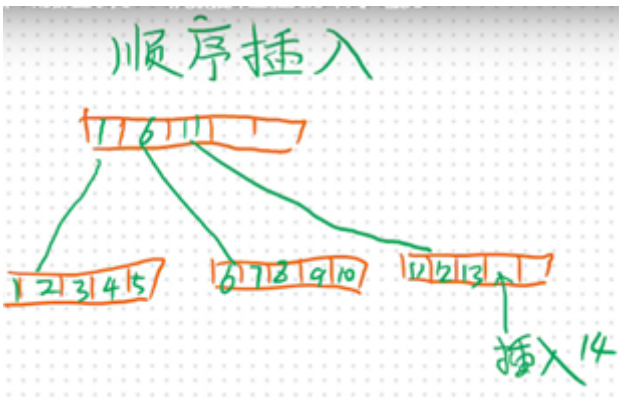

| MySQL | PostgreSQL |
| --- | --- |
| InnoDB的表和索引都是按相同的方式存储。也就是说表都是索引组织表。这一般要求主键不能太长而且插入时的主键最好是按顺序递增，否则对性能有很大影响。 | 不存在这个问题。 |
| 大部分查询只能使用表上的单一索引;在某些情况下，会存在使用多个索引的查询,但是查询优化器通常会低估其成本,它们常常比表扫描还要慢。 | 不存在这个问题 |
| 表增加列，基本上是重建表和索引，会花很长时间。 | 表增加列，只是在数据字典中增加表定义，不会重建表 |
| 存储过程与触发器的功能有限。可用来编写存储过程、触发器、计划事件以及存储函数的语言功 | 除支持pl/pgsql写存储过程，还支持perl、python、Tcl类型的存储过程：pl/perl、pl/python、 |

# Mysql采用B+树索引

主键为聚簇索引, 索引叶子节点存有数据,而且是顺序递增,正向排列

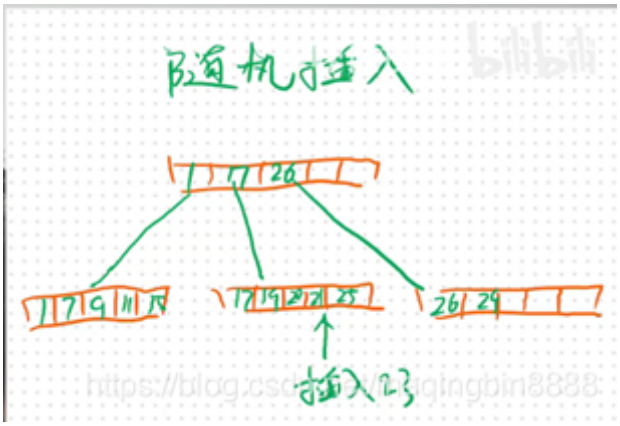
随机id插入的时候会比较慢

缺点:**随机插入就会产生了很多不满的节点,即在数据量一样的情况下随机插入产生了更多不满的节点，空间差很大 效率也不行 本地的,远程调用**



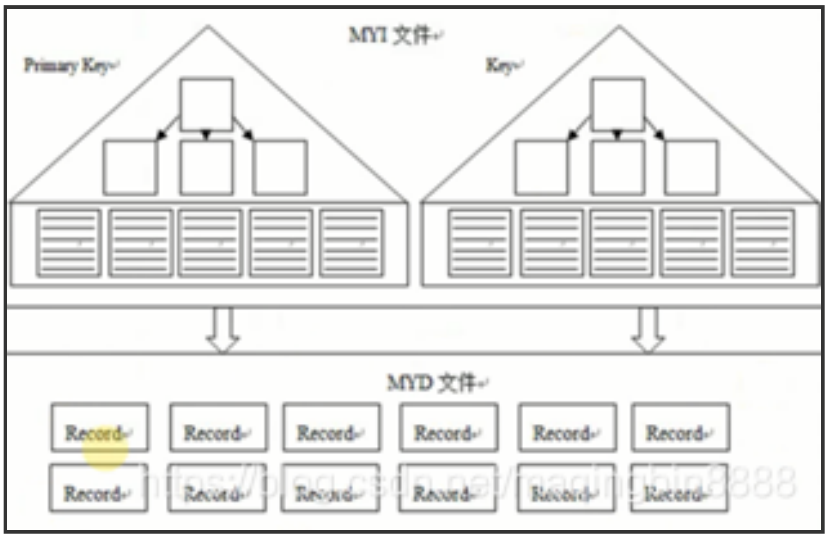

## id随机的方式插入

如果按照随机的方式插入,那么他就有可能插入到一个已经满的节点上,这时节点就会分裂,分裂成两个节点 就有可能每个节点上存储了三条数据



# Postgresql使用B树索引

- pg默认采用堆组织表方式存储数据
- 数据跟索引是完全分开的
- 堆表（heap table）数据插入时**存储位置是随机的**，主要是数据库内部块的空闲情况决定，获取数据是按照命中率计算，全表扫表时不见得先插入的数据先查到。

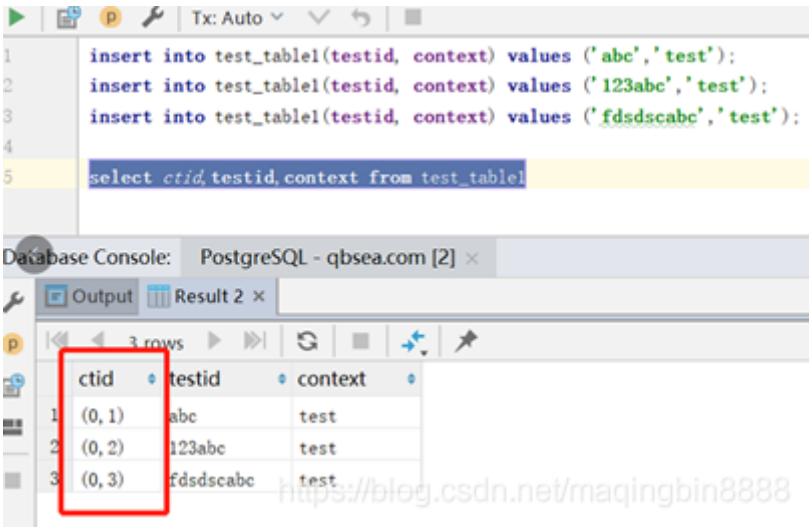



- 堆就是无序数据的集合,索引就是将数据变得有序,在索引中键值有序,数据还是无序的
- 堆表中,主键和普通索引基本上没区别,和非空的唯一索引没区别
- 堆表中,主键索引和普通索引一样的,叶子节点存放的是指向堆表中数据的指针（可以是一个页编号加偏移量）,指向物理地址,没有回表的说法-- 意思是通过索引顺序从堆里一直找每个数据

## CTID-- 表示数据行在它所处的表内的物理位置

PG数据索引的存储顺序并**不以某一列的排序顺序来存储**, 而是以行CTID号来存储(每一行有一个行ID号,并且这个行ID号是一个坐标)

即新增数据永远是在**索引存储的最后一个叶子**,即不会存在B+Tree的页分列



# SQL的远足、页面结构及索引查找原理

postgresql数据库通过数据多版本实现mvcc，pg又没有undo段，老版本的数据元组直接存放在数据页面中，这样带来的问题就是旧元组需要不断地进行清理以释放空间，这也是数据库膨胀的根本原因。

https://blog.csdn.net/weixin_41287260/article/details/120472311