面经预热Day15（数据库专题）

1、索引有哪些种类

从数据结构维度进行分类:

- B+树索引:所有数据存储在叶子节点，复杂度为O(logn)，适合范围查询。
- 哈希索引:适合等值查询，检索效率高，一次到位
- 全文索引: MyISAM 和 InnoDB 中都支持使用全文索引，一般在文本类型char，text，varcha类型上创建
- R-Tree 索引:用来对 GIS 数据类型创建 SPATIAL 索引

从物理存储维度进行分类:

- 聚集索引:数据存储与索引一起存放，叶子节点会存储一整行记录，找到索引也就找到了数据。
- 非聚集索引:数据存储与索引分开存放，叶子节点不存储数据，存储的是数据行地址。

从逻辑维度进行分类:

- 主键索引:一种特殊的唯一索引，不允许有空值。
- 普通索引:MySQL中基本索引类型，允许空值和重复值
- 联合索引:多个字段创建的索引，使用时遵循最左前缀原则
- 唯一索引:索引列中的值必须是唯一的，但是允许为空值
- 空间索引:MySQL5.7之后支持空间索引，在空间索引这方面遵循OpenGIS几何数据模型规则。

2、MySQL为什么使用B+树来作索引，它的优势是什么?

1. 单点查询:B 树进行单个索引查询时，最快可以在 O(1)的时间代价内就查到。从平均时间代价来看，会比 B+ 树稍快一些。但是 B 树的查询波动会比较大，因为每个节点即存索引又存记录所以有时候

访问到了非叶子节点就可以找到索引，而有时需要访问到叶子节点才能找到索引。B+ 树的非叶子节点不存放实际的记录数据，仅存放索引，数据量相同的情况下，B+树的非叶子节点可以存放更多的索引，查询底层节点的磁盘 I/O次数会更少。

2. 插入和删除效率: B+ 有大量的几余节点，删除一个节点的时候，可以直接从叶子节点中删除，甚至可以不动非叶子节点，删除非常快。B+ 树的插入也是一样，有几余节点，插入可能存在节点的分裂(如果节点饱和)，但是最多只涉及树的一条路径。B 树没有几余节点，删除节点的时候非常复杂，可能涉及复杂的树的变形。
3. 范围查询:B+ 树所有叶子节点间有一个链表进行连接，而 B 树没有将所有叶子节点用链表串联起来的结构，因此只能通过树的遍历来完成范围查询，范围查询效率不如 B+ 。B+ 的插入和删除效率更高。存在大量范围检索的场景，适合使用 B+树，比如数据库。而对于大量的单个索引查询的场景，可以考虑 B 树，比如nosql的MongoDB。

3、什么时候需要创建索引?

- 表的主关键字：自动建立唯一索引
- 直接条件查询的字段：经常用于WHERE查询条件的字段，这样能够提高整个表的查询速度
- 查询中与其他表关联的字段：例如字段建立了外键关系
- 查询中排序的字段：排序的字段如果通过索引去访问将大大提高排序速度
- 唯一性约束列：如果某列具有唯一性约束，那么为了确保数据的唯一性，可以在这些列上创建唯一索引
- 大表中的关键列：在大表中，如果查询的效率变得很低，可以考虑在关键列上创建索引

4、什么时候不需要创建索引?

- 小表： 对小表创建索引可能会带来额外的开销，因为在小数据集中扫描整个表可能比使用索引更快。
- 频繁的插入、更新和删除操作： 索引的维护成本会随着数据的插入、更新和删除操作而增加。如果表经常被修改，过多的索引可能会影响性能。

- 数据重复且分布平均的表字段：假如一个表有10万行记录，性别只有男和女两种值，且每个值的分布概率大约为50%，那么对这种字段建索引一般不会提高数据库的查询速度。 很
- 少被查询的列： 如果某列很少被用于查询条件，那么为它创建索引可能没有明显的性能提升。
- 查询结果总行数较少的表： 如果查询的结果集总行数很少，使用索引可能不会有太大的性能提升。