
CREATE TRIGGER trig_stu BEFORE DELETE
  ON t_student
  FOR EACH ROW
  BEGIN
    INSERT INTO t_time(tdate) VALUES(NOW());
  END$$
DELIMITER ;
```

> **注意：慎用触发器，尽量少使用触发器，不建议使用。触发器是针对每一行的；对增删改非常频繁的表上切记不要使用触发器，因为它会非常消耗资源。**

4.3 什么是存储过程?

存储过程是一组为了完成特定功能的 SQL 语句集合。使用存储过程的目的是将常用或复杂的工作预先用 SQL 语句写好并用一个指定名称存储起来，这个过程经编译和优化后存储在数据库服务器中，因此称为存储过程。

一个存储过程是一个可编程的函数，它在数据库中创建并保存，一般由 SQL 语句和一些特殊的控制结构组成。当希望在不同的应用程序或平台上执行相同的特定功能时，存储过程尤为合适。

存储过程是数据库中的一个重要功能，存储过程可以用来转换数据、数据迁移、制作报表，它类似于编程语言，一次执行成功，就可以随时被调用，完成指定的功能操作。
使用存储过程不仅可以提高数据库的访问效率，同时也可以提高数据库使用的安全性。

4.4 什么是存储函数（自定义函数）?

存储函数和存储过程一样，也是sql语句组成的代码块。

存储函数可以有输入参数，并且可以直接调用，不需要call语句，且必须有一条包含RETURN语句。

4.5 存储过程与存储函数（自定义函数）的区别?

```
    1.存储函数和存储过程统称为存储例程(store routine),存储函数的限制比较多,例如不能用临时表,只能用表变量,而存储过程的限制较少,存储过程的实现功能要复杂些,而函数的实现功能针对性比较强。
    2.返回值不同:
        存储函数必须有返回值,且仅返回一个结果值;
        存储过程可以没有返回值,但是能返回结果集(out,inout).
    3.调用时的不同:
        存储函数嵌入在SQL中使用,可以使用select 存储函数名(变量值) 来调用;
        存储过程通过call语句调用 call 存储过程名;
    4.参数的不同:
        存储函数的参数类型类似于IN参数
        存储过程的参数类型有三种:
        (1) in 数据只是从外部传入内部使用(值传递),可以是数值也可以是变量
        (2) out 只允许过程内部使用(不用外部数据),给外部使用的(引用传递:外部的数据会被先清空才会进入到内部),只能是变量
        (3) inout 外部可以在内部使用,内部修改的也可以给外部使用,典型的引用传递,只能传递变量
```

4.4 如何创建及调用一个存储过程?

> （1）创建存储过程语法： CREATE PROCEDURE 存储过程名称(参数列表) BEGIN SQL语句 END ;
>
> **备注**：可以不带参数列表。也可以带参数列表，可以使用IN，OUT,INOUT参数。
>
> 案例1：创建一个存储过程查看所有员工信息。

```
DELIMITER $$
CREATE PROCEDURE proc_queryEmp()
READS SQL DATA
BEGIN
  SELECT * FROM emp;
END
$$
```

> **备注**：MySQL中需要通过delimiter声明一个$$是用户定义的结束符，表示遇到$$才表示程序运行结束，否则遇到分号程序就结束了。通常这个结束符号可以是一些特殊的符号。另外应避免使用反斜杠,因为它是转义字符。
>
> （2）调用存储过程语法： call 存储过程名(); 案例1：call proc_queryEmp();

> 案例2：创建一个带参数的存储过程，根据员工编号查询员工的年薪.

```
DELIMITER $$
CREATE PROCEDURE proc_getEmpYearSal(IN eno int,OUT yearSal double(7,2))
READS SQL DATA
BEGIN
  SELECT sal*12  INTO yearSal FROM emp WHERE empno=eno;
END
$$
```

> 调用存储过程：

```
SET @empSal=0;
call proc_getEmpYearSal(7369,@empSal);
SELECT @empSal;
```

> 备注：@eno是定义的用户变量，SET @eno=0 表示设置初始值。

4.5 如何创建及调用一个存储函数?

> （1）创建存储函数（自定义函数）的语法： CRETAE FUNCTION 函数名() RETURNS 返回类型 BEGIN [DECLARE 变量 变量类型 ;] -- 如果需要定义局部变量就需要先声明。 sql 语句集合; RETURN 变量值; END;
>
> **备注**：在函数中必须要有RETURN返回值；而且MySQL中的存储函数的参数只有IN类型，默认就是IN类型。

> 案例：创建一个带参数的存储函数，根据员工编号查询员工的年薪.

```
DELIMITER $$
CREATE FUNCTION func_getEmpYearSal(eno int)
RETURNS double
READS SQL DATA
BEGIN
  DECLARE yearSal double(7,2);
  SELECT sal*12  INTO yearSal FROM emp WHERE empno=eno;
  RETURN yearSal;
END
$$
```

> （2）调用存储函数的语法： SELECT 函数名（参数）; 案例：

```
SELECT func_getEmpYearSal(7369);
```

4.6 如何使用游标?

> MySQL中的游标可以理解成一个可迭代对象(类比Python中的列表、字典等可迭代对象)，它可以用来存储SELECT语句查询到的结果集，这个结果集可以包含多行数据，从而使我们可以使用迭代的方法从游标中依次取出每行数据。
>
> 使用游标的四个步骤：
>
> （1） 声明游标
>
> 游标声明必须在变量声明之后。如果在变量声明之前声明游标，MySQL将会发出一个错误。游标必须始终与SELECT语句相关联。
>
> 语法： DECLARE cursor_name CURSOR FOR select_statement;
>
> （2）打开游标
>
> 使用OPEN语句打开游标，只有先打开游标才能读取数据。
>
> 语法： OPEN cursor_name;
>
> （3）读取游标
>
> 使用FETCH语句来检索游标指向的一行数据，并将游标移动到结果集中的下一行。

> 语法：FETCH cursor_name INTO var_name;
>
> （4）关闭游标
>
> 使用CLOSE语句关闭游标。
>
> 语法：CLOSE cursor_name;

4.7 如何删除存储过程?

> 语法：DROP PROCEDURE [IF EXISTS] 存储过程名称; 注意：它是不带括号的。
>
> 案例：

```
DROP PROCEDURE proc_queryEmp;
```

4.8 如何删除存储函数?

> 语法：DROP FUNCTION [IF EXISTS] 存储函数名称;
>
> 案例：

```
DROP FUNCTION func_getEmpYearSal;
```