**sql 中 limit 与 limit，offset连用的区别**

2017年05月29日 17:52:49 [AinUser](https://me.csdn.net/AinUser) 阅读数：3848

① select \* from table limit 2,1;                 

//含义是跳过2条取出1条数据，limit后面是从第2条开始读，读取1条信息，即读取第3条数据

② select \* from table limit 2 offset 1;      

//含义是从第1条（不包括）数据开始取出2条数据，limit后面跟的是2条数据，offset后面是从第1条开始读取，即读取第2,3条

使用查询语句的时候，经常要返回前几条或者中间某几行数据，这个时候怎么办呢？不用担心，已 经为我们提供了这样一个功能。  
 LIMIT 子句可以被用于强制 SELECT 语句返回指定的记录数。LIMIT 接受一个或两个数字参数。参数必须是一个整数常量。  
如果给定两个参数，第一个参数指定第一个返回记录行的偏移量，第二个参数指定返回记录行的最大数目。  
SELECT \* FROM table   LIMIT [offset,] rows | rows OFFSET offset

这是两个参数，第一个是偏移量，第二个是数目  
select \* from employee limit 3, 7; // 返回4-11行  
select \* from employee limit 3,1; // 返回第4行  
  
一个参数  
select \* from employee limit 3; // 返回前3行

以前我在[mysql](http://www.ityoudao.com/Web/Sjk/Mysql/)中分页都是用的 limit [10000](https://www.baidu.com/s?wd=10000&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)0,20这样的方式，我相信你也是吧，但是要提高效率，让分页的代码效率更高一些，更快一些,那我们又该怎么做呢?

第一部分：看一下分页的基本原理：

mysql explain SELECT \* FROM message ORDER BY id DESC LIMIT 10000, 20  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*  
id: 1  
select\_type: SIMPLE  
table: message  
type: index  
possible\_keys: NULL  
key: PRIMARY  
key\_len: 4  
ref: NULL  
rows: 10020  
Extra:  
1 row in set (0.00 sec)

　　对上面的mysql语句说明：limit 10000,20的意思扫描满足条件的10020行，扔掉前面的10000行，返回最后的20行，问题就在这里，如果是limit 100000,100，需要扫描100100行，在一个高并发的应用里，每次查询需要扫描超过10W行，性能肯定大打折扣。文中还提到limit n性能是没问题的，因为只扫描n行。

第二部分：根据雅虎的几位工程师带来了一篇Efficient Pagination Using MySQL的报告内容扩展：在文中提到一种clue的做法，给翻页提供一些线索，比如还是SELECT \* FROM message ORDER BY id DESC，按id降序分页，每页20条，当前是第10页，当前页条目id最大的是1020，最小的是1000，如果我们只提供上一页、下一页这样的跳转（不提供到第N页的跳转），那么在处理上一页的时候SQL语句可以是：

SELECT \* FROM message WHERE id>1020 ORDER BY id ASC LIMIT 20;//下一页

　　处理下一页的时候SQL语句可以是：

SELECT \* FROM message WHERE id<1000 ORDER BY id DESC LIMIT 20;//上一页

　　不管翻多少页，每次查询只扫描20行。

　　缺点是只能提供上一页、下一页的链接形式，但是我们的产品经理非常喜欢“上一页 1 2 3 4 5 6 7 8 9 下一页”这样的链接方式，怎么办呢？

　　如果LIMIT m,n[不可避免](https://www.baidu.com/s?wd=%E4%B8%8D%E5%8F%AF%E9%81%BF%E5%85%8D&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)的话，要优化效率，只有尽可能的让m小一下，我们扩展前面的clue做法，还是SELECT \* FROM message ORDER BY id DESC，按id降序分页，每页20条，当前是第10页，当前页条目id最大的是2519，最小的是2500;  
当是第10页的SQL如下：

SELECT \* FROM tb\_goods\_info WHERE auto\_id >=2500 ORDER BY auto\_id ASC LIMIT 0,20

比如要跳到第9页，SQL语句可以这样写：

SELECT \* FROM tb\_goods\_info WHERE auto\_id <2500 ORDER BY auto\_id desc LIMIT 0,20

比如要跳到第8页，SQL语句可以这样写：

SELECT \* FROM tb\_goods\_info WHERE auto\_id <2500 ORDER BY auto\_id desc LIMIT 20,20

比如要跳到第7页，SQL语句可以这样写：

SELECT \* FROM tb\_goods\_info WHERE auto\_id <2500 ORDER BY auto\_id desc LIMIT 40,20

跳转到第11页：

SELECT \* FROM tb\_goods\_info WHERE auto\_id >2519 ORDER BY auto\_id asc LIMIT 0,20

跳转到第12页：

SELECT \* FROM tb\_goods\_info WHERE auto\_id >2519 ORDER BY auto\_id asc LIMIT 20,20

跳转到第13页：

SELECT \* FROM tb\_goods\_info WHERE auto\_id >2519 ORDER BY auto\_id asc LIMIT 40,20

　　原理还是一样，记录住当前页id的最大值和最小值，计算跳转页面和当前页相对偏移，由于页面相近，这个偏移量不会很大，这样的话m值相对较小，大大减少扫描的行数。其实传统的limit m,n，相对的偏移一直是第一页，这样的话越翻到后面，效率越差，而上面给出的方法就没有这样的问题。

　　注意SQL语句里面的ASC和DESC，如果是ASC取出来的结果，显示的时候记得倒置一下。

　　已在60W数据总量的表中测试，效果非常明显。文尾给出第11页的图片