Atitit redis 存储数据与sql语法的转换

目录

[1. Table的模拟，使用hash模拟比较好 1](#_Toc22665)

[1.1. Where 查询 where col1=1 and col2=2 1](#_Toc5958)

[1.2. Orderby 排序 使用 Redis 有序集合(sorted set) 1](#_Toc22928)

[1.3. 1、ORDER BY的索引优化。如果一个SQL语句形如： 2](#_Toc30045)

[1.4. Group 分组 ，分组列作为key+有序列表 3](#_Toc8040)

[1.5. Top n 使用有序列表 3](#_Toc30886)

[1.6. 聚合函数sum avg等reduce运算 3](#_Toc14268)

[1.7. count等有api实现 3](#_Toc26422)

[1.8. 行的保存 kv字符串 4](#_Toc27399)

# Table的模拟，使用hash模拟比较好

Hash可以根据主键pk获取记录。。。

其他排序索引可以使用有序列表比较好，或者list

或者索引记录一体化List

只能使用hash作为存在主题，因为可以有pk作为记录。。如果以list存储，主键id不好查询。

或者一体化有序列表。。zset

## Where 查询 where col1=1 and col2=2

主键可以直接放入key\_pk。。非主键建立额外索引 key\_col1\_col2

## Orderby 排序 使用 Redis 有序集合(sorted set)

Redis 有序集合(sorted set)

Redis 有序集合和集合一样也是 string 类型元素的集合,且不允许重复的成员。

不同的是每个元素都会关联一个 double 类型的分数。redis 正是通过分数来为集合中的成员进行从小到大的排序。

有序集合的成员是唯一的,但分数(score)却可以重复。

集合是通过哈希表实现的，所以添加，删除，查找的复杂度都是 O(1)。 集合中最大的成员数为 232 - 1 (4294967295, 每个集合可存储40多亿个成员)

通过索引优化来实现MySQL的ORDER BY语句优化：

## 1、ORDER BY的索引优化。如果一个SQL语句形如：

SELECT [column1],[column2],…. FROM [TABLE] ORDER BY [sort];  
在[sort]这个栏位上建立索引就可以实现利用索引进行order by 优化。

2、WHERE + ORDER BY的索引优化，形如：  
SELECT [column1],[column2],…. FROM [TABLE] WHERE [columnX] = [value] ORDER BY [sort];  
建立一个联合索引(columnX,sort)来实现order by 优化。

注意：如果columnX对应多个值，如下面语句就无法利用索引来实现order by的优化  
SELECT [column1],[column2],…. FROM [TABLE] WHERE [columnX] IN ([value1],[value2],…) ORDER BY[sort];

3、WHERE+ 多个字段ORDER BY  
SELECT \* FROM [table] WHERE uid=1 ORDER x,y LIMIT 0,10;  
建立索引(uid,x,y)实现order by的优化,比建立(x,y,uid)索引效果要好得多。

MySQL Order By不能使用索引来优化排序的情况  
\* 对不同的索引键做 ORDER BY ：(key1,key2分别建立索引)  
SELECT \* FROM t1 ORDER BY key1, key2;

\* 在非连续的索引键部分上做 ORDER BY：(key\_part1,key\_part2建立联合索引;key2建立索引)  
SELECT \* FROM t1 WHERE key2=constant ORDER BY key\_part2;

## Group 分组 ，分组列作为key+有序列表

使用分组key作为 key\_col ：： 有序列表

## Top n 使用有序列表

Top 1

Set<String> maxOrdby= jedis. zrevrange("kaijiangTime\_caizhonID\_" + id,0,1);

Iterator<String> it = maxOrdby.iterator();

if(it.hasNext())

{

String str=it.next();

li.add(JSONObject.parse(str));

}

## 聚合函数sum avg等reduce运算

## count等有api实现

2 匹配查询

利用hash表的hget或hmget可以实现dept='IT'或者dept in ('IT', 'QA')这种单值或多值的完全匹配查询。拿到id列表后，再去查询key-value获得到对象。

3 范围查询

因为我们将age保存成zSet的score，value是id，所以可以利用zSet的zrangeByScore方法获得score在某一区间范围内的value值。

4 模糊查询

Redis 2.8.9后zSet加入了一个非常有用的方法zrangeByLex，我们将score都保存为0，value是姓名:id的格式，利用zrangeByLex可以获得字母在某一区间内的value值。例如，zrangeByLex name [A, (F，可以查询出Allen, Aaron, Carter。

5 分页查询

同时，zrangeByLex还支持分页查询，语法类似limit start, offset。

6 局限性

上述举例说明了几种常见查询在Redis的实现方式，但是Redis毕竟只是key-value存储，所以有很多局限性。例如，1）无法实现多条件组合的查询，例如age>25 AND name like 'A%'，硬要实现的话需要多条命令并计算并集或交集。2）模糊查询中文比较费劲：

## 行的保存 kv字符串

Key-value是为了保存id和整个对象，