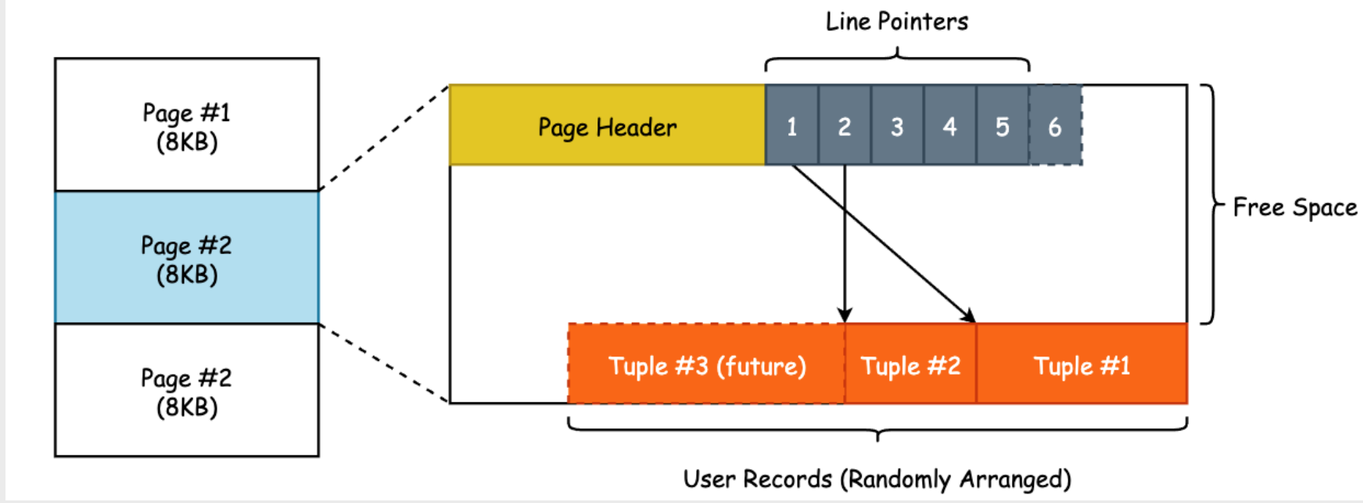
PostgreSQL 存储结构

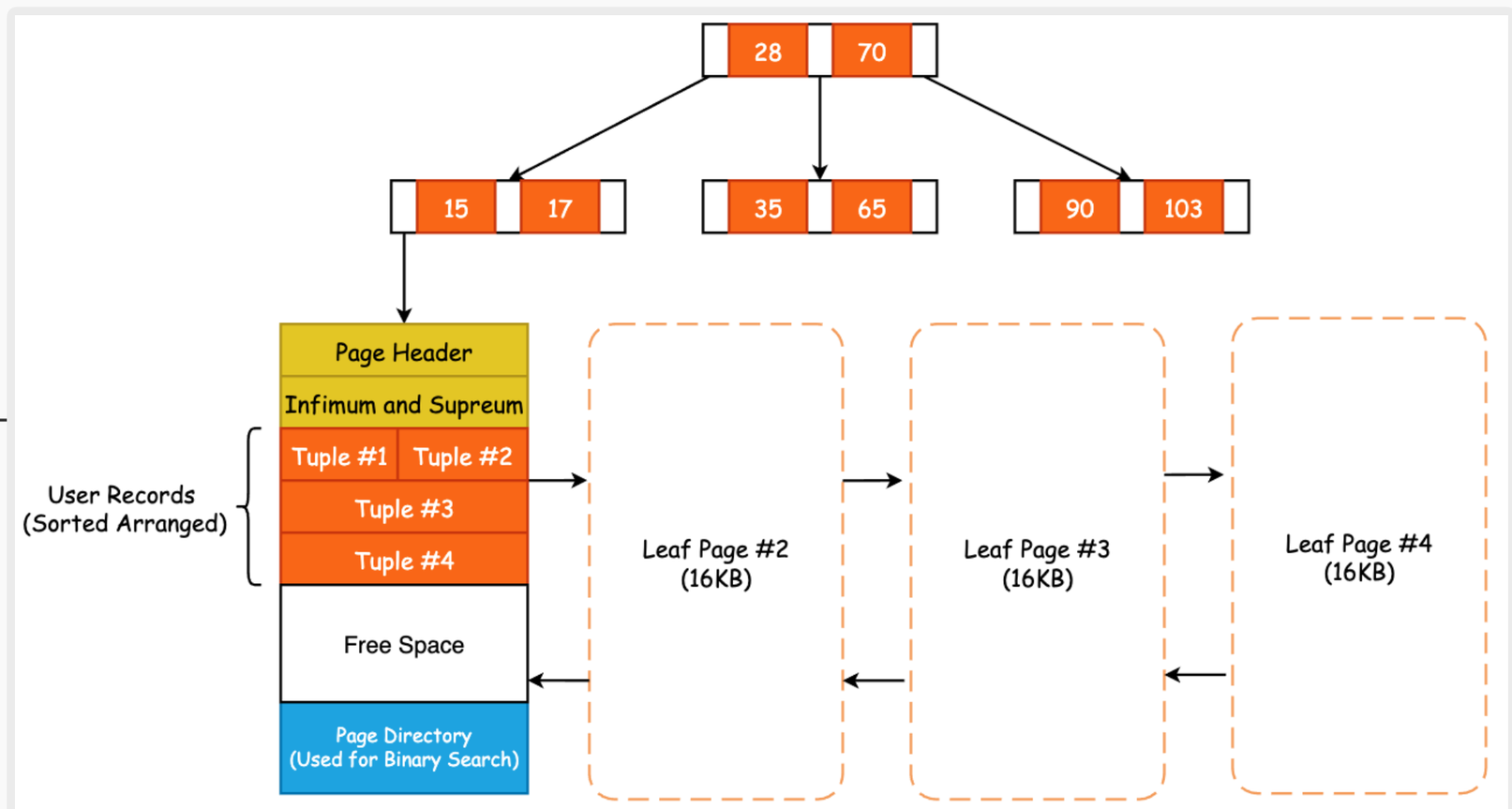
Heap File

Heap File 又被称之为堆表，这里 "heap" 的含义既不是操作系统中虚拟内存中堆内存，也不是数据结构中的二叉堆或者多叉堆，它其实就表示了数据（元组）在 PostgreSQL 中是堆叠而存储的



上图即为 PostgreSQL 堆表文件的内部布局，和其他 DB 一样，数据文件被划分成若干个固定长度的页，其大小为 8KB

- Page 布局
  - Page Header — 保存了校验和、最后一次变更所写入的 XLOG 记录对应的 LSN 等信息，属于 Meta 信息
  - Line Pointers
    - 又可以被称作为 Slots Array，是一个数组，扮演着 Tuple 索引的角色
    - 我们可以简单地将数组内容理解为句柄，用于快速定位到某一个 Tuple 的起始位置。每新增一个 Tuple，一个对应的行指针就会被添加至数组中
  - Free Space — 空闲区域
  - User Records
    - 从上图我们可以看到，用户数据实际上就是一个挨一个地存放的，并且它们是从页面底部开始堆叠的，这么做的目的在于便于 Line Pointers 数组的自然增长
    - 一个 Tuple 除了包含用户所存储的行数据以外，还会包含事务和 MVCC 相关的信息，例如 xmin、xmax
- 与 MySQL 的对比



如上图所示，MySQL 通过索引组织表（聚簇索引）的方式组织用户数据，和 PostgreSQL 的堆表有着非常明显的区别

在 MySQL 中，数据即索引，索引即数据，通过主键 ID 将 User Records 有序存放。如此一来，主键 ID 的范围查询效率将会非常高，直接顺序遍历聚簇索引即可

而对于 PostgreSQL 来说，User Records 并不会以某种顺序进行排列，及时在最初时刻以插入顺序排列，后续也会因为 VACUUM 等操作对 Tuple 进行重排导致乱序。因此，必须要有一个额外的索引树来保证其查询效率

为了标识表中的元组（Tuple），PostgreSQL 使用元组标识符（Tuple Identifier, TID），由两部分组成：元组所属页号和指向元组的行指针的偏移量

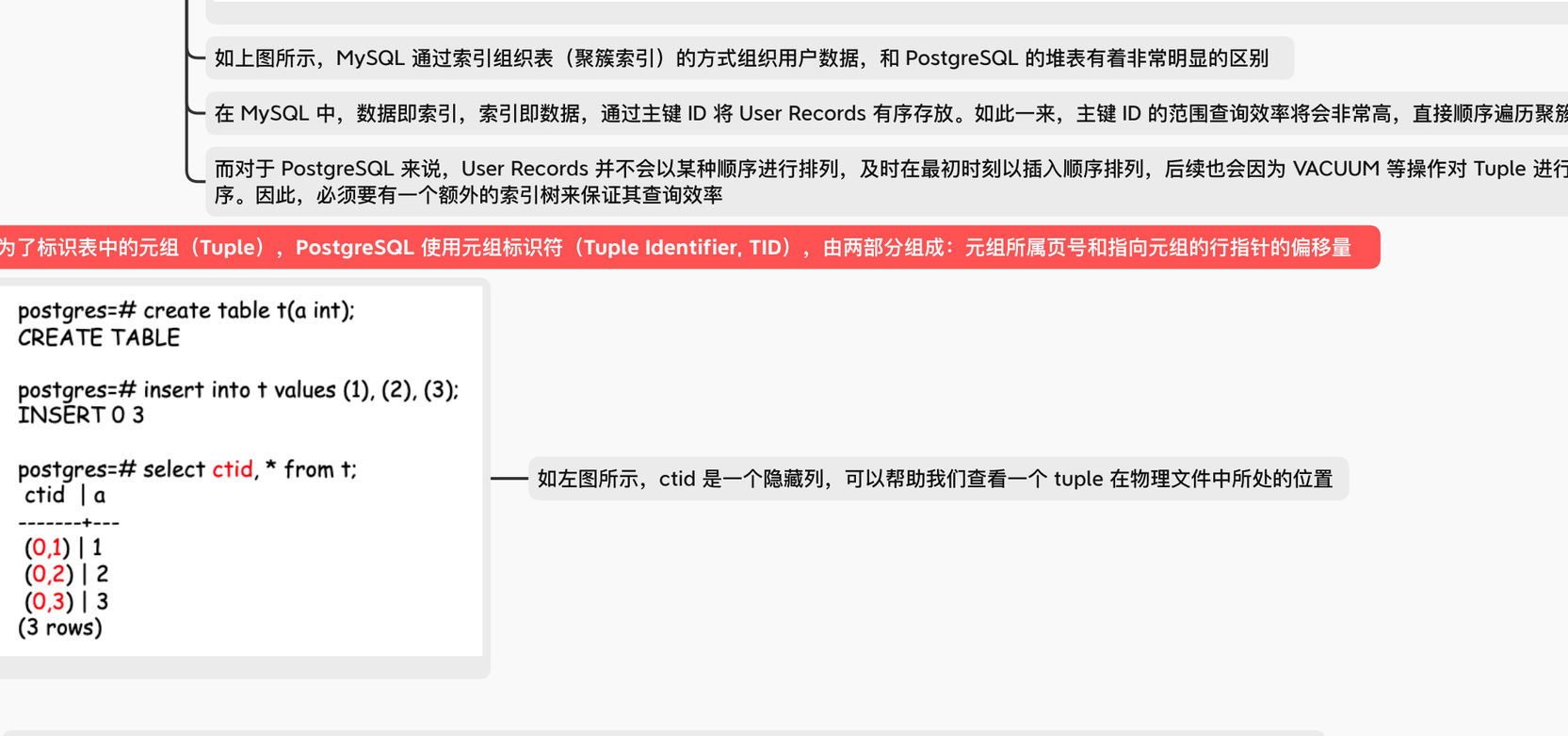
```
postgres=# create table t(a int);
CREATE TABLE

postgres=# insert into t values (1), (2), (3);
INSERT 0 3

postgres=# select ctid, * from t;
 ctid  | a
-------+---
 (0,1) | 1
 (0,2) | 2
 (0,3) | 3
(3 rows)
```

如左图所示，ctid 是一个隐藏列，可以帮助我们查看一个 tuple 在物理文件中所处的位置

数据的修改

首先，update 动作在 PostgreSQL 中将会由两个动作完成: 逻辑删除旧元组 + 插入新元组，具体可参考: https://smartkeyerror.oss-cn-shenzhen.aliyuncs.com/Phyduck/database/PostgreSQL%20%E4%B8%AD%E7%9A%84%20MVCC.pdf

PostgreSQL 为了尽可能地减少写放大，采用了 HOT (Heap Only Tuple) 的方式进行优化。简单来说就是通过新增和修改指针引用的方式来减少数据写入量，但增加了读取数据时的读取成本



如上图所示，如果新插入的元组可以和旧元组在同一个页面的话，我们只需要将旧元组的 t_ctid 指向新元组，同时更新一些标识位即可。此时完全不必更新索引表，插入效率将会非常高