Mysql\_Limit与分页

# Mysql分页优化

## Limit

### 小记

#### Limit的两个参数

SELECT \* FROM tb1 LIMIT startIndex, size;

参数1： startIndex //起始索引，起始行的脚标。

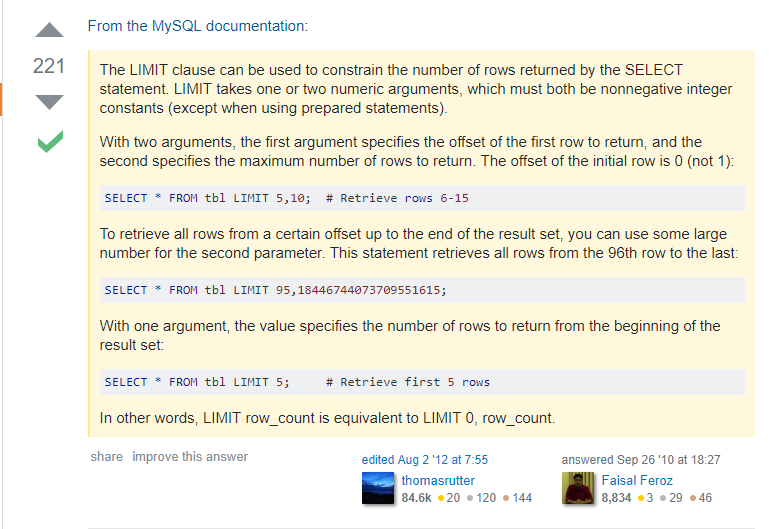
参数2：size //总共返回多少行

例如：

SELECT \* FROM tb1 LIMIT 0, 5

//

//就是返回：包括从脚标0开始，脚标为，0,1,2,3,4共5行数据。



要点：

With two arguments, the first argument specifies the offset of the first row to return, and the second specifies the maximum number of rows to return. The offset of the initial row is 0 (not 1):

即：

Mysql首行的脚标是0，而不是1.

#### 援引official文档：

The LIMIT clause can be used to constrain the number of rows returned by the [SELECT](https://dev.mysql.com/doc/refman/8.0/en/select.html) statement. LIMIT takes one or two numeric arguments, which must both be nonnegative integer constants, with these exceptions:

* Within prepared statements, LIMIT parameters can be specified using ? placeholder markers.
* Within stored programs, LIMIT parameters can be specified using integer-valued routine parameters or local variables.

With two arguments, the first argument specifies the offset of the first row to return, and the second specifies the maximum number of rows to return. The offset of the initial row is 0 (not 1):

SELECT \* FROM tbl LIMIT 5,10; # Retrieve rows 6-15

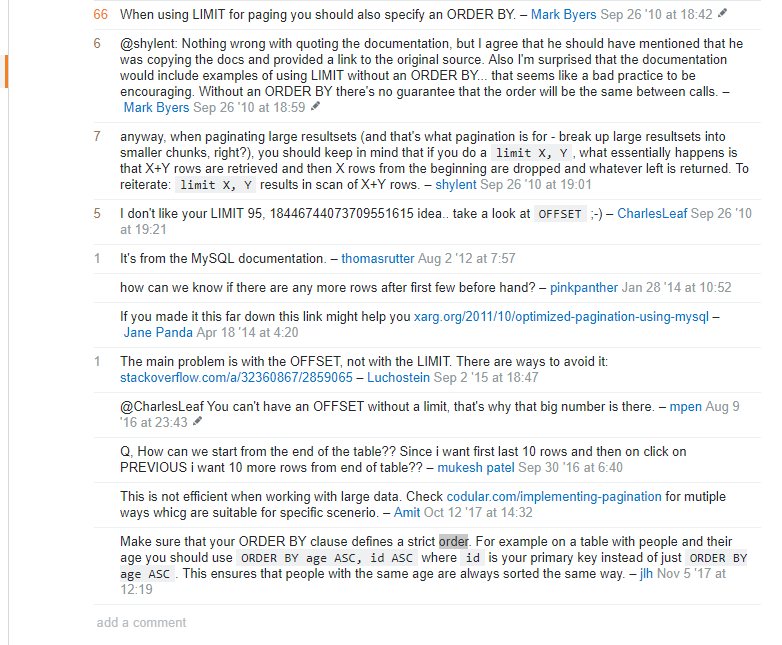
## 分页Paging

分页就是利用Limit来实现分页的功能。

即：如何把页号PageNum 和 起始索引startIndex关联在一起。

计算公式：

(页号-1)\*pageSize



## Mysql分页优化

### LIMIT和OFFSET的问题

“

在系统中需要进行分页操作的时候，我们通常会使用LIMIT加上偏移量的办法实现，同时

加上合适的ORDER BY子句。如果有对应的索引，通常效率会不错。否则，MySQL需要做

大量的文件排序操作。

一个非常常见又令人头疼的问题就是,在偏移量非常大的时候，例如可能是LIMIT 1000,20这样的查询，

这时Mysql需要查询10020条记录然后只返回最后20条，前面的10 000条都将被抛弃，这样的代价非常高。

如果所有的页面被访问的频率都相同，那么这样的查询平均需要访问半个表的数据。要优化这种查询，

要么是在分页中限制分页的数量，要么是优化大偏移量的性能。

优化此类分页查询的一个最简单的办法就是尽可能地使用索引覆盖扫描，而不是查询所有的列。

然后根据需要做一次关联操作再返回所需的列。对于偏移量很大的时候，这样做的效率会提升非常大。

考虑下面的查询：

mysql>

SELECT film\_id,description FROM film ORDER BY title LIMIT 50,5;

如果这个表非常大，那么这个查询最好改写成下面的样子：

#利用"延迟关联"策略进行优化

SELECT

film\_id,

description

FROM

film

INNER JOIN (

SELECT

film\_id

FROM

film

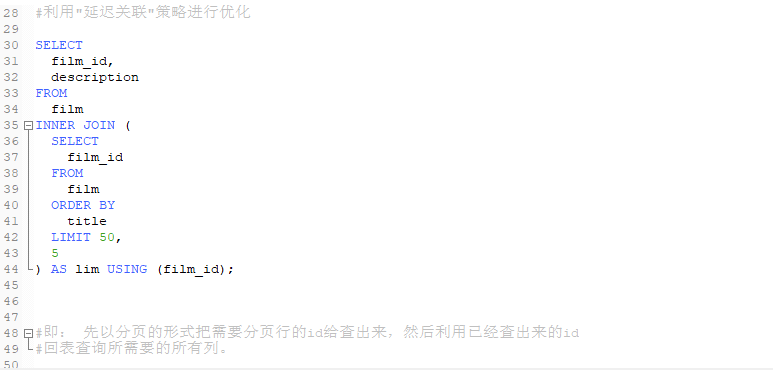
ORDER BY

title

LIMIT 50,

5

) AS lim USING (film\_id);



这里的“关联查询”将大大提升查询效率，它让MySQL 扫描尽可能少的页面，获取需要访问的记录后再跟进关联列回原表查询需要的所有列。

这个技术也可以用于优化关联查询中的LIMIT子句。

有时候也可以将LIMIT查询转换已知位置的查询，让MySQL通过范围扫描获得到对应的结果。例如，如果在一个位置上有索引，并且预先计算出了边界值，上面的查询就

可以改写为：

Mysql: SELECT film\_id,description FROM film WHERE position BETWEEN 50 AND 54 ORDER BY position

。

LIMIT和OFFSET 的问题，其实是OFFSET的问题，它会导致Mysql扫描大量不需要的行然后再抛弃掉。如果可以使用书签记录上次取数据的位置，

那么下次就可以直接从该书签记录的位置开始扫描，这样就可以避免使用OFFSET。例如，若需要按照租借记录做翻页，那么可以根据最新一条租借记录向后追溯，这种做法可行是因为租借记录的主键是单调增长的。

首先使用下面的查询获得第一组结果：

SELECT \* FROM rental ORDER BY rental\_id DESC LIMIT 20;

假设上面的查询返回的主键为16049到16030的租借记录，那么下一页查询就可以从16030这个点开始：

SELECT \* FROM rental WHERE rental\_id <16030

ORDER BY rental\_id DESC LIMIT 20;

该技术的好处是无论翻页到多么往后，其性能都会很好。

”

### 优化手段

#优化要达到的目的，就是让MySQL不再扫描那些不需要的行。即： 让MySQL扫描尽可能少的页面。

#### 延迟关联

利用延迟关联先把id主键列的分页给查出来，然后再回表查询所需列。

#优化LIMIT语句

#原始查询

SELECT film\_id,description FROM film ORDER BY title LIMIT 50,5;

#利用"延迟关联"策略进行优化

SELECT

film\_id,

description

FROM

film

INNER JOIN (

SELECT

film\_id

FROM

film

ORDER BY

title

LIMIT 50,

5

) AS lim USING (film\_id);

#即： 先以分页的形式把需要分页行的id给查出来，然后利用已经查出来的id

#回表查询所需要的所有列。

#### 书签记录位置

书签记录位置技术就是说：

记录下每次查询的所取的数据的位置，那么在下次查询时，直接利用上次查询的数据的位置进行筛选。

避免无谓的OFFSET

“

使用书签记录上次取数据的位置吗，那么下次就可以直接从该书签记录的位置开始扫描，这样

就可以避免使用OFFSET.

”