> 有道笔记地址： https://note.youdao.com/s/5yh3I3oO

关于《深入理解Mysql索引底层数据结构与算法》这节课的笔记补充
课程中联合索引案例使用的脚本

```
CREATE TABLE `employees` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(24) NOT NULL DEFAULT '' COMMENT '姓名',
  `age` int(11) NOT NULL DEFAULT '0' COMMENT '年龄',
  `position` varchar(20) NOT NULL DEFAULT '' COMMENT '职位',
  `hire_time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '入职时间',
  PRIMARY KEY (`id`),
  KEY `idx_name_age_position` (`name`,`age`,`position`) USING BTREE
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8 COMMENT='员工记录表';

INSERT INTO employees(name,age,position,hire_time) VALUES('LiLei',22,'manager',NOW());
INSERT INTO employees(name,age,position,hire_time) VALUES('HanMeimei', 23,'dev',NOW());
INSERT INTO employees(name,age,position,hire_time) VALUES('Lucy',23,'dev',NOW());


EXPLAIN SELECT * FROM employees WHERE name = 'Bill' and age = 31;
EXPLAIN SELECT * FROM employees WHERE age = 30 AND position = 'dev';
EXPLAIN SELECT * FROM employees WHERE position = 'manager';
```

关于最左前缀的补充

MySQL一定是遵循最左前缀匹配的，这句话在mysql8以前是正确的，没有任何毛病。但是在MySQL 8.0中，就不一定了。

索引跳跃扫描（Index Skip Scan）

参考：https://dev.mysql.com/doc/refman/8.0/en/range-optimization.html#range-access-skip-scan
官网示例

```
CREATE TABLE t1 (f1 INT NOT NULL, f2 INT NOT NULL, PRIMARY KEY(f1, f2));
INSERT INTO t1 VALUES
(1,1), (1,2), (1,3), (1,4), (1,5),
(2,1), (2,2), (2,3), (2,4), (2,5);
```

```
INSERT INTO t1 SELECT f1, f2 + 5 FROM t1;
INSERT INTO t1 SELECT f1, f2 + 10 FROM t1;
INSERT INTO t1 SELECT f1, f2 + 20 FROM t1;
INSERT INTO t1 SELECT f1, f2 + 40 FROM t1;
ANALYZE TABLE t1;

EXPLAIN SELECT f1, f2 FROM t1 WHERE f2 > 40;
```

```
EXPLAIN SELECT f1, f2 FROM t1 WHERE f2 > 40;
```

信息 | 结果 1 | 剖析 | 状态

id	select_type	table	partitions	type	possible_key	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	t1	(Null)	range	PRIMARY	PRIMARY	8	(Null)	53	100.00	Using where; Using index for skip scan

虽然我们的SQL中，没有遵循最左前缀原则，只使用了f2作为查询条件，但是经过MySQL 8.0的优化以后，还是通过索引跳跃扫描的方式用到了索引了。

索引跳跃扫描优化原理

mysql8.013后通过优化器帮我们加了联合索引，SQL执行过程如下：

1. 获取 f1 字段第一个唯一值，也就是 f1 = 1
2. 构造 f1 = 1 and f2 > 40，进行范围查询
3. 获取 f1 字段第二个唯一值，也就是 f1 = 2
4. 构造 f1 = 2 and f2 > 40，进行范围查询

```
SELECT f1, f2 FROM t1 WHERE f2 > 40;

执行的最终SQL:
SELECT f1, f2 FROM t1 WHERE f1 =1 and f2 > 40
UNION
SELECT f1, f2 FROM t1 WHERE f1 =2 and f2 > 40;
```

所以对于对于f1值很少，区分度不高的情况索引跳跃扫描会快一些；反之查询效率慢些。

我们不能依赖这个优化，建立索引的时候，还是优先把区分度高的，查询频繁的字段放到联合索引的左边。

限制条件

- 查询必须只能依赖一张表，不能多表JOIN。
- 查询中不能使用 GROUP BY 或 DISTINCT 语句。
- 查询的字段必须是索引中的列。

- 组合索引形式：([A_1, …, A_k,] B_1, …, B_m, C [, D_1, …, D_n])，A，D 可以为空，但是B ,C 不能为空。