



拒绝全表扫描！3个提升MySQL深度分页技巧！

四七佃 2025-02-17 180 阅读5分钟 专栏：MySQL 开发者宝典

「前言」

你有没有遇到过这种情况？在电商平台翻看自己的历史订单，前几页加载飞快，但翻到几十页之后，页面就像被按了慢放键。这背后隐藏的正是[MySQL深度分页](#)的典型问题——数据越往后查，速度越让人抓狂。今天我们就来拆解这个看似简单实则暗藏玄机的问题。



「一、案例分析：电商订单查询的分页问题」

假设某电商平台的订单表存储了1000万条记录。

创建orders表，SQL语句如下：

```
sql 代码解读 复制代码

1 CREATE TABLE `orders` (
2   `id` int NOT NULL AUTO_INCREMENT, -- id自增
3   `user_id` int DEFAULT NULL,
4   `amount` decimal(10,2) DEFAULT NULL,
5   `create_time` datetime DEFAULT CURRENT_TIMESTAMP, -- 创建时间默认为当前时间
6   PRIMARY KEY (`id`),
7   KEY `idx_create_time` (`create_time`) -- 创建时间设置为普通索引
8 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

生成1000w条数据，如下图：

提示：

- id 默认自增，create_time 是默认当前时间，这里只用给“user_id”和“amount”设置属性。
- 生成数据需要几分钟的时间。可以先删除create_time索引，数据插入完之后再新增索引，这样生成速度会快一点。



设置create_time为普通索引，SQL语句如下：

```
sql 代码解读 复制代码

1 -- 如果已经设置可以忽略
2 ALTER TABLE `orders`
3 ADD INDEX `idx_create_time` (`create_time` ASC) USING BTREE;
```

查询第一页的20条数据，作为参考，方便直观的对比深度查询的时间，SQL语句如下：

```
sql 代码解读 复制代码

1 SELECT * FROM orders
2 ORDER BY create_time DESC
3 LIMIT 0, 20;
```

第一页查询用时 [120ms]，如下图：

查询	消息	查询时间
SELECT * FROM orders ORDER BY create_time DESC LIMIT 0, 20	OK	

未做深度分页处理时，当用户查询第1000页的订单（每页20条），常见的分页写法如下：

```
sql 代码解读 复制代码

1 SELECT * FROM orders
2 ORDER BY create_time DESC
3 LIMIT 19980, 20; -- (1000-1)*20 = 19980
```

未特殊处理分页时，用时 [320ms]，如下图：

查询	消息	查询时间
SELECT * FROM orders ORDER BY create_time DESC LIMIT 19980, 20	OK	0.032s

执行流程解析：

- 通过索引 idx_create_time 读取19980+20条数据。
- 在内存中进行排序（即使已经索引）。
- 丢弃前19980条，返回最后20条。

随着页码增加，需要处理的数据量会线性增长。当offset达到10w时，查询耗时会显著增加，达到100w时，甚至需要数秒。

四七佃

Java全栈

榜上有名

52

文章

14k

阅读

41

粉丝

关注

私信

目录 收起

前言

一、案例分析：电商订单查询的分页问题

二、三种实用的深度分页优化方案

方案1：子查询接力（ID接力赛）

方案2：游标分页（时光机穿梭）

方案3：索引覆盖（轻装上阵）

三、深度分页优化方案选择指南

四、B+树原理与优化思路

结语：跳出分页思维定式

- 相关推荐
- 如何将传统线程池改造为虚拟线程，实...
139阅读 · 2点赞

RDS用多了，你还知道MySQL主从复制...
93阅读 · 0点赞

SQL-更进一步
69阅读 · 0点赞

MySQL 的JSON类型违反第一范式吗？
181阅读 · 2点赞

电商秒杀场景下的布隆过滤器与布谷鸟...
92阅读 · 1点赞

- 精选内容
2. Docker 在 Linux 当中安装(超详细附...
RainbowSea · 69阅读 · 1点赞

Spring Cloud Gateway新特性及高级开...
星辰聊技术 · 214阅读 · 1点赞

Python并发秘籍：如何一天内让你的程...
Y11_推特同名 · 48阅读 · 0点赞

1. Docker 的简介概述
RainbowSea · 33阅读 · 0点赞

JDK21 虚拟线程彻底杀死响应式编程
半夏之沫 · 783阅读 · 7点赞

找对属于你的技术圈子

回复「进群」加入官方微信群





「二、三种实用的深度分页优化方案」

» 方案1：子查询接力（ID接力赛）

适用场景：主键ID有序且递增的情况。

sql

代码解读复制代码

```
1 SELECT * FROM orders
2 WHERE id >= (
3     SELECT id FROM orders
4     ORDER BY create_time DESC
5     LIMIT 19980, 1
6 )
7 ORDER BY create_time DESC
8 LIMIT 20;
```

效果展示：(用时 190ms)

查询	消息	查询时间
SELECT * FROM orders WHERE id >= (SELECT id FROM orders ORDER BY create_time DESC LIMIT 19980,...	OK	0.019s
SELECT * FROM orders WHERE id >= (SELECT id FROM orders ORDER BY create_time DESC LIMIT 19980, 1) ORDER BY create_time DESC LIMIT 20		

掘金技术社区 @ 四七侧

优化原理：

- 子查询：快速定位分页起始ID
- 主查询：通过ID范围进行高效扫描

这种方式避免了从头遍历大量数据，大大提升了查询效率。

» 方案2：游标分页（时光机穿梭）

适用场景：支持连续分页（如无限滚动）。

sql

代码解读复制代码

```
1 -- 第一页
2 SELECT * FROM orders
3 ORDER BY create_time DESC, id DESC
4 LIMIT 20;
5
6 -- 后续页 (记住上一页最后一条的create_time和id)
7 SELECT * FROM orders
8 WHERE create_time < '2025-02-15 00:04:45'
9 OR (create_time = '2025-02-15 00:04:45' AND id < 9999981)
10 ORDER BY create_time DESC, id DESC
11 LIMIT 20;
```

效果展示：(用时 210ms)

查询	消息	查询时间
SELECT * FROM orders WHERE create_time < '2025-02-15 00:04:45' OR (create_time = '2025-02-15 00:04:4...	OK	0.021s
SELECT * FROM orders WHERE create_time < '2025-02-15 00:04:45' OR (create_time = '2025-02-15 00:04:45' AND id < 9999981) ORDER BY create_time DESC, id DESC LIMIT 20		

掘金技术社区 @ 四七侧

优化原理：

- 使用 create_time + id 组合排序，避免重复数据。
- 每次查询只处理当前页数据，避免了全表扫描。

» 方案3：索引覆盖（轻装上阵）

适用场景：当查询字段与联合索引中的字段完全匹配，并且查询只需要索引中的数据时，使用覆盖索引可以避免回表操作，显著提高查询效率。

sql

代码解读复制代码

```
1 SELECT * FROM orders
2 INNER JOIN (
3     SELECT id FROM orders
4     ORDER BY create_time DESC
5     LIMIT 19980, 20
6 ) AS tmp USING(id);
```

效果展示：(用时 180ms)

查询	消息	查询时间
SELECT * FROM orders INNER JOIN (SELECT id FROM orders ORDER BY create_time DESC LIMIT 19980, 2...	OK	0.018s
SELECT * FROM orders INNER JOIN (SELECT id FROM orders ORDER BY create_time DESC LIMIT 19980, 20) AS tmp USING(id)		

掘金技术社区 @ 四七侧

优化原理：

- 子查询只读取ID列，利用索引覆盖。
- 主查询通过主键快速定位数据。

这种方法减少了需要返回的字段数量，提升了性能。

「三、深度分页优化方案选择指南」

方案	响应时间	可跳页	适用场景
传统分页	慢	支持	小数据量
子查询接力	快	支持	ID连续
游标分页	最快	不支持	无限滚动
索引覆盖	快	支持	联合索引存在

架构建议：

- 前端采用“加载更多”按钮，代替页码跳转。
- 结合业务需求添加时间范围等筛选条件。
- 对历史数据进行归档处理（如将3个月前的订单转存到HBase）。



「四、B+树原理与优化思路」

MySQL的InnoDB存储引擎使用B+树存储索引数据。在执行类似 `LIMIT 100000, 20` 的查询时：

- 从根节点逐层定位到最左叶子节点。
- 向右遍历10万条记录（即便不需要数据）。
- 导致大量的随机I/O和CPU计算。

优化方案通过直接定位数据起始位置，显著提高查询效率，将时间复杂度从O(N)降到O(logN + M)。



「结语：跳出分页思维定式」

在面对千万级数据时，[分页](#)本身可能并不是最佳解决方案。建议根据实际业务需求，考虑以下替代方案：

- 搜索引擎**：如Elasticsearch的 `search_after` 参数，优化分页性能。
- 列式存储**：如ClickHouse的分区查询，快速处理大数据量。
- 预计算**：将热门查询结果缓存到Redis，提高响应速度。

技术选型时，记住要多问一句：“用户真的需要精确分页吗？”也许一个智能的搜索框，比精确的页码跳转更能提升用户体验。

标签：

后端

MySQL

话题：

每天一个知识点

本文收录于以下专栏



MySQL 开发者宝典

专栏目录

MySQL 深度实践、从原理到优化、实战指南

5 订阅 · 10 篇文章

订阅

上一篇

为什么90%业务选择InnoDB? MySQL存储...

评论 0



登录 / 注册

即可发布评论!



暂无评论数据

为你推荐

MySQL优化 -> 深分页

MAODO | 3月前 | 👁 330 | 👍 3 | 💬 1

后端面试

分页拉取数据重复的几种解决思路

狗肉是只猫 | 3年前 | 👁 5.6k | 👍 11 | 💬 评论

Java

服务:使用django构建:处理分页和查询

楽码 | 1年前 | 👁 3.2k | 👍 1 | 💬 评论

后端架构Python

limit与分页键

安妮的心动录 | 11月前 | 👁 750 | 👍 1 | 💬 评论

MySQL后端数据库

MySQL深度分页的问题及优化方案：千万级数据量如何快速分页

少侠露飞 | 3年前 | 👁 3.3k | 👍 14 | 💬 评论

后端

MySQL分页及优化

喵_2b98 | 6月前 | 👁 157 | 👍 2 | 💬 评论

数据库性能优化

Elasticsearch分页查询

AndyH | 3年前 | 👁 1.2k | 👍 6 | 💬 评论

Elasticse...后端

亿级数据量场景下，如何优化数据库分页查询方法

华为云开发者联盟 | 2年前 | 👁 2.6k | 👍 10 | 💬 评论

数据库MySQL

四选一，如何选择适合你的分页方案？

五阳 | 1年前 | 👁 2.5k | 👍 28 | 💬 11

后端架构数据库

数据库控制：如何优化数据库的性能？ 方向

楽码 | 1年前 | 👁 934 | 👍 2 | 💬 评论

后端架构掘金·日...

Elasticsearch 分页查询

狼爷 | 3年前 | 👁 13k | 👍 11 | 💬 1

Elastic...

SQL阶段性优化

是江迪呀 | 1年前 | 👁 315 | 👍 4 | 💬 1

后端SQL性能优化

数据库深分页介绍及优化方案 | 京东云技术团队

京东云开发者 | 1年前 | 👁 1.5k | 👍 16 | 💬 6

数据库

基于偏移量、游标分页的详解

6r0wn_Jay | 1年前 | 👁 783 | 👍 7 | 💬 1

后端Go数据库

索引深分页问题

lunarYoung | 8月前 | 👁 239 | 👍 点赞 | 💬 评论

后端