链滴

MYSQL 之查询篇

作者：vcjmhg

原文链接：https://ld246.com/article/1569720141271

来源网站：链滴

<h2 id="2-数据库操作">2 数据库操作</h2>
数据库在创建以后最常见的操作便是 <code>查询</code>
<h3 id="2-1-查询">2.1 查询</h3>
为了便于学习和理解，我们预先准备了两个表分别是 <code>stduents</code> 表和 <code>cl
sses</code> 表两个表的内容和结构如下所示
<code>students</code> 表的内容:
<table>
<thead>
<tr>
<th>id</th>
<th>class_id</th>
<th>name</th>
<th>gender</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>小明</td>
<td>M</td>
<td>90</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>小红</td>
<td>F</td>
<td>95</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>小军</td>
<td>M</td>
<td>88</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>小米</td>
<td>F</td>
<td>73</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>小白</td>
<td>F</td>
<td>81</td>
</tr>
<tr>
<td>6</td>

<td>2</td>
<td>小兵</td>
<td>M</td>
<td>55</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>小林</td>
<td>M</td>
<td>85</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>小新</td>
<td>F</td>
<td>91</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>小王</td>
<td>M</td>
<td>89</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>小丽</td>
<td>F</td>
<td>85</td>
</tr>
<tr>
<td>创建 <code>students</code> 表的 SQL 命令:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">/*创建表的sql语句*/<span class="highl
ght-w">
<span cla
s="highlight-w">CREATE<span class="highlight
w"> TABLE </span
students <span clas
="highlight-p">(
<span cla
s="highlight-w">   `i
` <span clas
="highlight-nb">int(<span class="highlight-mi"

```
11</span><span class="highlight-p">)</span><span class="highlight-w"> </span><span c
ass="highlight-k">NOT</span><span class="highlight-w"> </span><span class="highlight-
">NULL</span><span class="highlight-w"> </span><span class="highlight-n">AUTO_INCR
MENT</span><span class="highlight-p">,</span><span class="highlight-w">
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-w">  </span><span class="highlight-o">`</span><span class="highlight-n">c
ass_id</span><span class="highlight-o">`</span><span class="highlight-w"> </span><sp
n class="highlight-nb">int</span><span class="highlight-p">(</span><span class="highlig
t-mi">11</span><span class="highlight-p">)</span><span class="highlight-w"> </span>
span class="highlight-k">DEFAULT</span><span class="highlight-w"> </span><span class
"highlight-k">NULL</span><span class="highlight-p">,</span><span class="highlight-w">
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-w">  </span><span class="highlight-o">`</span><span class="highlight-n">n
me</span><span class="highlight-o">`</span><span class="highlight-w"> </span><span
lass="highlight-nb">varchar</span><span class="highlight-p">(</span><span class="highl
ght-mi">10</span><span class="highlight-p">)</span><span class="highlight-w"> </spa
><span class="highlight-k">DEFAULT</span><span class="highlight-w"> </span><span cl
ss="highlight-k">NULL</span><span class="highlight-p">,</span><span class="highlight-
">
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-w">  </span><span class="highlight-o">`</span><span class="highlight-n">g
nder</span><span class="highlight-o">`</span><span class="highlight-w"> </span><span
class="highlight-nb">char</span><span class="highlight-p">(</span><span class="highlig
t-mi">1</span><span class="highlight-p">)</span><span class="highlight-w"> </span><
pan class="highlight-k">DEFAULT</span><span class="highlight-w"> </span><span class=
highlight-k">NULL</span><span class="highlight-p">,</span><span class="highlight-w">
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-w">  </span><span class="highlight-o">`</span><span class="highlight-n">s
ore</span><span class="highlight-o">`</span><span class="highlight-w"> </span><span
lass="highlight-nb">int</span><span class="highlight-p">(</span><span class="highlight
mi">3</span><span class="highlight-p">)</span><span class="highlight-w"> </span><sp
n class="highlight-k">DEFAULT</span><span class="highlight-w"> </span><span class="h
ghlight-k">NULL</span><span class="highlight-p">,</span><span class="highlight-w">
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-w">  </span><span class="highlight-k">PRIMARY</span><span class="highli
ht-w"> </span><span class="highlight-k">KEY</span><span class="highlight-w"> </span
<span class="highlight-p">(</span><span class="highlight-o">`</span><span class="highl
ght-n">id</span><span class="highlight-o">`</span><span class="highlight-p">)</span>
span class="highlight-w">
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-w">  </span><span class="highlight-p">)</span><span class="highlight-w">
<
span><span class="highlight-n">ENGINE</span><span class="highlight-o">=</span><sp
n class="highlight-n">InnoDB</span><span class="highlight-w"> </span><span class="hig
light-k">DEFAULT</span><span class="highlight-w"> </span><span class="highlight-n">
HARSET</span><span class="highlight-o">=</span><span class="highlight-n">utf8</spa
><span class="highlight-p">;</span><span class="highlight-w">
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-w">  </span><span class="highlight-cm">/*插入测试数据*/</span><span class=
highlight-w">
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-w">  </span><span class="highlight-k">INSERT</span><span class="highlight
w"> </span><span class="highlight-k">INTO</span><span class="highlight-w"> </span><
pan class="highlight-o">`</span><span class="highlight-n">students</span><span class=
highlight-o">`</span><span class="highlight-w"> </span><span class="highlight-k">VALU
```

S (<span cl
ss="highlight-s1">'1',<span class="highlight-w
> '1',<span
class="highlight-w"> '小明'<span class="highli
ht-p">, 'M'</span
, <span class="highl
ght-s1">'90');
<span cla
s="highlight-w">INSERT<span class="highlight
w"> INTO <
pan class="highlight-o">`students<span class=
highlight-o">` VALU
S (<span cl
ss="highlight-s1">'2',<span class="highlight-w
> '1',<span
class="highlight-w"> '小红'<span class="highli
ht-p">, 'F'
span class="highlight-p">, <span class="highli
ht-s1">'95');
<span cla
s="highlight-w">INSERT<span class="highlight
w"> INTO <
pan class="highlight-o">`students<span class=
highlight-o">` VALU
S (<span cl
ss="highlight-s1">'3',<span class="highlight-w
> '1',<span
class="highlight-w"> '小军'<span class="highli
ht-p">, 'M'</span
, <span class="highl
ght-s1">'88');
<span cla
s="highlight-w">INSERT<span class="highlight
w"> INTO <
pan class="highlight-o">`students<span class=
highlight-o">` VALU
S (<span cl
ss="highlight-s1">'4',<span class="highlight-w
> '1',<span
class="highlight-w"> '小米'<span class="highli
ht-p">, 'F'
span class="highlight-p">, <span class="highli
ht-s1">'73');
<span cla
s="highlight-w">INSERT<span class="highlight
w"> INTO <
pan class="highlight-o">`students<span class=
highlight-o">` VALU
S (<span cl
ss="highlight-s1">'5',<span class="highlight-w
> '2',<span
class="highlight-w"> '小白'<span class="highli
ht-p">, 'F'
span class="highlight-p">, <span class="highli

ht-s1">'81');
<span cla
s="highlight-w"> INSERT<span class="highlight
w"> INTO <
pan class="highlight-o">`students<span class=
highlight-o">` VALU
S (<span cl
ss="highlight-s1">'6',<span class="highlight-w
> '2',<span
class="highlight-w"> '小兵'<span class="highli
ht-p">, 'M'</span
, <span class="highl
ght-s1">'55');
<span cla
s="highlight-w"> INSERT<span class="highlight
w"> INTO <
pan class="highlight-o">`students<span class=
highlight-o">` VALU
S (<span cl
ss="highlight-s1">'7',<span class="highlight-w
> '2',<span
class="highlight-w"> '小林'<span class="highli
ht-p">, 'M'</span
, <span class="highl
ght-s1">'85');
<span cla
s="highlight-w"> INSERT<span class="highlight
w"> INTO <
pan class="highlight-o">`students<span class=
highlight-o">` VALU
S (<span cl
ss="highlight-s1">'8',<span class="highlight-w
> '3',<span
class="highlight-w"> '小新'<span class="highli
ht-p">, 'F'
span class="highlight-p">, <span class="highli
ht-s1">'91');
<span cla
s="highlight-w"> INSERT<span class="highlight
w"> INTO <
pan class="highlight-o">`students<span class=
highlight-o">` VALU
S (<span cl
ss="highlight-s1">'9',<span class="highlight-w
> '3',<span
class="highlight-w"> '小王'<span class="highli
ht-p">, 'M'</span
, <span class="highl
ght-s1">'89');
<span cla
s="highlight-w"> INSERT<span class="highlight
w"> INTO <
pan class="highlight-o">`students<span class=
highlight-o">` VALU

S (<span cl
ss="highlight-s1">'10',<span class="highlight-w
> '3',<span
class="highlight-w"> '小丽'<span class="highli
ht-p">, 'F'
span class="highlight-p">, <span class="highli
ht-s1">'85');
</code></pre>
<code>classes</code> 表的内容和结构:
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>一班</td>
</tr>
<tr>
<td>2</td>
<td>二班</td>
</tr>
<tr>
<td>3</td>
<td>三班</td>
</tr>
<tr>
<td>4</td>
<td>四班</td>
</tr>
<tr>
<td>创建 <code>classes</code> 表的 SQL 命令:</td>
<td></td>
</tr>
</tbody>
</table>
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">/*创建表的sql语句*/<span class="highl
ght-w">
<span cla
s="highlight-w">CREATE<span class="highlight
w"> TABLE </span
`classes<span class=
highlight-o">` (</sp
n>
<span cla
s="highlight-w"> `i
` <span clas
="highlight-nb">int(<span class="highlight-mi"
11) <span c
ass="highlight-k">NOT <span class="highlight-

```
">NULL</span><span class="highlight-w"> </span><span class="highlight-n">AUTO_INCR
MENT</span><span class="highlight-p">,</span><span class="highlight-w">
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-w">  </span><span class="highlight-o">`</span><span class="highlight-n">n
me</span><span class="highlight-o">`</span><span class="highlight-w"> </span><span
lass="highlight-nb">varchar</span><span class="highlight-p">(</span><span class="highl
ght-mi">5</span><span class="highlight-p">)</span><span class="highlight-w"> </span
<span class="highlight-k">DEFAULT</span><span class="highlight-w"> </span><span clas
="highlight-k">NULL</span><span class="highlight-p">,</span><span class="highlight-w
>
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-w">  </span><span class="highlight-k">PRIMARY</span><span class="highli
ht-w"> </span><span class="highlight-k">KEY</span><span class="highlight-w"> </span
<span class="highlight-p">(</span><span class="highlight-o">`</span><span class="highl
ght-n">id</span><span class="highlight-o">`</span><span class="highlight-p">)</span>
span class="highlight-w">
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-w">  </span><span class="highlight-p">)</span><span class="highlight-w"> <
span><span class="highlight-n">ENGINE</span><span class="highlight-o">=</span><sp
n class="highlight-n">InnoDB</span><span class="highlight-w"> </span><span class="hig
light-n">AUTO_INCREMENT</span><span class="highlight-o">=</span><span class="highl
ght-mi">5</span><span class="highlight-w"> </span><span class="highlight-k">DEFAUL
</span><span class="highlight-w"> </span><span class="highlight-n">CHARSET</span><
pan class="highlight-o">=</span><span class="highlight-n">utf8</span><span class="hig
light-p">;</span><span class="highlight-w">
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-w">  </span><span class="highlight-cm">/*插入测试数据*/</span><span class=
highlight-w">
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-w">  </span><span class="highlight-k">INSERT</span><span class="highlight
w">  </span><span class="highlight-k">INTO</span><span class="highlight-w">  </span><
pan class="highlight-o">`</span><span class="highlight-n">classes</span><span class="h
ghlight-o">`</span><span class="highlight-w">  </span><span class="highlight-k">VALUE
</span><span class="highlight-w">  </span><span class="highlight-p">(</span><span cla
s="highlight-s1">'1'</span><span class="highlight-p">,</span><span class="highlight-w">
</span><span class="highlight-s1">'一班'</span><span class="highlight-p">);</span><sp
n class="highlight-w">
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-w">  </span><span class="highlight-k">INSERT</span><span class="highlight
w">  </span><span class="highlight-k">INTO</span><span class="highlight-w">  </span><
pan class="highlight-o">`</span><span class="highlight-n">classes</span><span class="h
ghlight-o">`</span><span class="highlight-w">  </span><span class="highlight-k">VALUE
</span><span class="highlight-w">  </span><span class="highlight-p">(</span><span cla
s="highlight-s1">'2'</span><span class="highlight-p">,</span><span class="highlight-w">
</span><span class="highlight-s1">'二班'</span><span class="highlight-p">);</span><sp
n class="highlight-w">
</span></span></span><span class="highlight-line"><span class="highlight-cl"><span cla
s="highlight-w">  </span><span class="highlight-k">INSERT</span><span class="highlight
w">  </span><span class="highlight-k">INTO</span><span class="highlight-w">  </span><
pan class="highlight-o">`</span><span class="highlight-n">classes</span><span class="h
ghlight-o">`</span><span class="highlight-w">  </span><span class="highlight-k">VALUE
</span><span class="highlight-w">  </span><span class="highlight-p">(</span><span cla
s="highlight-s1">'3'</span><span class="highlight-p">,</span><span class="highlight-w">
</span><span class="highlight-s1">'三班'</span><span class="highlight-p">);</span><sp
```

n class="highlight-w">
<span cla
s="highlight-w"> INSERT<span class="highlight
w"> INTO <
pan class="highlight-o">`classes<span class="h
ghlight-o">` VALUE
  (<span cla
s="highlight-s1">'4',
'四班');<sp
n class="highlight-w">
</code></pre>
<h4 id="2-1-1基本查询">2.1.1 基本查询</h4>
查询数据库中某个表的所有内容：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">SELECT </s
an>* <span class="h
ghlight-n">FORM &
t;table_name></s
an>
</code></pre>
例如查询 students 表中的所有内容
<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/%E5%
F%BA%E6%9C%AC%E6%9F%A5%E8%AF%A2.gif" alt="查询 students 表中的所有内容">
注意：
对于 select 语句来说，并不一定非要有 from 子句，例如如下语句
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>1+<span class=
highlight-mi">2; </
pan>/* 结果为3*/
</code></pre>
上述查询会直接计算出表达式的结果。虽然 SELECT 可以用作计算，但它并不是 SQL 的强项。
是，不带 FROM 子句的 SELECT 语句有一个有用的用途，就是用来判断当前到数据库的连接是否有
。许多检测工具会执行一条 SELECT 1;来测试数据库连接。
<h4 id="2-1-2-条件查询">2.1.2 条件查询</h4>
定义：
大部分情况下，我们查询一张表的时候并不想获取一张表中的所有内容，而是想从所有记录筛选出我
所需要，此时便需要我们在查询过程中对查询条件进行限制，这边是 <code>条件查询</code>
条件查询的语法：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">SELECT </s
an>* <span class="h
ghlight-k">FROM &
t;表名>
span class="highlight-w"> WHERE<span class=
highlight-w"> <
件表达式>
<span cla
s="highlight-w">
</code></pre>
例如查询 students 表中分数 <code>大于等于80 (score>=80)</code> 的学生信息
查询分数大于等于 80 的学生信息 sql 命令
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl"> SELECT</s

an> *<span class="h
ghlight-w"> FROM
/span>students <sp
n class="highlight-k">WHERE <span class="hig
light-n">score >
 80<span c
ass="highlight-p">;
</code></pre>
查询结果
<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/%E6%
F%A5%E8%AF%A21.gif" alt="条件查询">
条件表达式中常用的查询条件有如下这些
<table>
<thead>
<tr>
<th>查询条件</th>
<th>谓词</th>
</tr>
</thead>
<tbody>
<tr>
<td>比较</td>
<td>=,>,<,>=,<=,!=,<>,!>,!<； NOT+前边的比较符</td>
</tr>
<tr>
<td>确定范围</td>
<td>BETWEEN AND,NOT BETWEEN AND</td>
</tr>
<tr>
<td>确定集合</td>
<td>IN,NOT IN</td>
</tr>
<tr>
<td>字符匹配</td>
<td>LIKE,NOT LIKE</td>
</tr>
<tr>
<td>空值</td>
<td>IS NULL,IS NOT NULL</td>
</tr>
<tr>
<td>多重条件（逻辑运算）</td>
<td>AND,OR,NOT</td>
</tr>
</tbody>
</table>
<h4 id="2-1-3-多重条件查询">2.1.3 多重条件查询</h4>
在实际生产过程中我们查询一个表可能查询条件并不仅仅只有一个，此时我们便需要 <code>A
D</code>、<code>OR</code> 和 <code>NOT</code> 来进行连接和限定。
<code><条件一> AND <条件二></code>:查询结果既需要满足 < 条件一 > 同时
需要满足 < 条件二 >。
例如查询分数大于等于 60 且小于 80 的学生信息：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">SELECT </s

an>* <span class="h
ghlight-k">FROM st
dents WHERE</spa
> score<span class=
highlight-w"> >=<span class="highlight-mi"
60 and<spa
n class="highlight-w"> score<span class="highl
ght-w"> < </spa
>80;<span class="h
ghlight-w">
</code></pre>
查询结果：
<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/And%
6%9D%A1%E4%BB%B6%E6%9F%A5%E8%AF%A2.gif" alt="">
<code><条件一> OR <条件二></code>:查询结果需要满足条件一或者条件二。<b
>
例如查询分数小于 60 或者大于等于 80 的学生信息：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>* <span class="h
ghlight-k">from stu
ents where
span class="highlight-w"> score<span class="hi
hlight-w"> <<span class="highlight-mi">60<
span> or<span clas
="highlight-w"> score<span class="highlight-w
> >= <s
an class="highlight-mi">80;<span class="highli
ht-w">
</code></pre>
查询结果:
<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/20190
12190734.gif" alt="">
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>* <span class="h
ghlight-k">from stu
ents where
span class="highlight-w"> score<span class="hi
hlight-w"> = </spa
>90;<span class="h
ghlight-w">
<span cla
s="highlight-w">select<span class="highlight-w
> * <span c
ass="highlight-k">from <span class="highlight
n">students where<
span> NOT<span cl
ss="highlight-w"> score<span class="highlight
w"> = <sp
n class="highlight-mi">90;<span class="highlig
t-w">
</code></pre>
查询结果：

<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/20190
12192005.gif" alt="">
<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/20190
12191821.gif" alt="">
<h3 id="2-2-投影查询">2.2 投影查询</h3>
有时我们在查询一个表时，可能并不需要所有表的信息，而只是需要一个表的部分列，此时我们
可以通过 <code>SELECT 列1, 列2, 列3 FROM ...</code>，让结果集仅包含指定列。这种操作称为
code>投影查询</code>。
<code>eg:</code> 查询所有学生的姓名和班级信息
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>name,<span clas
="highlight-n">class_id <span class="highlight-
">from students</s
an>;
</code></pre>
查询结果：
<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/20190
12192744.gif" alt="">
同时在查询过程中我们可以对查询后的属性名称设计别名，并且也可以指定结果列的顺序（可以
原表的顺序不同）。
使用 <code>SELECT 列1, 列2, 列3 FROM ..</code>.时，还可以给每一列起个别名，这样，结果集
列名就可以与原表的列名不同。它的语法是 <code>SELECT 列1 别名1, 列2 别名2, 列3 别名3 FROM .
.</code>。
<code>eg:</code> 将插叙结果中属性名 <code>name</code> 改成 <code>student_name</c
de>
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>name <span clas
="highlight-n">student_name,<span class="hig
light-n">class_id fr
m students
span class="highlight-p">;
</code></pre>
查询结果：
<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/20190
12194546.gif" alt="">
<h3 id="2-3查询结果排序">2.3 查询结果排序</h3>
细心的读者可能已经发现：我们在之前所做的查询最终的查询结果都是按照 id（或者 class_id）
升序排列的，那么我们如何来改变查询结果的排序顺哪？
其实我们可以通过 <code>ORDER BY</code> 语句来进行查询结果出顺序的控制。
<code>eg:</code> 按照 <code>score</code> 从小到大对结果进行排序
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>* <span class="h
ghlight-k">from stu
ents ORDER</span
 BY<span class="hig
light-w"> score;</sp
n>
</code></pre>
查询结果：
<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/20190
13085927.gif" alt="">

默认的排序规则是 <code>ASC</code> 升序即从小到大，当然如果我们想要查询结果是降序列，我们可以通过加上 <code>DESC</code> 来进行进行降序结果输出。
<code>eg</code>：将上边的查询结果按照 <code>score</code> 降序来进行输出

<h3 id="2-4-分页查询">2.4 分页查询</h3>
有时在查询的过程中我们查询的到的结果集比较大，而程序在处理和显示这些数据时空间有限不够一下子完全显示出来，此时便需要将查询到的结果集分成不同的页来进行显示，这边是 <code>分查询</code>。<code>分页查询</code> 实际上也就是将大的数据集（比如几万条）进行拆分，成若干页，比如 <code>1-100</code> 第一页、<code>101-200</code> 第二页、<code>201-00</code> 第三页，以次类推。具体使用分页查询时我们需要通过 <code> LIMIT <M> OFFST <N></code> 子句对查询结果集的大小以及页数进行控制。我很还以 <code>students</coe> 为例，首先查询到它所有的结果集
查询 <code>studnets</code> 表所有结果集的 sql 语句
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>* <span class="h
ghlight-k">from stu
ents;
</code></pre>
查询结果：

现在我们将查询到数据集进行分页，
查询第一页的数据（每页 3 条数据）：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>* <span class="h
ghlight-k">from stu
ents limit<
pan class="highlight-w"> 3<span class="highli
ht-w"> offset </spa
>0;<span class="hi
hlight-w">
</code></pre>
查询结果为：

上述查询 <code>limit 3 offset 0</code> 表示，结果集从 0 号记录开始查询。注意/strong> SQL 记录集的索引从 0 开始。如果需要查询第 2 页，我们需要跳过前边的三条记录，索引时应该从 3 开始，即我们需要将 <code>offest</code> 设置为 3。
查询第 2 页的结果集：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">/*每页显示三条数据，获取其中的第2页*/</span

<span cla
s="highlight-w">select<span class="highlight-w
> * <span c
ass="highlight-k">from <span class="highlight
n">students limit</
pan> 3<span class
"highlight-w"> offset<span class="highlight-w"
 3;<span c
ass="highlight-w">

</code></pre>
查询结果：

同样的查询第 3 页的结果集时应该讲 <code>offset</code> 设置为 6；
查询第 3 页的结果集：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>* <span class="h
ghlight-k">from stu
ents limit<
pan class="highlight-w"> 3<span class="highli
ht-w"> offset </spa
>6;<span class="hi
hlight-w">
</code></pre>
查询结果：

查询第 4 页的结果集：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>* <span class="h
ghlight-k">from stu
ents limit<
pan class="highlight-w"> 3<span class="highli
ht-w"> offset </spa
>9;<span class="hi
hlight-w">
</code></pre>
查询结果为：

由于第 4 页只有一条记录所以查询结果只显示一条记录。<code>limit 3</code> 表示每页最多“
示三条记录”。
由此可见我们在进行分页查询的时候最关键的问题是设计每页需要显示的结果集大小 <code>pageSi
e</code>(这里设置的是 3)，然后根据当前页的索引 <code>pageIndex</code>（需要查第几页
结果），确定 <code>limit</code> 以及 <code>offset</code> 的值：

<code>limit</code> 一般设置为 <code>pageSize</code>
<code>offset</code> 设置为 <code>pageSize*(pageIndex-1)</code>

注意：

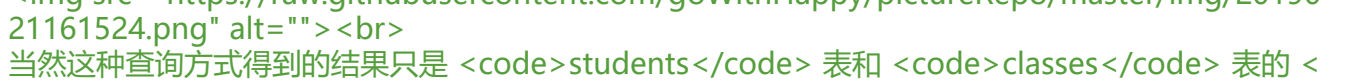
<code>offset</code> 值的设置是可选的，如果只写 <code>limit 3</code>，DBMS 不会
错，而是默认认为是 <code>limit 3 offset 0</code>。
<code>offset</code> 设置的值如果大于最大数量并不会报错，而只是得到一个空的结果集</l
>
在 <code>MYSQL</code> 中，<code>limit 3 offset 6</code> 还可以写为 <code>limit 3,
</code>
在使用 <code>limit <m> offset <n></code> 时随着 n 的值越来越大，查询的效
也会越来越低

<h3 id="2-5-聚集查询">2.5 聚集查询</h3>

在日常开发的某些应用场景中，我们并不需要获得具体数据集，而只是想要获得满足条件的数据的条数（例如：查询班级表中男生的人数），此时便需要 <code>嵌套查询</code>。对于统计总数、平均数这类查询来说，<code>SQL</code> 已经为我们提供了专门的 <code>聚合函数</code>，使用聚合函数进行查询，我们便称之为 <code>聚合查询</code>，仍然以查询 <code>students</code> 表中男生的人数为例,我们可以通过 <code>SQL</code> 内置的 <code>count()</code> 函数来进行查询。
查询 <code>students</code> 表中男生的人数：
<pre><code class="language-sql highlight-chroma">select count(*) from students where gende='M';
</code></pre>
查询结果：

当然除了 <code>count()</code> 函数之外 <code>SQL</code> 还提供了如下的聚合函数
<table>
<thead>
<tr>
<th>函数</th>
<th>说明</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUM</td>
<td>计算某一列的合计值，该列必须为数值类型</td>
</tr>
<tr>
<td>AVG</td>
<td>计算某一列的平均值，该列必须为数值类型</td>
</tr>
<tr>
<td>MAX</td>
<td>计算某一列的最大值</td>
</tr>
<tr>
<td>MIN</td>
<td>计算某一列的最小值</td>
</tr>
</tbody>
</table>
注意：

<code>MAX()</code> 个 <code>MIN()</code> 函数并不仅限于数值类型。如果字符类型 <code>MAX()</code> 和 <code>MIN()</code> 会返回排序在最后边和最前边的字符
如果 <code>聚合查询</code> 的结果没有匹配到任何行，<code>count()</code> 会返回 0 而 s <code>um()</code>、<code>avg()</code>、<code>max()</code> 和 <code>min()</code> 会返回 <code>null</code>。

<h3 id="2-6-多表查询">2.6 多表查询</h3>
<code>select</code> 查询不仅可以从一张表中查询出结果，还可以同时在多张表中查询出结，其语法为：<code>select * form <table 1>,<table 2></code>。
例如，从同时从 <code>students</code> 表和 <code>classes</code> 表中查询出结果：
查询所用 sql 语句：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>* <span class="h
ghlight-k">from stu
ents,classes
span class="highlight-p">;
</code></pre>
查询结果为：


当然这种查询方式得到的结果只是 <code>students</code> 表和 <code>classes</code> 表的 <ode>笛卡尔积</code>，它是 <code>students</code> 表和 <code>classes</code> 表的“积”，即 <code>students</code> 表的每一行与 <code>classes</code> 表的每一行都两两拼一起返回。结果集的列数是 <code>students</code> 表和 <code>classes</code> 表的列数之，行数是 <code>students</code> 表和 <code>classes</code> 表的行数之积。
当然如果简单的使用上边的查询方法不加限定条件，那么查询的结果几乎没有任何意义。因此我们可通过 <code>where</code> 子句对查询的结果进行限定。
例如查询男生且位于一班的信息
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">SELECT
<span cla
s="highlight-w">   s
id <span cl
ss="highlight-n">sid,<span class="highlight-w"

<span cla
s="highlight-w">   s
name,<span
class="highlight-w">
<span cla
s="highlight-w">   s
gender,<sp
n class="highlight-w">
<span cla
s="highlight-w">   s
score,<span
class="highlight-w">
<span cla
s="highlight-w">   c
id <span cl
ss="highlight-n">cid,<span class="highlight-w"

<span cla
s="highlight-w">   c
name <spa
class="highlight-n">cname
<span cla
s="highlight-w"> FROM<span class="highlight-

"> students s, classes c
WHERE s.gender = 'M' AND c.id = 1;
</code></pre>
查询结果为：

<h3 id="2-7-连接查询">2.7 连接查询</h3>
在上一小结我们提到的连接查询，所得到的结果只是两个表 <code>stuents</code> 表 <code>和classes</code> 表的笛卡尔积，两个表直接没有任何的逻辑联系，而实际上我们进行多表查询更多情况下两张表之间是有关系联系的，比如将 <code>students</code> 表中 <code>class_id</code> 和 <code>classes</code> 表中的 <code>id</code> 建立相等的联系 <code>stduents.class_id=classes.id</code>，这种连接两个表进行 <code>JOIN</code> 运算的的查询方式，我们之为 <code>连接查询</code>。
<code>连接查询</code> 是我们在实际开发过程最常用的查询方式，<code>连接查询</code> 查询过程中又分为 <code>自然连接查询</code>、<code>内连接查询</code>、<code>外连接询</code> 等。
<code>自然连接查询</code>：在查询过程中，我们将目标列中重复的属性列去掉这一过程我们称为 <code>自然连接查询</code>。
例如我们查询所有学生信息（包括班级信息），其中信息如姓名、性别、id 等信息都在 <code>students</code> 表中，而班级信息却在 <code>classes</code> 表中，此时如果直接通过上一节的夺查询的查询方法进行查询。这样查询的结果会有许多重复的结果值。此时我们便可以通过 <code>自连接查询的方式</code>（限定 <code>students</code> 表和 <code>classes</code> 表的属关系）来进行查询。
自然连接查询方法查询所有学生信息：
<pre><code class="language-sql highlight-chroma">select s.id,s.name,s.class_id,s.gender,s.score,c.name from student s,classes c where s.class_id</sp

n>=c<span class="h
ghlight-p">.id
</code></pre>
查询结果为：


当然我们在查询班级所有信息时我们也可以通过 <code>内连接查询</code> 来获取相同的查询结
。
内连接查询方式查询所有学生信息：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>s.<span class="hi
hlight-n">id,s</spa
>.name<span class=
highlight-p">,s.</sp
n>class_id,<span cla
s="highlight-n">s.g
nder,s<span
class="highlight-p">.score<span class="highligh
-p">,c.<spa
class="highlight-n">name <span class="highli
ht-k">from student
 s<span cla
s="highlight-w"> inner<span class="highlight-w
> join <spa
class="highlight-n">classes <span class="highl
ght-k">c on</span
 s<span class="highl
ght-p">.class_id=</
pan>c.<span class=
highlight-n">id;
</code></pre>
查询结果为：

此时可能有人要问了，既然自然连接查询和内连接查询可以相同结果，那么我们在实际开发过程中应
选择哪种查询方法？
针对这个问题，首先我们要明白虽然通过 <code>自然连接查询</code> 以及 <code>内连接查询<
code> 可以得到相同的查询结果，但是它们在底层的实现原理是不同的。一般来说能获得相同的查
结果条件下，我们也一般都是通过 <code>内连接来查询</code> 的，因为内连接使用 <code>ON
/code>，而 <code>自然连接</code> 是通过使用 <code>WHERE子句</code> 来进行限定，而
WHERE 的效率没有 <code>ON</code> 高（<code>ON</code> 指匹配到第一条成功的就结束
其他不匹配；若没有，不进行匹配,而 <code>WHERE</code> 会一直匹配，进行判断。）
注意： inner join 查询的写法为：

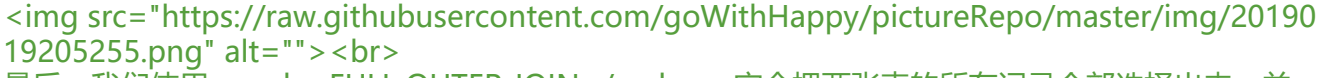
先确定主表，仍然使用 FROM < 表 1> 的语法；
再确定需要连接的表，使用 INNER JOIN < 表 2> 的语法；
然后确定连接条件，使用 ON < 条件...>，这里的条件是 s.class_id = c.id，表示 students
表的 class_id 列与 classes 表的 id 列相同的行需要连接；
可选：加上 WHERE 子句、ORDER BY 等子句。

那可能又有人问了“既然有 <code>外连接</code>，那么是否有 <code>内连接</code> 那
” 答案是肯定的，我们暂且不说什么是外连接，我们先通过一个例子来看下 <code>内连接</code>
和 <code>外连接</code> 之间的区别，我们还是查询所有学生的信息，但是不同之处在于我们将 <

ode>内连接</code> 换成 <code>外连接</code>。
通过 <code>外连接</code> 查询所有学生信息:
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">SELECT </s
an>s.<span class="h
ghlight-n">id, </sp
n>s.<span class="hi
hlight-n">name, </
pan>s.<span class=
highlight-n">class_id,<span class="highlight-w
> c.<span cl
ss="highlight-n">name <span class="highlight
n">class_name, </s
an>s.<span class="h
ghlight-n">gender,
/span>s.<span class
"highlight-n">score
<span cla
s="highlight-w">
FROM<span class="highlight-
"> students </span
s
<span cla
s="highlight-w">
RIGHT<span class="highlight-
"> OUTER
span class="highlight-k">JOIN <span class="hi
hlight-n">classes c<
span>
<span cla
s="highlight-w">
ON
s.<span clas
="highlight-n">class_id <span class="highlight
o">= c<sp
n class="highlight-p">.id<span class="highlight
p">;
</code></pre>
查询结果为：

我们容易发现，此时的查询结果比 <code>内连接</code> 方式多了一行，多出来的一行是“<cod
>四班</code>”，但是，学生相关的列如 <code>name</code>、<code>gender</code>、<
ode>score</code> 都为 <code>NULL</code>。这也容易理解，因为根据 <code>ON</code>
条件 <code>s.class_id = c.id</code>，<code>classes</code> 表的 <code>id=4</code> 的
正是 <code>“四班”</code>，但是，<code>students</code> 表中并不存在 <code>class_id
4</code> 的行。
当然有 <code>RIGHT OUTER JOIN</code>，就有 <code>LEFT OUTER JOIN</code>，以及 <
ode>FULL OUTER JOIN</code>。它们的区别是：<code>INNER JOIN</code> 只返回同时存
于两张表的行数据，由于 <code>students</code> 表的 <code>class_id</code> 包含 1，2，3
<code>classes</code> 表的 id 包含 1，2，3，4，所以，<code>INNER JOIN</code> 根据条件
<code>s.class_id = c.id</code> 返回的结果集仅包含 1，2，3。<code>RIGHT OUTER JOIN</c
de> 返回右表都存在的行。如果某一行仅在右表存在，那么结果集就会以 <code>NULL</code>
充剩下的字段。<code>LEFT OUTER JOIN</code> 则返回左表都存在的行。如果我们给 <code>st
dents</code> 表增加一行，并添加 <code>class_id=5</code>，由于 <code>classes</code> 1
表并不存在 <code>id=5</code> 的行，所以，<code>LEFT OUTER JOIN</code> 的结果会增
一行，对应的 <code>class_name</code> 是 <code>NULL</code>:

先插入一个不含 <code>class_id</code> 的学生信息：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">insert </sp
n>into <span class=
highlight-n">stduents <span class="highlight-p
>(id,<span c
ass="highlight-w"> name<span class="highlight
p">,gender,</span
score)<span class="h
ghlight-w"> values(
/span>default,<span
class="highlight-s1">'李平',<span class="highli
ht-s1">'M',95</sp
n>);
<span cla
s="highlight-w">
select<span class="highlight-w
> * <span c
ass="highlight-k">from <span class="highlight
n">students;
</code></pre>
<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/2
190919204540.png" alt="">
通过外连接查询所有学生信息：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">SELECT </s
an>s.<span class="h
ghlight-n">id, </sp
n>s.<span class="hi
hlight-n">name, </
pan>s.<span class=
highlight-n">class_id,<span class="highlight-w
> c.<span cl
ss="highlight-n">name <span class="highlight
n">class_name, </s
an>s.<span class="h
ghlight-n">gender,
/span>s.<span class
"highlight-n">score
<span cla
s="highlight-w">
FROM<span class="highlight-
"> students </span
s
<span cla
s="highlight-w">
LEFT<span class="highlight-w
> OUTER <
pan class="highlight-k">JOIN <span class="hig
light-n">classes c</
pan>
<span cla
s="highlight-w">
ON
s.<span clas
="highlight-n">class_id <span class="highlight
o">= c<sp
n class="highlight-p">.id<span class="highlight

p">;
</code></pre>
查询结果为:


最后，我们使用 <code>FULL OUTER JOIN</code>，它会把两张表的所有记录全部选择出来，并
，自动把对方不存在的列填充为 <code>NULL</code>：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">SELECT </s
an>s.<span class="h
ghlight-n">id, </sp
n>s.<span class="hi
hlight-n">name, </
pan>s.<span class=
highlight-n">class_id,<span class="highlight-w
> c.<span cl
ss="highlight-n">name <span class="highlight
n">class_name, </s
an>s.<span class="h
ghlight-n">gender,
/span>s.<span class
"highlight-n">score
<span cla
s="highlight-w">
<span cla
s="highlight-w">FROM<span class="highlight-
"> students </span
s
<span cla
s="highlight-w">FULL<span class="highlight-w
> OUTER <
pan class="highlight-k">JOIN <span class="hig
light-n">classes c</
pan>
<span cla
s="highlight-w">ON
s.<span clas
="highlight-n">class_id <span class="highlight
o">= c<sp
n class="highlight-p">.id<span class="highlight
p">;
</code></pre>
查询结果：
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>class_id</th>
<th>class_name</th>
<th>gender</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>

<td>1</td>
<td>小明</td>
<td>1</td>
<td>一班</td>
<td>M</td>
<td>90</td>
</tr>
<tr>
<td>2</td>
<td>小红</td>
<td>1</td>
<td>一班</td>
<td>F</td>
<td>95</td>
</tr>
<tr>
<td>3</td>
<td>小军</td>
<td>1</td>
<td>一班</td>
<td>M</td>
<td>88</td>
</tr>
<tr>
<td>4</td>
<td>小米</td>
<td>1</td>
<td>一班</td>
<td>F</td>
<td>73</td>
</tr>
<tr>
<td>5</td>
<td>小白</td>
<td>2</td>
<td>二班</td>
<td>F</td>
<td>81</td>
</tr>
<tr>
<td>6</td>
<td>小兵</td>
<td>2</td>
<td>二班</td>
<td>M</td>
<td>55</td>
</tr>
<tr>
<td>7</td>
<td>小林</td>
<td>2</td>
<td>二班</td>
<td>M</td>
<td>85</td>

</tr>
<tr>
<td>8</td>
<td>小新</td>
<td>3</td>
<td>三班</td>
<td>F</td>
<td>91</td>
</tr>
<tr>
<td>9</td>
<td>小王</td>
<td>3</td>
<td>三班</td>
<td>M</td>
<td>89</td>
</tr>
<tr>
<td>10</td>
<td>小丽</td>
<td>3</td>
<td>三班</td>
<td>F</td>
<td>88</td>
</tr>
<tr>
<td>11</td>
<td>李平</td>
<td>5</td>
<td>NULL</td>
<td>M</td>
<td>95</td>
</tr>
</tbody>
</table>
注意： MYSQL 是不支持 FULL OUTER JOIN 查询的。
为了便于大家理解 JON 查询，下边我们用图来表示各种查询的关系：
假设查询语句是：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">SELECT
</s
an>...
<span class="
ighlight-k">FROM
t
bleA
???<s
an class="highlight-w">
JOIN<span class="highl
ght-w">
tableB
</s
an>ON
<span class
"highlight-n">tableA.<span class="highlight-n"
column1
=
span class="highlight-w">
tableB<span class="h
ghlight-p">.column2

</code></pre>
我们把 tableA 看作左表，把 tableB 看成右表，那么 INNER JOIN 是选出两张表都存在的记录

<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/20190
29091941.png" alt="">
LEFT OUTER JOIN 是选出左表存在的记录：
<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/20190
29092025.png" alt="">
RIGHT OUTER JOIN 是选出右表存在的记录：
<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/20190
29092059.png" alt="">
FULL OUTER JOIN 则是选出左右表都存在的记录：
<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/20190
29092122.png" alt="">
<h3 id="2-7-嵌套查询">2.7 嵌套查询</h3>
基于上边几节的描述常用的查询方法都已基本囊括，但是某些情况下我们需要将一个 <code>查
块</code>（一个 <code>SELECT-FROM-WHERE</code> 子句称为 <code>查询块</code>）
套在另一个 <code>查询块</code> 的 <code>WHERE 子句</code> 或 <code>HAVING 短语</
ode> 的条件中的查询称为 <code>嵌套查询</code>。例如：
通过嵌套查询 <code>二班</code> 的学生信息：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>* <span class="h
ghlight-k">from stu
ents where
span class="highlight-w"> class_id<span class=
highlight-w"> in
<span cla
s="highlight-w">(sel
ct id<span c
ass="highlight-w"> from<span class="highlight
w"> classes </span
where <span class=
highlight-n">name=<span class="highlight-s1"
'二班');<sp
n class="highlight-w">
</code></pre>
查询结果:
<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/20190
21161616.png" alt="">
<code>嵌套查询</code> 可以使用户使用多个简单查询构成复杂的查询，从而增强 <code>S
L</code> 的查询能力。以层层嵌套的方式来构造程序正式 <code>SQL</code> 中“结构化”的
义所在。在使用 <code>嵌套查询</code> 的过程中根据 <code>子查询</code> 的方式不同，我
将查询分为下边三大类：
<h4 id="2-7-1-使用IN谓词的子查询">2.7.1 使用 IN 谓词的子查询</h4>
在 <code>嵌套查询</code> 中，<code>子查询</code> 的结果往往是一个集合，所以谓词
N 是 <code>嵌套查询</code> 中经常使用的谓词。比如我们要查询跟 <code>小丽</code> 同
考同样分数学生的信息我们可以通过如下步骤来构造嵌套查询。

首先确定小丽同学所考的分数：
查询小丽同学所考的分数：

<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>score <span clas
="highlight-k">from <span class="highlight-n"
students where</sp

n> name<span clas
="highlight-o">='小丽'<span class="highlight-
">;
</code></pre>
查询结果为：


<ol start="2">
查询 <code>students</code> 表中所有 <code>score=85</code> 的学生信息:
查询 <code>score=85</code> 的学生信息：

<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>* <span class="h
ghlight-k">from stu
ents where
span class="highlight-w"> score<span class="hi
hlight-o">=85;</s
an>
</code></pre>
查询结果:


3. 将第一步构造的查询语句嵌套到第二句中构造 <code>嵌套查询</code>。
<code>查询语句为：</code>
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>* <span class="h
ghlight-k">from stu
ents where
span class="highlight-w"> score<span class="hi
hlight-w"> in
<span cla
s="highlight-w">(sel
ct score<spa
n class="highlight-w"> from<span class="highli
ht-w"> students </s
an>where <span cla
s="highlight-n">name=<span class="highlight-
1">'小丽');
</code></pre>
查询结果为：


在这个例子程序中细心的同学会发现，<code>子查询</code> 的查询条件是和 <code>父查询</c
de> 是相互独立的，对于此类查询我们称之为 <code>不相关子查询</code>。对于此种查询我们
际上是可以通过连接查询来获取同样的查询结果的。例如上边“查询跟 <code>小丽</code> 同学
同样分数学生的信息”，我们可以通过如下的 SQL 语句进行替代：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>s1.<span class="h
ghlight-o">* from</
pan> students<span
class="highlight-w"> s1<span class="highlight-

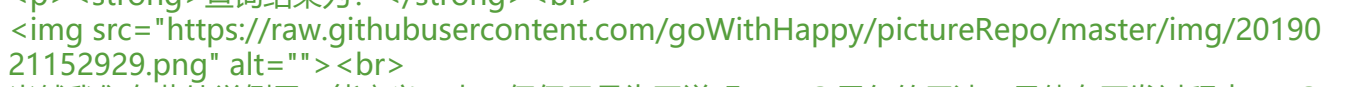
">,students </span
s2 <span class="hig
light-k">where s1</
pan>.score<span cla
s="highlight-o">=s2
score <span
class="highlight-k">and <span class="highlight
n">s2.name</span
='小丽'<span class=
highlight-p">;
</code></pre>
查询结果：
<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/20190
21133240.png" alt="">
由此可见实现同一个结果的查询方式有很多种，但是不同方法查询的效率确实不同的。这就是数
编程人员需要掌握的数据性能调优技术，后续时间充裕，我会对常用的 mysql 查询的调优技术进行
结，此处便不进行详尽阐述了。
<h4 id="2-7-2-带有比较符号的">2.7.2 带有比较符号的 <code>子查询</code></h4>
有时某些 <code>嵌套子查询</code> 中，我们并不需要通过 <code>IN</code> 详尽匹配
查询中的结果，而只是需要对子查询返回的单个结果进行比较，此时我们便需要使用 <code>比较运
符</code> 对查询结果进行限定。比如我们查询分数高于平均分的学生信息：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>id,<span class="h
ghlight-n">class_id,
ame,gender</span
,score<span class="h
ghlight-w"> from </
pan>students s <span class="highlight-k
>where score</spa
> >=<span class
"highlight-w">
<span cla
s="highlight-w"> (sel
ct avg(score<span class="highlig
t-p">) from</span
 students<span clas
="highlight-p">);
</code></pre>
查询结果为：
<img src="https://raw.githubusercontent.com/goWithHappy/pictureRepo/master/img/20190
21134702.png" alt="">
注意：在使用比较符号时子查询的应当只有一个否则，查询结果是不正确的。
<h4 id="2-7-3-带有-或者位于的子查询">2.7.3 带有 <code>ANY（SOME）</code> 或者 <code
ALL</code> 位于的子查询</h4>
子查询的结果有一个时可以使用比较运算符，但是当返回值有多个时我们需要通过 <code>ANY
SOME）</code> 或者 <code>ALL</code> 等谓语以及和比较符号一起连用来对结果进行限定，
见的用法以及含义如下表所示
<table>
<thead>
<tr>
<th>限定符</th>

<th>作用</th>
</tr>
</thead>
<tbody>
<tr>
<td>>ANY</td>
<td>大于子查询中的某个结果值</td>
</tr>
<tr>
<td>>ALL</td>
<td>大于子查询的所有结果值</td>
</tr>
<tr>
<td><ANY</td>
<td>小于子查询结果中的某个值</td>
</tr>
<tr>
<td><ALL</td>
<td>小于子查询结果中的所有值</td>
</tr>
<tr>
<td>>=ANY</td>
<td>大于等于子查询中的某个结果只</td>
</tr>
<tr>
<td>>=ALL</td>
<td>大于等于子查询中的所有结果只</td>
</tr>
<tr>
<td><=ANY</td>
<td>小于等于子查询中的某个结果值</td>
</tr>
<tr>
<td><=ALL</td>
<td>小于等于子查询的所有结果值</td>
</tr>
<tr>
<td>=ANY</td>
<td>等于子查询中的某个结果值</td>
</tr>
<tr>
<td>=ALL</td>
<td>等于子查询中的所有结果值（通常没有任何意义）</td>
</tr>
<tr>
<td>!=ANY</td>
<td>不等于子查询中的某个结果值</td>
</tr>
<tr>
<td>!=ALL</td>
<td>不等于子查询中的所有结果值</td>
</tr>
</tbody>
</table>

比如查询所有学生中分数最低的学生的姓名和成绩
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>name,<span clas
="highlight-n">score <span class="highlight-k
>from students</sp
n> where<span clas
="highlight-w"> score<span class="highlight-o
><=ALL(
span class="highlight-k">select <span class="h
ghlight-n">score fr
m students
span class="highlight-p">);
</code></pre>
查询结果为：

<h4 id="2-7-4-带的子查询">2.7.4 带 <code>EXISTS</code> 的子查询</h4>
<code>EXISTS</code> 代表存在量词。带有 EXISTS 谓词的子查询不返回任何数据，只产生
辑真“true”和逻辑假“flase”。
例如我们可以通过 <code>嵌套查询</code>，查询 <code>二班</code> 所有学生的信息。
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp
n>* <span class="h
ghlight-k">from stu
ents where
span class="highlight-w"> exists<span class="h
ghlight-w">
<span cla
s="highlight-w">(sel
ct *<span c
ass="highlight-w"> from<span class="highlight
w"> classes </span
where <span class=
highlight-n">students.<span class="highlight-n
>class_id=id</span
 and<span class="hi
hlight-w"> name=</
pan>'二班');<span c
ass="highlight-w">
</code></pre>
查询结果：

注意：通过 <code>EXISTS</code> 引出的子查询，其目标表达式通常都是使用 <code>*
/code>，因为带有 <code>EXISTS</code> 的子查询只返回真值或者假值，给出列名没有任何意
。
<h3 id="2-8-基于派生表的查询">2.8 基于派生表的查询</h3>
其实子查询有时候并不一定非要出现在 <code>WHERE子句</code> 中，还可以出现在 <cod
>FROM子句</code> 中，这是子查询产生的 <code>临时派生表</code> 成为朱查询的查询对象
这种查询方式我们称为 <code>基于派生表的查询</code>。
例如查询所有学生的姓名信息我们通过派生表查询方式实现：
<pre><code class="language-sql highlight-chroma"><span cla
s="highlight-cl">select </sp

n>s1.<span class="h
ghlight-n">name fr
m
<span cla
s="highlight-w">(sel
ct *<span c
ass="highlight-w"> from<span class="highlight
w"> students)</spa
> as<span class="hi
hlight-w"> s1
</code></pre>
查询结果为：

当然我们在此处举例子可能意义不大，仅仅只是为了说明 EXITS 子句的用法。具体在开发过程中 EXIS
S 的具体使用，需要读者自己去发掘和探索。