MySQL 学习笔记 知识点三

作者：PeterChu
原文链接：https://ld246.com/article/1598961410697
来源网站：链滴

Oracle 与 MySQL 中的一些语法区别以及知识点

1. 聚合函数 NVL

```
SELECT ename, sal + nvl(comm, 0) * 0.8 money FROM emp ORDER BY money; -- nvl(comm, 0) oracle中的聚合函数，如果第一个参数为null，则会返回第二个参数
SELECT ename, sal+IFNULL(comm, 0)*0.8 money FROM emp ORDER BY money;
```

2. 日期类型的区别

```
INSERT INTO myemp(id, name, job, birth) VALUES(1003, 'donna', 'MANAGER', TO_DATE('1978 09-01', 'YYYY-MM-DD')); -- Oracle 中可用
```

在 MySQL 中指定日期格式需要使用：DATE_FORMAT(d,f)

```
SELECT DATE_FORMAT('2011-11-11 11:11:11','%Y-%m-%d %r')
-> 2011-11-11 11:11:11 AM
```

3. 重命名表格

```
RENAME employee TO myemp; -- Oracle 中语法，MySQL中错误
RENAME TABLE employee TO myemp;     --  MySQL
```

4. MySQL 中日期和时间类型

类型	大小(bytes)	范围	格式

途

DATE	3	1000-01-01/9999-12-31	YYYY MM-DD	日期值
TIME	3	'-838:59:59'/'838:59:59'	HH:M :SS	时间值或持续时间
YEAR	1	1901/2155	YYYY	份值
DATETIME	8	1000-01-01 00:00:00/9999-12-31 23:59:59	YYY-MM-DD HH:MM:SS	混合日期和时间值
TIMESTAMP	4	1970-01-01 00:00:00/2038 结束时间是第 2 47483647 秒，北京时间 2038-1-19 11:14:07，格林尼治时间 2038年1月19日 凌晨 03:14:07	YYYMMDD HHMMSS	混合日期和时间值，时间戳

5. DATE_FORMAT(date, format) 根据 format 字符串安排 date 的格式。

说明符	说明
%a	工作日的缩写名称（Sun..Sat)
%b	月份的缩写名称（Jan..Dec)
%c	月份，数字形式(0..12)
%D	带有英语后缀的该月日期（0th, 1st, 2nd, 3rd, ...)
%d	该月日期, 数字形式 (00..31)
%e	该月日期, 数字形式(0..31)
%f	微秒 (000000..999999)
%H	小时(00..23)
%h	小时(01..12)
%I	小时 (01..12)
%i	分钟,数字形式 (00..59)
%j	一年中的天数 (001..366)
%k	小时 (0..23)
%l	小时 (1..12)
%M	月份名称 (January..December)
%m	月份, 数字形式 (00..12)
%p	上午（AM）或下午（ PM）
%r	时间 , 12小时制 (小时hh:分钟mm:秒数ss 后加 AM或PM)
%S	秒 (00..59)
%s	秒 (00..59)
%T	时间 , 24小时制 (小时hh:分钟mm:秒数ss)

%U 周 (00..53), 其中周日为每周的第一天

%u 周 (00..53), 其中周一为每周的第一天

%V 周 (01..53), 其中周日为每周的第一天 ; 和 %X 同时使用

%v 周 (01..53), 其中周一为每周的第一天 ; 和 %x 时使用

%W 工作日名称 (周日..周六)

%w 一周中的每日 (0=周日..6=周六)

%X 该周的年份，其中周日为每周的第一天, 数字式,4位数;和%V同时使用

%x 该周的年份，其中周一为每周的第一天, 数字式,4位数;和%v同时使用

%Y 年份, 数字形式,4位数

%y 年份, 数字形式 (2位数)

%% ‘%’ 文字字符

6. 查询语句的执行顺序

当一条查询语句中包含所有的子句，执行顺序依下列子句次序：

a. FROM 子句：执行顺序为从后往前、从右到左。数据量较少的表尽量放在后面。

b. WHERE子句：执行顺序为自下而上、从右到左。将能过滤掉最大数量记录的条件写在WHERE 子的最右。

c. GROUP BY：执行顺序从左往右分组，最好在GROUP BY前使用WHERE将不需要的记录在GROUP Y之前过滤掉。

d. HAVING 子句：消耗资源。尽量避免使用，HAVING 会在检索出所有记录之后才对结果集进行过，需要排序等操作。

e. SELECT子句：少用号，尽量取字段名称。ORACLE 在解析的过程中, 通过查询数据字典将号依次转成所有的列名, 消耗时间。

f. ORDER BY子句：执行顺序为从左到右排序，消耗资源。

7. 关联查询中驱动表

外连接查询的例子,Emp表做驱动表:

```
SELECT e.ename, d.dname
FROM emp e LEFT OUTER JOIN dept d
ON e.deptno = d.deptno;
```

Dept表做驱动表:

```
SELECT e.ename, d.dname
FROM emp e RIGHT OUTER JOIN dept d
ON e.deptno = d.deptno;
```

左连接中 LEFT OUTER JOIN 左边的表为驱动表。右连接中 RIGHT OUTER JOIN 右边的表为驱动表。

哪张表做驱动表，则查询的结果则为用驱动表中的所有记录去关联查询非驱动表，获取到的记录中必包含了所有的驱动表中的记录。

8. 全连接：FULL JOIN

全外连接是指除了返回两个表中满足连接条件的记录，还会返回不满足连接条件的所有其它行。即是外连接和右外连接查询结果的总和。例如：

```
SELECT e.ename, d.dname
FROM emp e FULL OUTER JOIN dept d
ON e.deptno = d.deptno;
```

9. 自连接

自连接是一种特殊的连接查询，数据的来源是一个表，即关联关系来自于单表中的多个列。表中的列照同一个表中的其它列的情况称作自参照表。

自连接是通过将表用别名虚拟成两个表的方式实现，可以是等值或不等值连接。例如查出每个职员的理名字，以及他们的职员编码：

```
SELECT worker.empnow_empno, worker.enamew_ename, manager.empnom_empno, manager
enamem_ename
FROM emp worker join emp manager
ON worker.mgr = manager.empno;

SELECT e1.ename, e2.ename FROM emp e1, emp e2 WHERE e1.mgr = e2.empno;  -- 查询每
员工的上级管理者
SELECT e1.ename, e2.ename FROM emp e1, emp e2 WHERE e1.empno = e2.mgr;  -- 查询每
管理者的下级员工
```

10. SQL中where子句中不能出现聚合函数的原因

首先我们应该熟悉什么聚合函数：

例如SUM(),MIN(),Max()这类的，我们称作是聚合函数。

那么我们不能在where子句中使用这些函数，为什么呢?

聚集函数也叫列函数，它们都是基于整列数据进行计算的，而where子句则是对数据行进行过滤的，筛选过程中依赖“基于已经筛选完毕的数据得出的计算结果”是一种悖论，这是行不通的。更简单地，因为聚集函数要对全列数据时行计算，因而使用它的前提是：结果集已经确定!

而where子句还处于“确定”结果集的过程中，因而不能使用聚集函数。

与where子句不能出现聚集函数正相反的是，我们几乎看不到不使用聚集函数的having子句。为什么因为在水平方向上根据外部指定条件的筛选（也就是对行的筛选），where子句可以独立完成，剩下往往都是需要根据结果集自身的统计数据进一步筛选了，这时，几乎都需要通过having子句配合聚集数来完成。

按照下面这个就是错误的，会报一个错误：Group function is not allowed here

```
select department_id,avg(salary)
from employees
where  avg(salary)>6000
group by department_id
--having  avg(salary)>6000
```

原因如下

sql语句的执行顺序为：

from子句

where 子句

group by 子句

having 子句

order by 子句

select 子句

首先得知道聚合函数是对结果集运算的，当在where子句使用聚合函数时，此时根据group by 分割果集的子句还没有执行，此时只有from 后的结果集。

所以无法在where子句中使用聚合函数。

11. LIKE 查询

expr LIKE pat [ESCAPE 'escape-char']

模式匹配，使用SQL简单正规表达式比较。返回1 (TRUE) 或 0 (FALSE)。 若 expr 或 pat 中任何一个 NULL,则结果为 NULL。

模式不需要为文字字符串。例如，可以被指定为一个字符串表达式或表列。

可以同LIKE一起使用以下两种通配符：

% 匹配任何数目的字符，甚至包括零字符。

_只能匹配一种字符

若要对通配符的文字实例进行检验, 可将转义字符放在该字符前面。如果没有指定 ESCAPE字符, 则假为 '\' 。

\% 匹配一个 '%' 字符。

_ 匹配一个 '_' 字符。

在MySQL中, LIKE 允许出现在数字表达式中。 (这是标准SQL LIKE 的延伸）。

```
SELECT 10 LIKE '1%'; -- ->1 true
```

注意：由于 MySQL在字符串中使用 C转义语法(例如, 用 '\n' 代表一个换行字符)，在LIKE字符串中必须将用到的 '\' 双写。例如， 若要查找 '\n' , 必须将其写成 '\\' 。而若要查找 '\' , 则必须将其写成 it as '\\\\' ;原因是反斜线符号会被语法分析程序剥离一次，在进行模式匹配时，又会被剥离一次，最后会剩下一反斜线符号接受匹配。

```
SELECT ename FROM emp WHERE ename LIKE '__A%';  -- 查询名字中第三个字母是A 的员工
```

12. 拼接字段

语法：

COUNT(DISTINCT expr ,[expr ...])

函数使用说明：返回不同的非NULL 值数目。若找不到匹配的项，则COUNT(DISTINCT) 返回 0

```
SELECT ename ||','|| job ||','|| sal OUT_PUT FROM emp ; -- ORACLE 中可用
SELECT CONCAT(ename, ',', job, ',', sal) AS OUT_PUT FROM emp;
```

13. BETWEEN 运算符的用法

BETWEEN...AND...操作符用来查询符合某个值域范围条件的数据，最常见的是使用在数字类型的数范围上，但对字符类型和日期类型数据也同样适用。

比较值可以是固定的，也可以是表中的某列中的数据。

```
-- 查询编号为1的员工的税别
SELECT sal FROM empe WHERE empno = 1;
SELECT grade FROM taxgrade WHERE taxmin < (SELECT sal FROM empe WHERE empno = 1)
< taxmax; -- 错误，查询出所有的 grade
SELECT grade FROM taxgrade WHERE taxmin < (SELECT sal FROM empe WHERE empno = 1)
&& (SELECT sal FROM empe WHERE empno = 1) < taxmax; -- ok
SELECT e.empname, e.empno, e.sal, t.grade FROM empe e JOIN taxgrade t ON e.sal BETWEEN
t.taxmin AND t.taxmax;
```