SQL优化总结

2023年8月16日 16:56

SQL语句的执行顺序
先执行FROM选择表，接着进行表的链接，然后进行where条件过滤，group by分组，接着执行聚合函数、执行分组后的过滤，再然后进行select选择字段并去重，最后执行排序和分页。

(8) SELECT(9) DISTINCT column,...
选择字段 、去重
(6) AGG_FUNC(column or expression),...
聚合函数
(1) FROM [left_table]
选择表
(3) <join_type> JOIN <right_table>
链接
(2) ON <join_condition>
链接条件
(4) WHERE <where_condition>
条件过滤
(5) GROUP BY <group_by_list>
分组
(7) HAVING <having_condition>
分组过滤
(10) ORDER BY <order_by_list>
排序
(11) LIMIT count OFFSET count;
分页

插入操作:
```
INSERT INTO `table_name` (`column1`,`column2`,`column3`,...) VALUES (value1,value2,value3);
```
更新操作:
```
UPDATE `table_name` SET `column1` = value1,`column2` = value2,...
WHERE `some_column` = some_value;
```
删除操作:
```
DELETE FROM `table_name` WHERE `some_column` = `some_column`;
```

修改表结构来创建索引:
```
-- 唯一索引
alter table examination_info
add unique index uniq_idx_exam_id(exam_id);

-- 全文索引
alter table examination_info
```

```
add fulltext idex full_idx_tag(tag);

-- 普通索引
alter table examination_info
add index idx_duration(duration);
```

直接创建索引:

```
--普通索引
create index idx_duration on examination_info(duration);

--唯一索引
create unique index uniq_idx_exam_id on examination_info(exam_id);

--全文索引
create fulltext index full_idx_tag on examination_info(tag);
```

常见优化方式:

1、查询时尽量使用selct去查询具体字段，而不是select *;

2、用union all代替union进行连接，这个会查询出重复数据，确保不会出现数据丢失

3、小表驱动大表，使用in关键字，左边为大表，右边为小表；使用exists，左边小表

4、批量操作，一次数据库请求插入多条数据insert into order(id,code,user_id) values(123,'001',100),(124,'002',100),(125,'003',101);

5、多用limit进行筛选，加快查询速率

6、减少in关键字中的值

7、增量查询，按id和时间查询出一批数据后，保存最大的id和时间下次接着进行查询提升效率

8、高效分页，先找到上次分页最大的id，然后利用id上的索引查询进行分页

9、用连接查询代替子查询

10、join表不宜过多，最多不超过3个

11、使用left join关联查询时，左边要用小表，右边可以用大表。如果能用inner join的地方，尽量少用left join

12、控制索引数量，单表的索引数量应该尽量控制在5个以内，并且单个索引中的字段数不超过5个。

13、选择合适的字段类型，节约存储空间

14、提升group by的效率，进行比较耗时的SQL操作前，尽量缩小数据范围，提升SQL整体的性能

15、索引优化，尽可能地提升索引查询的效率