1 多表关联查询语句的书写;

2 数据库事务/锁机制;

2-0:在Mysql中,事务支持是在引擎层面实现的; 比如:InnoDB支持,MyISAM不支持;

2-1: 事务的四大特性: ACID 原子性/一致性/隔离性/持久性---具体解释;

2-2: 如果事务之间不是隔离的,会出现以下问题: 脏读/不可重复读/幻读; 为了解决上面的问题,需要设置隔离级别;

**读未提交; 读已提交; 可重复读(Mysql默认); 串行化;**



2-3:Mysql数据库引擎:

Innodb: Mysql5.5之后默认; 支持事务;提供行级锁和外键约束; 适用于写 多的场景;----**聚集索引;** InnoDB的数据文件本身就是索引文件;

MyISAM:不支持事务,不支持行级锁和外键约束;适用于读 多的场景;-----**非聚集索引;** 索引文件和数据文件是分离的,索引文件仅保存数据记录的地址;

**3 数据库索引---**排好序的快速查找的数据结构 参考: <https://blog.csdn.net/tongdanping/article/details/79878302>;

优势: 加速检索,降低排序成本, 减少I/O次数 劣势:索引本身也是表,会占用存储空间; 索引表的维护和创建需要时间成本;

**索引类型:** 主键索引: 不允许重复, 不允许空值; 唯一索引:不允许重复,允许空值; 普通索引,全文索引,

**组合索引--**

**遵循最左前缀原则:把最常用作为检索或排序的列放在最左;**

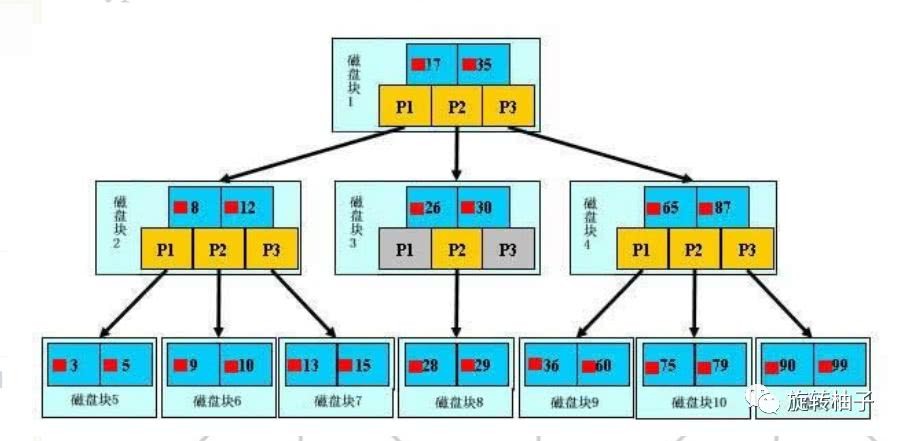
**最左前缀实现原理:**

**索引的实现原理 查询算法: 1 简单的全表扫描算法 2 索引扫描算法;**

**索引算法有多种: Btree/ Hash/ Full-text, 索引扫描算法: 类似于查字典....**

**Hash:计算索引列值得hashCode,得到数据的物理位置,一个值只能对应一个hashCode,不支持范围查找和排序功能;**

**Btree:实现的结构是B+树;**

****

**详细说明图;**

**提高查找效率的关键在于减少IO次数, 每个节点的key个数越多,那么树的高度越小,需要I/O次数就少;**

**B+tree 非叶子节点不存储数据,可以储存key; 叶子节点存储数据;**

**优化: 添加了指向相邻节点的指针,形成了带有顺序访问指针的B+tree,有利于提高区间查找效率;**



**索引的使用策略:**

主键自动建立唯一索引; 经常作为查询条件在where或 order by语句中的列建立索引;

作为排序的列要建立索引; 用于聚合函数的列可以建立索引;

**什么时候不要用索引:** 经常增删改的列; 表记录太少,不要建立索引;

**索引失效的情况:** 组合索引中有的列值为Null; 不符合最左前缀原则; like 操作中,% 在前面会使索引时效,在后面不会失效;

**索引的优化:** 1 组合索引, 符合最左前缀原则,把排序分组频率最高的列放在最左边;

**优化措施:** 1 创建索引 2 复合索引 3 索引不会包含有NULL值的列 4 使用短索引 5不要在列上进行计算 6 不使用NOT IN 操作;

**4:锁机制----针对隔离性;**

**共享锁:**

**排它锁:**

悲观锁: 一锁二查三更新; 要使用悲观锁, 要关闭mysql数据库的自动提交属性 **Select status from t\_goods where id=1 for update;**

乐观锁:假设数据不会发生冲突,只在数据提交更新的时候,会监测是否冲突; 加一个Version 字段; 每次对数据的更新操作都对版本号执行+1操作;

**5数据库优化经验:**