Mysql数据操作

添加记录

- ```
  insert [into] tbl_name[{col_name,...}]{value|values}(values...);
  ```
- 不指定字段名称
  - ```
    insert tbl_name value(value....);
    ```
  - 需要按照建表时的字段顺序给每一个字段赋值
- 列出指定字段
  - ```
    insert tbl_name(字段名称,...)values(值,....);
    ```
- `insert ... set`的形式
  - ```
    insert tbl_name set 字段名称=值,....;
    ```
- `insert ... select`
  - ```
    insert tbl_name[(字段名称)] select 字段名称,... from tbl_name [where 条件];
    ```
- 一次添加多条记录
  - ```
    insert tbl_name[(字段名称,....)]values(值,....),
    (值,....),
    (值,....);
    ```

修改记录

- ```
  update tbl_name set 字段名称=值，字段名称=值[where 条件];
  ```
- 如果不添加条件，整个表中的记录都会被更新

删除记录

- ```
  delete from tbl_name[where 条件];
  ```
- 如果不添加条件，表中所有记录都会被删除
- delete清空数据表的时候不会重置AUTO_INCREMENT的值，可以通过ALTER语句将其重置为1

- 彻底清空数据表
  - `truncate [table] tbl_name`

查询记录

- ```
  select select_expr,... from tbl_name
  [where 条件]
  [group by{col_name|position} having 二次筛选]
  [order by{col_name|position} [asc|desc]]
  [limit 限制结果集的显示条数]
  ```
- 查询表中所有记录
  - ```
    select * from tbl_name;
    ```
  - `*`代表所有字段
- 指定字段的信息
  - ```
    select 字段名称,... from tbl_name;
    ```
- 库名.表名
  - ```
    select 字段名称,... from db_name.tbl_name;
    ```
- 给字段起别名
  - ```
    select 字段名称 [as] 别名名称,... from db_name.tbl_name;
    ```
- 给数据表起别名
  - ```
    select 字段名称,.... from tbl_name [as] 别名;
    ```
- 表名.字段名称
  - ```
    select tbl_name.col_name,... from tbl_name;
    ```
- where条件
  - 会筛选出符合条件的记录
    - 比较运算符 `>,<,>=,<=,!=,<>,<=>`
    - `<=>`和`=`的区别
      - `<=>`可以检测null值
    - `is [not] null`
    ```

        - 检测值是否为null或者not null
    - 指定范围
        - ```
          [not] between ... and
          ```
    - 指定集合
        - ```
          [not] in(值,...)
          ```
    - 逻辑运算符
        - `and` 逻辑与
        - `or` 逻辑或
    - 匹配字符
        - ```
          [not] like
          ```
        - `%`:任意长度的字符串
        - `_`：任意一个字符
- GROIP BY分组
    - 把值相同的放到一个组中，最终查询出的结果只会显示组中的一条记录

多表查询

- 笛卡尔积的形式
- 内连接的形式
    - 查询两个表中符合连接条件的记录
    - ```
      select 字段名称,.... from tbl_name1
      inner join tbl_name2
      on 连接条件
      ```
- 外连接的形式
    - 左外连接
        - 先显示左表中的全部记录，再去右表中查询符合条件的记录，不符合的以null代替
        - ```
          select 字段名称,... from tbl_name1
          left [outer] join tbl_name2
          on 条件;
          ```
    - 右外连接
        - 先显示右表中的全部记录，再去左表中查询符合条件的记录，不符合的以null代替
          ```

```
select 字段名称,... from tbl_name1
right [outer] join tbl_name2
on 条件;
```

外键约束

只有InnoDB存储引擎支持外键

- 创建外键
  - 建表时指定外键

```
[constraint 外键名称]foreign key(字段名称) references 主表(字段名称)
```

    - 子表的外键字段和主表的主键字段类型要相似，如果是数值型要求一致，并且无符号也要一致；如果是字符型，要求类型一致，长度可以不同。
    - 如果外键字段没有创建索引，Mysql会自动帮我们添加索引
    - 子表的外键关联必须是父表的主键
    - 外键约束的参照操作
      - `cascade` 从父表删除或更新，子表也跟着删除或更新，级联的操作
      - `set null` 从父表删除或者更新记录，并设置子表的外键列为null
      - `no action|restrict` 拒绝对父表做更新或者删除操作
  - 动态添加外键
    - 动态添加外键

```
alter table tbl_name
add [constraint 外键名称] foreign key(外键字段) references 主表(主键字段);
```

      - 动态添加外键之前表中的记录一定要是合法记录，否则动态添加外键是不成功的
    - 动态删除外键

```
alter table tbl_name
drop foreign key 外键名称
```

特殊形式的查询

- 子查询

```
select 字段名称 from tbl_name where col_name=(select col_name from tbl_name)
```

- 内层语句查询的结果可以作为外层语句查询的条件
- 由In引发的子查询

```
/*测试由in引发的子查询*/
select * from emp
where depId  in (select id from dep);
```

- 由比较运算符引出子查询

```
select id,username,score from stu
where score>=(select score from level where id = 1 );
```

- 由exists引发的子查询

```
select * from emp where exists (select depName from dep where id =10) ;
```

- any some all

```
select * from stu
where score>= any(select score from level);

select * from stu
where score>= some(select score from level);

select * from stu
where score>= all(select score from level);
```

- 自身连接查询
  - 无限级分类的实现形式

MySQL常用函数

- 数学函数

数学函数

CEIL()	进一取整	ABS()	取绝对值
FLOOR()	舍掉小数部分	POWER()	幂运算
ROUND()	四舍五入	PI()	圆周率
TRUNCATE()	截取小数点后几位	RAND()或者RAND(X)	0~1之间的随机数
MOD()	取余数	SIGN(X)	得到数字符号
		EXP()	计算e的x次方

- 字符串函数

```
/*char_length():等到字符串的字符数*/
select char_length('abc');

/*length():等到字符串的长度*/
select length('你好啊');

/*concat(s1,s2,....):将多个字符串合并成一个字符串*/
select concat('jf','你好啊');
/*如果字符串中包含null，返回拼接结果就是null*/
select concat('jf','你好啊',null);

/*concat_ws(x,s1,s2,...):以x为拼接符来拼接字符串*/
select concat_ws('-','jf','你好啊');

/*将字符串转换成大写或者小写 upper()|ucase()|lower()|lcase()*/
select upper('hello king'),ucase('hello IMOOC'),lower('HELLO imooc'),lcase('HELLO
IMOOC');

/*字符串的反转reverse()*/
select reverse('abc');

/*left()|right():返回字符串的前几个字符或者后几个字符*/
select left('hello',2),right('hello',2);

/*lpad()|rpad():用字符串填充到指定长度*/
select lpad('abc',10,'?');
select rpad('abc',10,'!');

/*去掉字符串两端的空格trim()|ltrim()|rtrim()*/
select concat('*',trim('  asd  '),'*'),concat('*',ltrim('   sd
'),'*'),concat('*',rtrim('   dsd    '),'*');

/*repeat():重复指定的次数*/
select repeat('hello',3);

```

```
/*字符串替换: replace()*/
select replace('hello king','king','queen');

/*截取字符串 substring*/
select substring('abcdef',1,3);

/*比较字符串 strcmp 按照字符串的ASCII来比较*/
select strcmp('a','b');

```

- 日期时间函数

```
/*返回当前日期*/
select curdate(),current_date();

/*返回当前时间*/
select curtime(),current_time();

/*返回当前日期时间*/
select now(),current_timestamp(),sysdate();

/*返回日期中的月份*/
select month('2018-10-3');
select month(curdate()),monthname(curdate());

/*返回星期几*/
select dayname(now());

/*返回一周内的第几天*/
select dayofweek(now());

/*返回一年内第几个星期*/
select week(now());

select year(now()),month(now()),day(now()),hour(now()),minute(now()),second(now());

/*计算两个日期相差的天数: datediff()*/
select datediff('2018-03-01',now());
```

- 其他常用函数

```
/*得到数据库版本和得到服务器连接数*/
select version(),connection_id();

/*得到当前用户*/
select user(),current_user(),system_user(),session_user();

/*得到上一步插入操作产生的auto_increment的值*/
select last_insert_id();
```