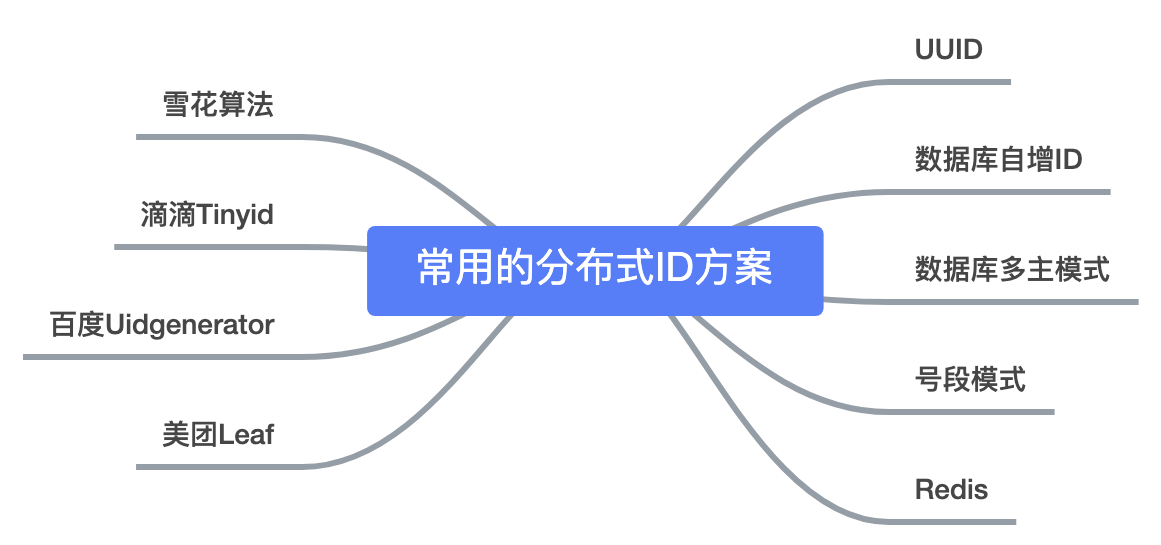
分布式ID需要满足那些条件？

1.全局唯一

2. 趋势递增：InnoDB引擎中使用B-tree聚集索引，在主键的选择上应该尽量使用有序的主键保证写入性能

3. 单调递增：保证下一个ID一定大于上一个ID，例如事务版本号

分布式ID生成方式



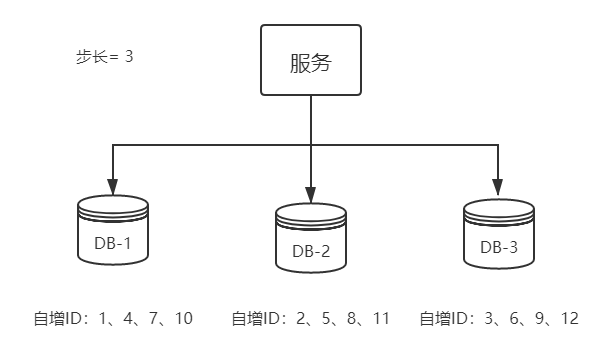
1. UUID

**优点**：简单；本地生成，无网络消耗，，性能高

**缺点**：

* UUID太长，不易于存储
* 不适用做MySQL主键，官方有明确的建议主键要尽量越短越好
* 数据库主键 UUID 的无序性会导致数据位置频繁变动，严重影响性能

1. 数据库多主模式（自增ID）



具体分两步：

1. 设置起始值：如：1，2，3
2. 自增步长：按照机器数量来设置

**缺点**：扩容的时候比较麻烦

1. 数据库的号段模式

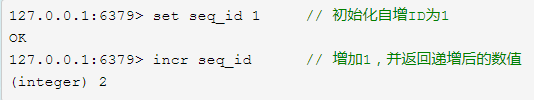
号段模式是当下分布式ID生成器的主流实现方式之一。

从数据库批量的获取自增ID，每次从数据库取出一个号段范围，例如 (1,1000] 代表1000个ID。

这种生成方式，不会频繁的访问数据库，对数据库的压力小很多。

1. 基于Redis

利用redis的 incr命令实现ID的原子性自增



* 风险：要考虑到redis持久化的问题。

RDB会定时打一个快照进行持久化，假如连续自增但redis没及时持久化，而这会Redis挂掉了，重启Redis后会出现ID重复的情况。

AOF会对每条写命令进行持久化，即使Redis挂掉了也不会出现ID重复的情况，但由于incr命令的特殊性，会导致Redis重启恢复的数据时间过长。

1. 雪花算法（Snowflake）

Snowflake生成的是Long类型的ID。利用时间戳、工作机器id、序列号计算得来。

