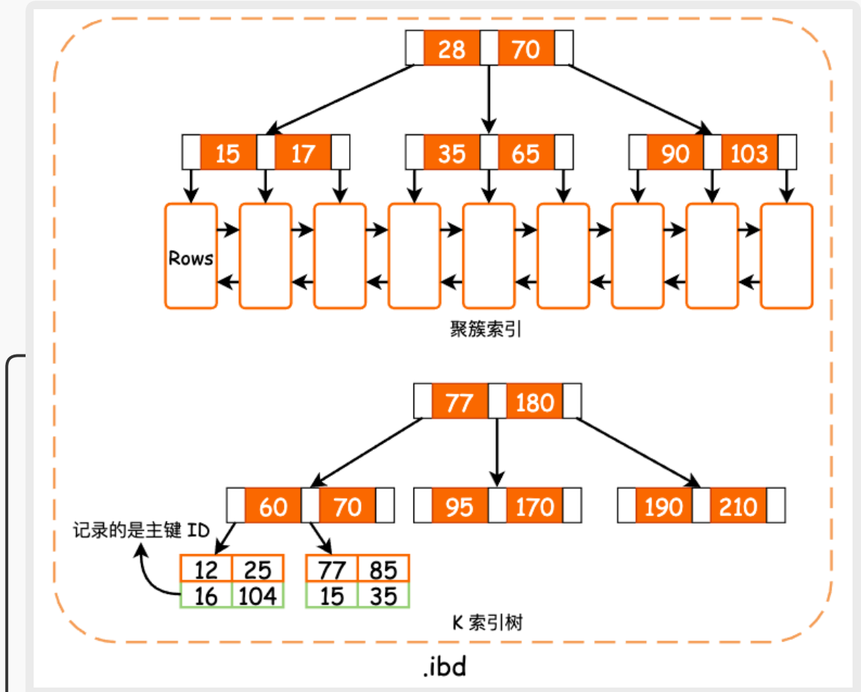
数据组织方式

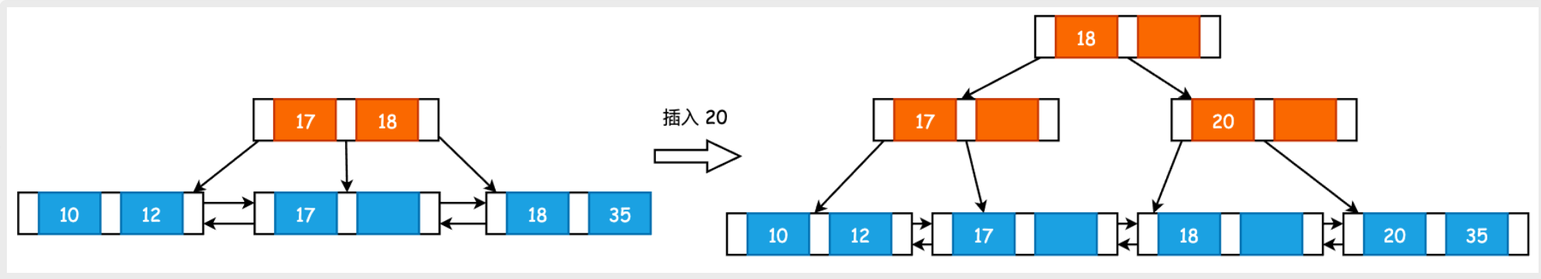
索引组织表 (indexed organized table)



如上图所示，MySQL InnoDB 存储引擎采用了经典了 B+Tree 索引组织方式。简单地来说，索引即数据，数据及索引

在索引组织表中，数据将有序存放，这里既可以使用经典的 B+Tree 或者是 B-Tree，也可以使用最简单的二分搜索树。索引组织表强调的是数据的有序存放，不在乎实现方式

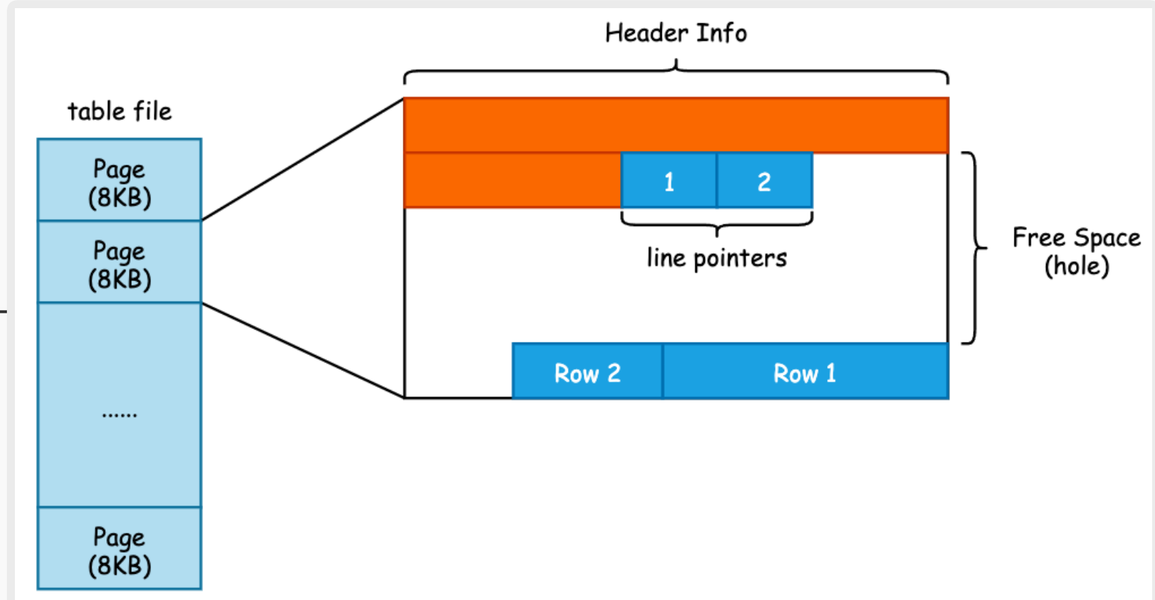
- 查询
  - 主键查询
    - 由于在聚簇索引中数据就是通过主键组织在一起的，因此使用主键查询某一行数据时速度将会非常快，只需要在聚簇索引中进行少量的逻辑 I/O 即可
  - 普通索引查询
    - 在使用普通索引查询一整行数据时，需要首先到 K 索引树找到该索引对应的主键 ID，然后再根据主键 ID 去聚簇索引中查询整行数据。这个过程称之为回表，覆盖索引就是一种去除回表的优化方式
    - 也就是说，对于普通索引查询来说，假如聚簇索引和 K 索引树的高度均为 3 的话，那么取出一行数据则需要 6 次逻辑 I/O
- 插入
  - 以 B+Tree 为例，其插入过程按照 "当前节点-兄弟节点-父亲节点" 的顺序依次插入，当某一页的数据已经达到最大容量时，会触发页分裂机制，这是一个递归的过程，将直接影响数据库的插入性能



  - 如上图所示，在一个 3 阶的 B+Tree 中插入元素 20 时，可能会导致多个节点发生变动。这些过程发生在磁盘文件中，因此在 InnoDB 存储引擎中，主键的随机插入将会导致非常严重的性能问题
- 综上，当数据使用索引组织表的形式存储以后，数据在物理结构上即为有序。因此，使用主键查询的效率非常之高，但随机插入的性能较低，因为要维持 B+Tree 的有序性

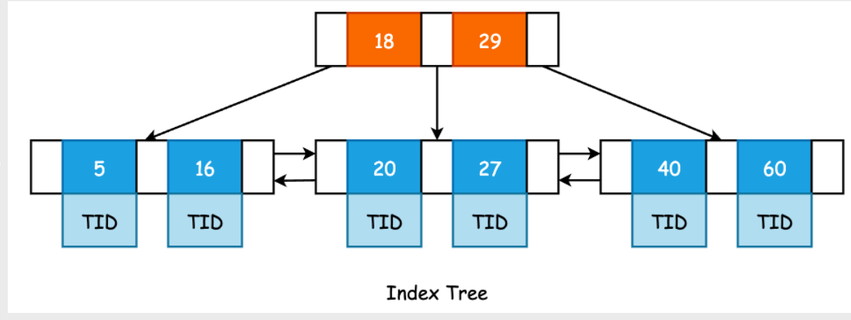
堆表（heap table）

堆表是 PostgreSQL（下简称为 PG） 所默认使用的数据组织方式，这里的堆既不是指数据结构中的二叉堆，也不是虚拟内存中的堆内存，个人理解这里应该表示的是 "堆叠" 之意: 即数据的存放是无序的



如上图所示，PG 中的数据表被划分成固定长度的页（Page，默认大小为 8KB，MySQL 一页默认为 16KB）。数据记录，或者说行数据，从页面底部开始堆叠。当行数据被删除时，这部分的空间会被重用，也就是说，数据组织是无序的

- 查询
  - 虽然行数据本身是无序存储的，但是主键 ID 却依然使用 B-Tree 的形式进行存储，显而易见地，在这棵 B+Tree 中，除了主键 ID 以外还必须存储主键对应行记录的物理地址（文件偏移量），这个物理地址通常被称之为 TID（tuple id）



  - 如上图所示，在 PG 中，不管是使用主键查询，还是使用普通的二级索引查询数据，都需要在对应的索引树上查找到记录行的 TID，然后拿着 TID 再去堆表中进行查询（使用 TID 取出一行数据的时间复杂度为 O(1)）。也就是说，主键查询的效率要略低于索引组织表
- 插入
  - 由于堆表并不需要保证数据的有序性，因此在插入数据时，只需要找到一块儿空地即可，因此插入性能要高于索引组织表

小结

	索引组织表	堆表
数据组织方式	数据排列顺序于数据的插入、删除顺序无关，数据按规则有序存储	数据排列顺序于数据的插入、删除顺序有关，数据无序存储
随机插入性能	低	高
主键范围查询效率	优于堆表	低于索引组织表
主键范围查询效率	高	可能较低
二级索引查询效率	需回表	无需回表