什么是索引下推？什么是索引覆盖？什么是回表？

原创 womubuji womubuji 2025年02月10日 09:26 上海

索引下推（Index Condition Pushdown，ICP）

定义

索引下推是 MySQL 5.6+ 引入的优化技术，核心思想是将 `WHERE` 子句中的过滤条件下推到存储引擎层处理，减少回表次数。
适用场景：复合索引中，非最左前缀的列过滤条件可以通过索引下推提前过滤数据。

案例

- **表结构**：

```
CREATE TABLE `user` (
`id` INT PRIMARY KEY,
`name` VARCHAR(20),
`age` INT,
`city` VARCHAR(20),
KEY `idx_age_city` (`age`, `city`)
);
```

- **查询**：

```
SELECT * FROM user
WHERE age > 18 AND city = 'Shanghai' AND name LIKE '%Alice%';
```

- **未使用 ICP**：

  1. 存储引擎通过索引 `idx_age_city` 找到 `age > 18` 的记录。
  2. 将所有符合条件的 `age` 记录回表到主键索引，获取完整数据行。
  3. 服务器层再过滤 `city = 'Shanghai'` 和 `name LIKE '%Alice%'`。
- **使用 ICP**：

  1. 存储引擎通过索引 `idx_age_city` 找到 `age > 18` 的记录。
  2. **直接利用索引中的 `city` 列过滤 `city = 'Shanghai'`**，减少回表次数。
  3. 仅将符合 `age` 和 `city` 条件的记录回表，服务器层再处理 `name` 条件。

索引覆盖（Covering Index）

定义

索引覆盖指查询所需的所有列都包含在索引中，无需回表。
核心优势：减少 I/O 操作，提升查询性能。

案例

- **表结构**：

```
CREATE TABLE `order` (
`order_id` INT PRIMARY KEY,
`user_id` INT,
`product` VARCHAR(50),
`price` DECIMAL(10,2),
KEY `idx_user_product` (`user_id`, `product`)
);
```

- **查询 1**（索引覆盖）：

```
SELECT user_id, product FROM order
```

```
WHERE user_id = 100;
```

- **索引生效**
  `idx_user_product` 包含 `user_id` 和 `product`，无需回表。

- **查询 2**（未覆盖索引）：

```
SELECT * FROM order
WHERE user_id = 100;
```

- **需回表**
  索引未包含 `price` 字段，需回主键索引获取完整数据。

回表（Bookmark Lookup）

定义

回表是通过普通索引查询到主键后，再回到聚簇索引（主键索引）中查找完整数据行的过程。
性能问题：回表会导致额外的磁盘 I/O，是查询性能瓶颈的常见原因。

案例

- **表结构**：

```
CREATE TABLE `employee` (
`id` INT PRIMARY KEY,
`name` VARCHAR(20),
`department` VARCHAR(20),
`salary` INT,
KEY `idx_department` (`department`)
);
```

- **查询**：

```
SELECT * FROM employee
WHERE department = 'Engineering';
```

- **执行过程**：

1. 通过索引 `idx_department` 找到 `department = 'Engineering'` 的所有主键 `id`。
2. 根据每个 `id` 回表到主键索引，获取完整的行数据（包含 `name`, `salary` 等字段）。

三者的关系与优化

综合优化案例

- **问题**：以下查询性能较差：

```
SELECT id, name, age FROM user
WHERE city = 'Beijing' AND age > 25;
```

- **优化步骤**：
  1. **创建覆盖索引**： ALTER TABLE user ADD INDEX idx_city_age_name (city, age, name);
     - 索引包含所有查询字段（ city ， age ， name ）和主键 id 。
  2.
  3. **利用索引下推**：
     - 存储引擎直接通过 idx_city_age_name 过滤 city = 'Beijing' 和 age > 25 。
     - 无需回表，直接返回 id ， name ， age 。

总结

- **索引下推**
  将过滤条件下推到存储引擎，减少回表数据量。
- **索引覆盖**
  通过设计索引包含查询字段，彻底避免回表。
- **回表**
  普通索引查询的额外 I/O 过程，需通过覆盖索引或减少查询字段优化。

womubuji

喜欢作者

MySQL · 目录

上一篇
千万级别的大表分页查询非常慢，该怎么处理？

下一篇
300 秒到3秒，如何将 MySQL 批量写入的耗时缩短 99%？