# MySqlNote

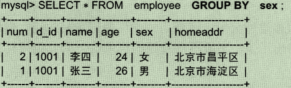
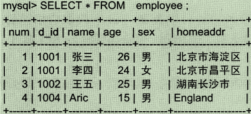
## 查询（SELECT）

### 分组（GROUP BY）

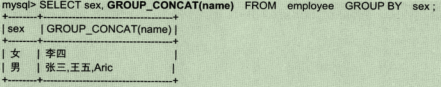
语法：GROUP BY 属性名 [HAVING 条件] [WITH ROLLUP]

“属性名“是指按照该字段的值进行分组，“HAVING 条件 ”是用来限制分组后的结果，“WITH ROLLUP“ 将会在所有记录的最后加上一条记录，该记录是上面所有记录的总和。

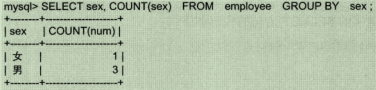
1. 不使用集合函数，则只显示每个分组中的第一条记录，意义不大。



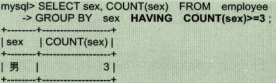
1. **与GROUP\_CONCAT(属性名)**：该函数会将分组中指定的属性，全部显示出来



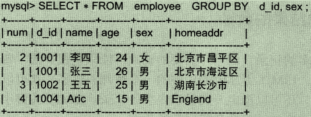
1. **与集合函数**：通常情况下，GROUP BY 与集合函数一起使用，集合函数包括COUNT(),SUM(),AVG(),MAX(),MIN().通常先使用group by 将记录分组，然后对每个分组使用集合函数进行计算。在统计时经常需要使用GROUP BY和集合函数



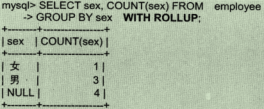
1. **HAVING :**与WHERE一样都是用来限制显示的，区别：where作用与表或者视图，是表和视图的查询条件。HAVING作用于分组后的记录，用于选择满足条件的组。

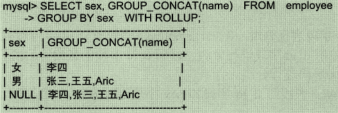


1. **多字段进行分组：**先按照第一个字段分组，如果字段的值相等，则按照第二个字段分组，以此类推。



1. **WITH ROLLUP:** 将在所有记录的最后加上一条记录，该记录是上面所有记录的中使用集合函数列的总和，如：COUNT(),GROUP\_CONCAT()。





## 表连接

**外连接与内连接的主要区别**是：内连接仅仅选出两张表中互相匹配的记录，而外连接会选出其他不匹配的记录。

## 索引

|  |  |  |  |
| --- | --- | --- | --- |
| 元组 | 属性（字段）1 | 属性（字段）2 | 属性（字段）3 |
| 元组1 |  |  |  |
| 元组2 |  |  |  |

**定义：**

索引是一种数据结构，提高对一个或多个属性的查找速率。就相当于书本里的目录，可以快速找到相关的内容，而不用从头开始一页一页的查找。

索引可以来自任何一个或多个属性，而不必主键。

**作用：**

当关系表变得很大的时候，通过扫描关系表中所有的元组来找出符合查询条件的元组代价太高。

SELECT \*

FROM `city`

WHERE name='Kabul' AND CountryCode='AFG';

如上的SQL语句，如果对CountryCode建立索引，那么就可以快速找到满足‘AFG’的元组，然后再筛选出满足name='Kabul'的元组。

**声明：**

CREATE INDEX nameIndex ON 表名(属性1，属性2……)

对于多属性索引，需要注意的是，最好按照哪些属性可能被查询的次数多少来排序，

如：如果对属性1查询要比属性2要多，则建立（属性1，属性2），反之建立（属性2，属性1）

**优缺点：**

1. 如果在属性上存在索引，则对该属性的进行定值或范围查询（name=’ jizx’或者age<19）,就会极快。
2. 索引的建立会使得对关系的插入，删除和修改变得更加复杂，费时。

**选择：**

代价模型：

一个关系表，被分配到磁盘的多个页面上，每个页面存储多个元组。

查询时，哪怕只检查一个元组也要将包含它的整个磁盘页加载如内存。因此，检查一个页面上所有分组与检查一个元组所花时间几乎没有差别。

**因此减少代价，提高索引效率就是将要加载如内存的页面尽量减少的最少。基于如此，可以参考如下原则建立索引**

1. **主键**建立索引：因为主键上的索引会被频繁的使用，而且键值是唯一的，对于给定的值，最多只有一个元组匹配，最多只有一个磁盘页需要被读入内存
2. **非主键**建立索引：分情况，可能会加速检索速度，可能不会，如下情况的索引是有效的：
   1. 该属性几乎可以看作是一个键，也就是说基本上没有多少元组在该属性上符合给定查询条件的值。因此即使每个满足条件的元组分别在不同磁盘页上，也不需要检索大量的磁盘页。
   2. 元组按该属性在磁盘也上“聚集在一起”。也就是说在该属性上具有相同值的元组，被分组到尽可能少的磁盘页里。在这样的情况下，即使符合条件的元组可能非常多，但是却不必检索与符合要求的元组数目同样多的磁盘页。

例子：

SELECT \*

FROM Movies

WHERE year = 1990；

首先假设Movies的元组不是按照year进行“聚集”的，而是按照title属性的值按照词典排序。那么属性year上的索引基本不会带来任何改进。因为：如果每个磁盘页存储100个元组，那么有相当的几率使得每页至少包含一部1990的电影。这样，用于存储关系Movies的页将会被大量地读入内存。

然而，如果Movies元组是按照year值进行“聚集“的，则利用其上的索引就会找到只有少量的页包含year=1990的元组。

反过来，如果在year和title上都建立索引将不会起任何作用，并且不论是在哪个属性上进行聚集？？？

**最佳索引选择：**

1. 如果插入操作时主要的，只有少量查询，那么就不要索引。
2. 如果某类查询特别频繁，那么就仅仅创建该字段的索引
3. 如果都需要大量的查询，那么可以建立2个或多个索引。