Page 1

MySQL中的SQL调优设计SQL调优有哪些基本原则? 导致SQL查询效率比较低的原因,主要包括数据量,数据访问量,数据 - 掘金 https://juejin.cn/post/7358109207994712075

TRAE 2.0 SOLO 出道,一键贯通从灵感火花到上线部署的全程协作

立即体验≺→



这 稀土掘金

首页

探索稀土掘金

创作者中心

Q

登录|注册



MySQL中的SQL调优设计

user8288612657632 2024-04-16 ◎ 771 ⑤ 阅读10分钟

关注











TRAE 2.0 SOLO 出道,一键贯通从灵感火花到上线部署的全程协作

SQL调优有哪些基本原则?

导致SQL查询效率比较低的原因,主要包括数据量,数据访问量,数据计算,SQL语句的设计几个层面

- 减少数据量(表中数据太多可以分表,例如双11是一个小时一张订单表)
- 减少数据访问量(将全表扫描可以调整为基于索引去查询)
- 减少数据计算操作(将数据库中的计算拿到程序内存中计算)

SQL优化的基本逻辑是怎样的?

- 良好SQL编码的习惯(熟悉SQL编码规范、例如关键字大写,避免使用select *)
- 优秀SQL的编写逻辑(例如表关联时小表驱动大表)
- 定位需要优化的慢SQL语句(耗时多长时间的SQL是慢SQL)
- 调整优化策略并进行测试。(SQL结构上的调整、索引应用)
- 按业务进行分库分表。(分表可以在应用逻辑中减少单表数据量)

有哪些优秀SQL编写案例?

- 查询时尽量避免使用select *;
- 尽量避免在where子句中使用or作为查询条件。
- where 条件中尽量不要出现与null值的比较。
- 避免在查询中存在隐式转换。(... where id='1')
- 避免在where子句中使用!=或者<>操作符。
- 使用like查询条件时应尽量避免前缀使用"%"。
- 执行查询时尽量采用最左匹配原则。(where first_name='A' and email='A@t.com')
- 避免在查询条件中使用一些内置的SQL函数。(MySQL8中现在可以基于SQL函数 添加索引了)
- 假如in表达式后面的数据太多,尽量避免使用in作为查询条件。(where id in (1,2,3,4,5,....))
- 当有多个查询条件、分组条件、排序条件时,尽量使用联合索引(组合索引)
- 表连接时优先使用内连接(inner join),使用小表驱动大表。
- 进行表关联的字段尽量使用相同的编码(不能一个字段utf8,一个字段utf8mb4)

- 表设计时字段类型能用简单数据类型不用复杂类型。(例如能用 int 不用 varchar)
- 清空表中数据可优先使用truncate.(truncate删除数据时不记录日志)
- 插入多条数据时可考虑使用批量插入。(insert into xxx values (...),(...),(...))

如何基于慢SQL日志查询慢SQL?

线上环境,我们对SQL进行调优,首先要发现执行慢的SQL,然后再对SQL进行分析和优化。如何找到执行慢的SQL呢,可以通过慢查询日志进行分析,具体可以参考如下步骤:

- 1. 开启慢查询日志(一般默认是关闭状态)
- 2. 设置慢查询阀值(响应速度是多长时间被定为是慢查询)
- 3. 确定慢查询日志路径(日志文件在哪里)
- 4. 确定慢查询日志的文件名(具体日志文件是哪个), 然后对文件内容进行分析。
- 5. 打开慢查询日志文件,检查慢SQL。

实操演示:

查看慢查询日志的打开状态?

```
▼ sql

1 show variables like '%slow_query_log%';
```

默认环境下,MySQL5.7默认慢查询日志状态是关闭的(OFF)。

如何开启慢查询日志?

```
▼ sql

1 set global slow_query_log=0N --MySQL5.7, MySQL8.0
2 set slow_query_log=0N --10.5.17-MariaDB
```

查看默认慢查询阈值 (默认为10秒,假如一个SQL查询耗时超过了10秒钟,就会认为是慢SQL)

```
▼ sql

1 show variables like '%long_query_time%';
```

如何设置慢查询的阈值?

MySQL5.7设置慢查询时间阀值(响应时间是多长时间是慢查询)

```
▼ sql

1 mysql> set long_query_time = 1;
2 Query OK, 0 rows affected (0.00 sec)
```

• 如何知道慢查询日志路径?

慢查询日志的路径默认是 MySQL 的数据目录

```
▼ sql

1 mysql> show global variables like 'datadir';
2
3 +-----+
4 | Variable_name | Value |
5 +-----+
6 | datadir | /mysql/data/ |
7 +-----+
8 1 row in set (0.00 sec)
```

• 如何知道慢查询日志的文件名?

执行一个耗时SQL,然后查看慢SQL日志,例如

```
▼ sql

1 select * from employees where salary between 100 and 30000
2 union
3 select * from employees where salary between 100 and 30000
```

打开日志文件,可以对日志文件中的内容进行分析,常用选项说明:

Time: 慢查询发生的时间

User@Host: 客户端用户和IP

Query_time: 查询时间

Lock_time: 等待表锁的时间

Rows_sent: 语句返回的行数

Rows_examined: 语句执行期间从存储引擎读取的行数

如何对慢SQL查询进行分析?

工欲善其事,必先利其器,分析慢查询可以通过 explain、show profile 等工具来实现。

执行计划(Explain)是什么?

执行计划是mysql优化器对SQL进行默认调优后,给出的一种执行方案,这个方案我们可以通过explain 这个指令进行查询。例如,对select语句进行分析,并输出select执行时的详细信息,开发人员可以 基于这些信息进行有针对性的优化,例如:

```
▲ 体验AI代码助手 ☑代码解读 复制代码
    sql
   mysql> explain select * from employees where employee_id<100 \G;
                row
3 id: 1
4 select_type: SIMPLE
5 table: employees
6 partitions: NULL
   type: range
8 possible_keys: PRIMARY
   key: PRIMARY
10 key_len: 4
11 ref: NULL
12 rows: 1
13 filtered: 100.00
14 Extra: Using where
15 1 row in set, 1 warning (0.00 sec)
```

分析执行计划的目的是什么?

分析执行计划的目的就为了了解SQL的执行逻辑,例如: - 检查关联查询、嵌套查询的执行顺序. - 查询操作的具体类型 - 哪些索引可能会命中以及实际命中的索引有哪些 - 每张表可能有多少条记录参与到了查询中

说说执行计划中几个常见的字段?

id

select 的序列号,有几个select 就有几个 id, id 的顺序是按 select 出现的顺序增长的。 即:id 越大语句执行的优先级越高,id相同则从上往下依次执行,id 为NULL最后执行。

```
▼ sql

A 体验AI代码助手 ☑代码解读 复制代码

explain

select last_name,salary

from employees

where employee_id=(

select manager_id

from employees

where employees

where employees

where employee(id=206);
```

当id值相同时,优先级从上到下,例如:

• select_type表示的查询类型有哪些?

1. SIMPLE: 表示查询语句不包含子查询或 union

2. PRIMARY:表示此查询是最外层的查询

3. UNION:表示此查询是 union 的第二个或后续的查询

4. UNION RESULT: union 的结果

- 5. DEPENDENT UNION:子查询中的UNION操作,UNION后的所有select都是DEPENDENT UNION。
- 6. SUBQUERY: SELECT 子查询语句
- 7. DEPENDENT SUBQUERY:子查询中的第一个SELECT,SELECT 子查询语句依赖外层查询。
- 8. DERIVED: from 子句后的相对比较复杂的子查询(相当于一个临时表),当看到derivedN时,这里N表时查询id

案例分析:

select_type为SIMPLE (表示查询语句不包含子查询或 union)

select_type为PRIMARY、SUBQUERY (PRIMARY表示最外层查询,SUBQUERY表示嵌套查询)

```
▼ sql

A 体验AI代码助手 近代码解读 复制代码

1 explain
2 select last_name, salary
3 from employees
4 where employee_id=(
5 select manager_id
6 from employees
7 where employee_id=206);
```

select_type为 UNION(union操作)、UNION RESULT(union的结果)

```
▼ sql

A 体验AI代码助手 位代码解读 复制代码

explain

select first_name,hire_date,salary

from employees

where job_id='AD_VP'

union

select first_name,hire_date,salary

from employees

where salary>15000;
```

select_type为DEPENDENT UNION (子查询中的UNION操作,UNION后的所有select都是DEPENDENT UNION。)

```
▲ 体验AI代码助手 ☑代码解读 复制代码
    sql
   explain
2 select *
   from employees e3
   where first_name in (
            select first_name
            from employees e1
            where job_id = 'AD_VP'
            union
8
            select first_name
            from employees e2
10
            where salary > 15000
11
12
```

select_type 为 DEPENDENT SUBQUERY

```
▼ sql

A 体验AI代码助手 应代码解读 复制代码

explain

select employee_id,first_name,salary

from employees e1

where salary=(

select max(salary)

from employees e2

where e1.department_id=e2.department_id);
```

说明,一般出现DEPENDENT SUBQUERY时,SQL的执行效率都会比较低,可以调整为多表查询,例如:

```
▼ sql

1 explain
2 select e2.department_id,e2.employee_id,e2.first_name,e2.salary
3 from (
4 select department_id,max(salary) max_salary
5 from employees
6 group by department_id) e1 join employees e2
7 on e1.department_id=e2.department_id
8 where e1.max_salary=e2.salary
```

select_type为DERIVED(这里一般表示from后面的一个衍生表-临时表)

```
▼ sql

A 体验AI代码助手 应代码解读 复制代码

explain

select min(avg_salary)

from (

select avg(salary) avg_salary

from employees

group by department_id) emp;
```

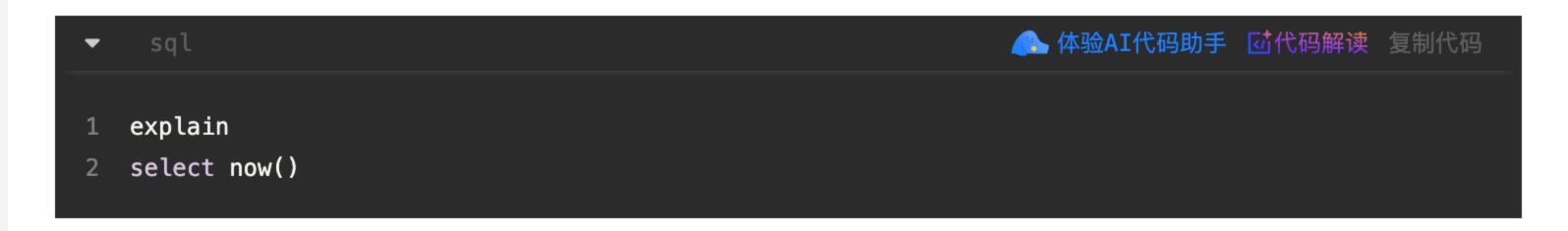
• type表示查询数据的方式。(重点)

type是一个比较重要的一个属性,通过它可以判断出查询是全表扫描还是基于索引的部分扫描。 常用属性值如下,从上至下效率依次增强。调优时,建议type类型至少要为range,才能提高查询效率。

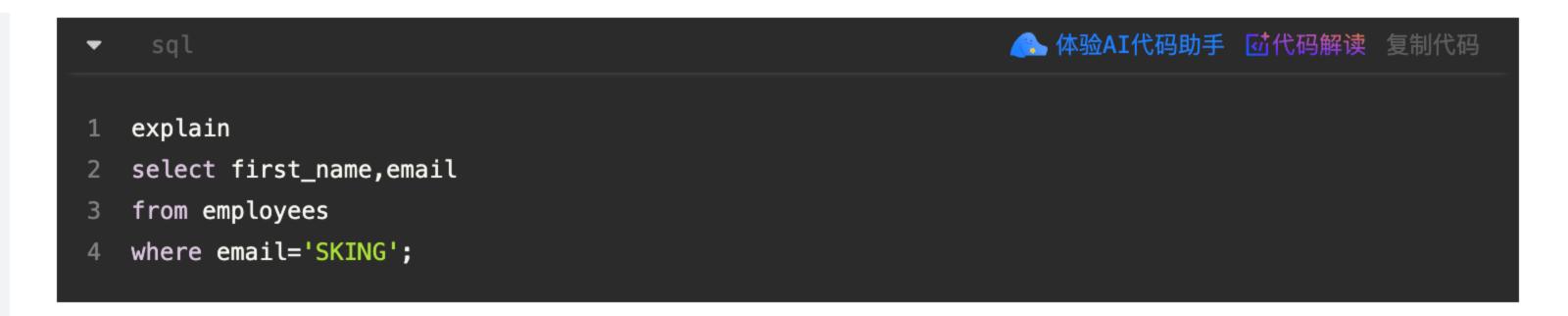
- 1. ALL: 表示全表扫描, 性能最差。(数据量小时无所谓)
- 2. index:表示基于索引的全表扫描,先扫描索引再扫描全表数据。
- 3. range: 表示使用索引范围查询。使用 >、>=、<、<=、in 等等。
- 4. index_merge: 表示查询中使用到了多个索引,然后进行了索引合并
- 5. ref: 表示使用非唯一索引进行单值查询。
- 6. eq_ref: 一般情况下出现在多表 join 查询,表示前面表的每一个记录,都只能匹配后面表的一行结果。
- 7. const:表示使用主键或唯一索引做等值查询,常量查询。(效率非常高)
- 8. NULL:表示不用访问表,也没有索引,速度最快。(了解,例如select version())

案例分析:

type为null



type 为const (基于主键或唯一键执行的查询)



type 为eq_ref (多表join, 前面表的每一行记录, 只能匹配后面表的一行记录)

```
▼ sql

1 explain
2 select d.department_id,d.department_name,e.first_name
3 from departments d join employees e on d.manager_id = e.employee_id;
```

type为 ref (使用非唯一索引进行的等值查询)

```
▼ sql

↑ create index index_first_name on employees(first_name);
2 explain
3 select *
4 from employees
5 where first_name='Steven';

▼ sql

↑ create index index_salary on employees (salary);
2 explain
3 select salary, first_name from employees where salary=17000;
```

type 为 index_merge (索引合并,同时应用两个索引)

```
▼ sql

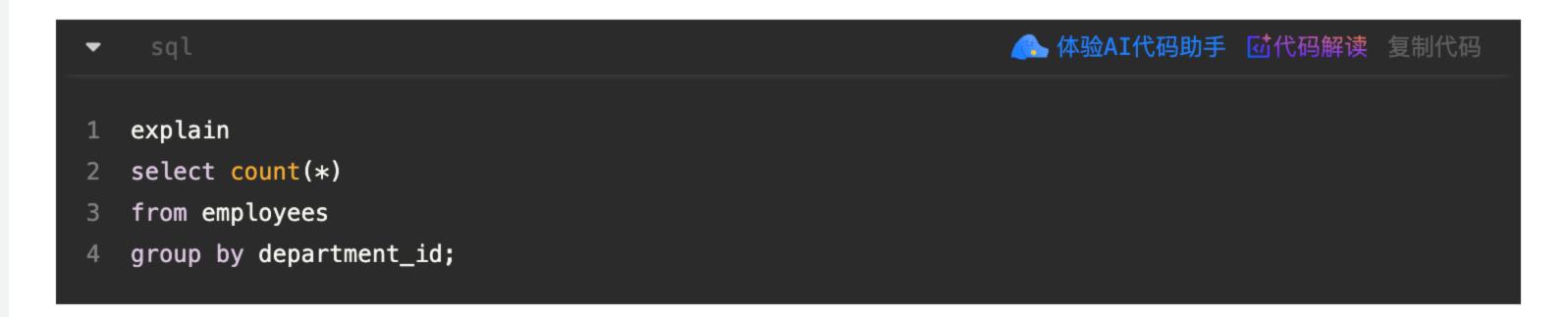
1 create index index_salary on employees(salary);
2 explain
3 select first_name,hire_date,salary
4 from employees
5 where job_id='AD_VP' or salary>15000;
```

type 为 range (这里的range表示一个范围查询)

```
▼ sql

↑ create index index_salary on employees (salary);
2 explain
3 select first_name,salary
4 from employees
5 where salary between 10000 and 30000;
```

type 为index (基于索引的全表扫描)



type 为 all (表示全表扫描)



• Extra 中值 的含义是什么? Extra 表示很多额外的信息,各种操作会在 Extra 提示相关信息,常见几种如下:

"Using where"表示查询需要通过where条件查询数据(可能没有用到索引,也可能一部分用到了索引)。

Using index 表示查询需要通过索引,索引就可以满足所需的数据(不需要再回表查询-基于普通索引找到主键,然后再基于主键查找对应纪录,当前查询中应用了覆盖索引-select列表中的值都是索引值)。

```
▼ sql

↑ create index index_hire_date_salary on employees(hire_date,salary);
2 explain
3 select employee_id,hire_date,salary
4 from hr.employees
5 where hire_date>'2000-03-06' and salary>10000;
```

Using index condition 表示查询的记录,在索引中没有完全覆盖(可能要基于where或二级索引对应的主键再次查询-回表查询)。

```
▼ sql

↑ create index index_hire_date_salary on employees (hire_date,salary)

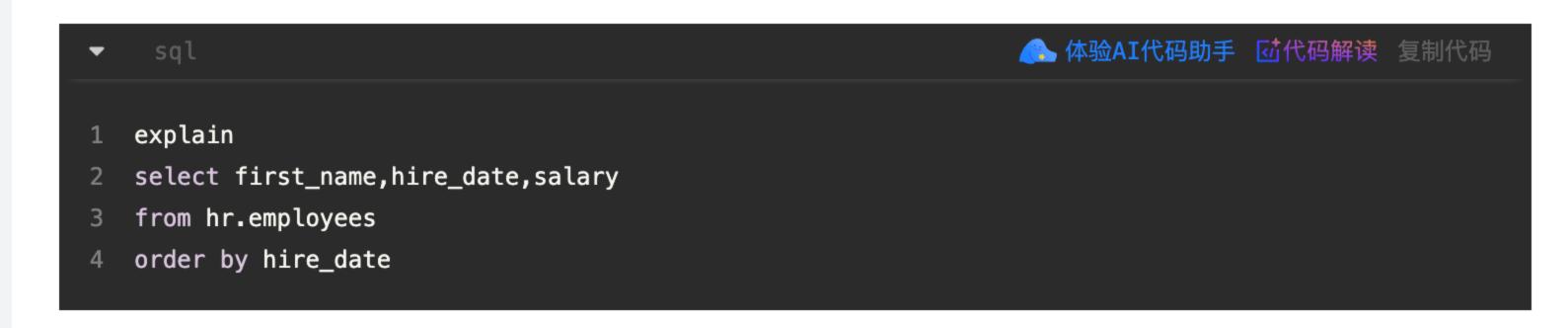
2 explain

3 select employee_id,hire_date,salary,commission_pct

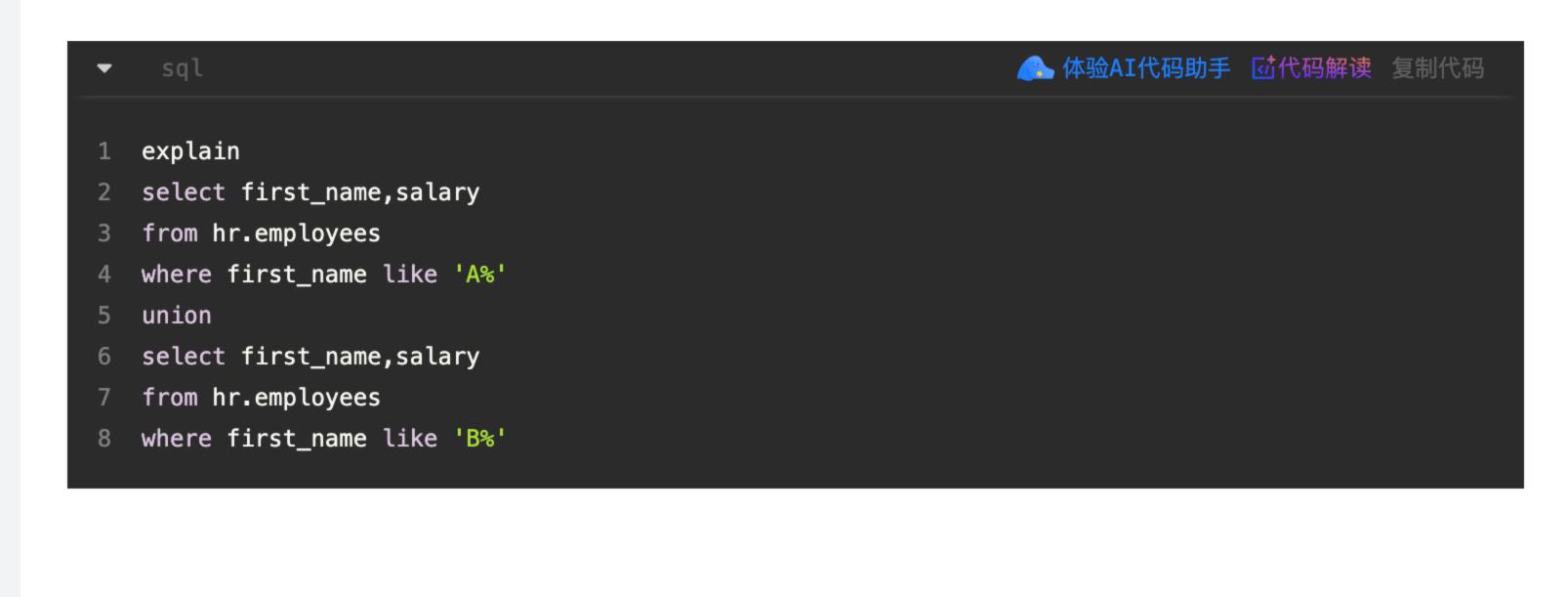
4 from hr.employees

5 where hire_date>'2000-03-06' and salary>10000;
```

Using filesort 表示查询出来的结果需要额外排序,数据量小在内存,大的话在磁盘,因此有 Using filesort 建议优化。



Using temprorary 表示查询使用到了临时表,一般出现于去重、分组等操作(这里一般也需要优化)。



标签: 后端 数据库





Page 11

MySQL中的SQL调优设计SQL调优有哪些基本原则? 导致SQL查询效率比较低的原因,主要包括数据量,数据访问量,数据 - 掘金 https://juejin.cn/post/7358109207994712075

评论 0



登录 / 注册

即可发布评论!



暂无评论数据

相关推荐

MySQL中的SQL查询性能调优

295阅读 · 2点赞

MySQL 性能调优与 SQL 调优指南

300阅读·4点赞

全解MySQL终章:这份爆肝30W字的数据库宝典赠与有缘的你!

48k阅读 · 494点赞

MySQL索引和SQL调优

28k阅读 · 617点赞

年薪近百万架构师纯手写的MySQL笔记,看完感觉之前读的都是渣渣!

505阅读·0点赞

精选内容

centos如何使用高版本gcc

苏三的开发日记:31阅读:0点赞

Java【问题 07】SSH不同版本使用jsch问题处理(7.4升级9.7及欧拉原生8.8)

yuanzhengme · 28阅读 · 1点赞

Redis 分布式锁深度解析: setnx 命令的核心作用与实现

Code季风·37阅读·0点赞

OpenSSH【安装 02】离线升级异常问题解决、无法升级时的失败恢复

yuanzhengme · 19阅读 · 0点赞

java 面试八股这一篇就够之java集合篇

小厂永远得不到... · 94阅读 · 3点赞

为你推荐

性能优化篇: SQL数据库查表速度优化

为了WLB努力 │ 1年前 │ ◎ 1.0k 1 15 ፡ 评论

后端 SQL MySQL

后端 笔记

尚硅谷MySQL高级学习笔记 -- 4.查询截取分析

exodus3 4年前 🔘 310 🜓 点赞 💬 评论

MySql优化-上|8月更文挑战

MySQL调优问题与解决方案

MySQL查询语句优化的十个小技巧!

麒麟IT | 4年前 | ◎ 787 | △ 4 | ፡□ 评论 | Java | 后端

MySQL高级篇 - 性能优化

slowlybutsurely 4年前 ◎ 1.1k ⑥ 8 评论

MySQL的常用优化方案

MySQL的常用优化方案

KwingTaai 2年前 ◎ 1.6k ⑥ 11 ※ 评论

关于 MySQL 慢查询优化的一些思路与知识点

Page 13

MySQL中的SQL调优设计SQL调优有哪些基本原则? 导致SQL查询效率比较低的原因,主要包括数据量,数据访问量,数据 - 掘金 https://juejin.cn/post/7358109207994712075

