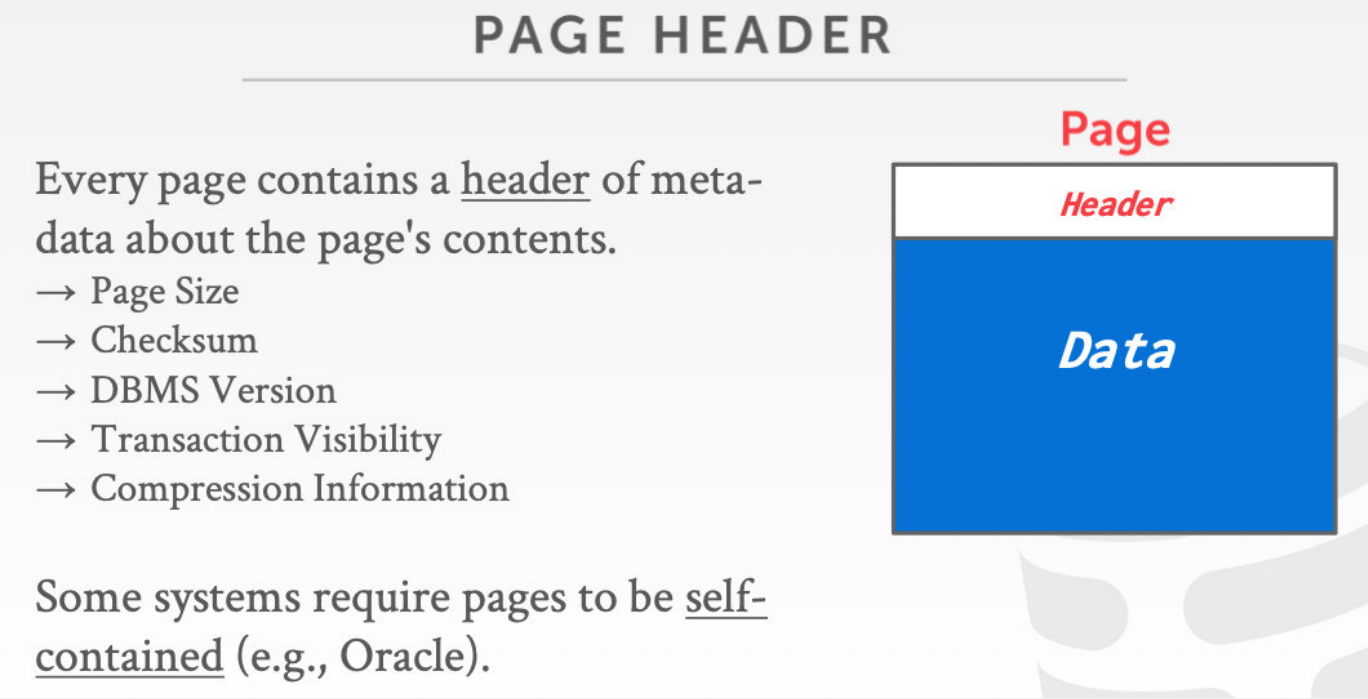
P.s.本archive是笔者通过CMU15-445课件学习时基于个人理解的补充

# 关于Page Header




存储引擎所管理的数据库的每个页中的header里会存储各种信息，其中Checksum是用于校验数据存储的正确性的，可以理解成对页里面所存储的数据进行类似于哈希运算这样的用于校验的运算，如果运算的结果，即Checksum和数据源得到的Checksum一致，那就表明数据存储没有出现偏差，在通过网络下载文件时，也会有相应的用于校验的Checksum

Transaction Visibility，翻译成中文，大概就是事务可见性，事务是数据库中很重要的概念，在有些时候，比如说我们要修改数据库中某个页里的某个tuple的信息时，会将它锁住，此时不允许其他的线程来访问，这么做可以保证事务的原子性

为了让有限的磁盘空间能存储更多的数据，数据库的文件中的数据会有压缩，因此每个page会有一些和数据压缩有关的字段，便于解压缩

至于数据库文件的页的self-contain，我们可以选择让数据库文件的页的data只包含数据库的表的信息，再额外花费一个页来存储meta data，用它来说明每个page所存储的数据的格式，也可以说是这个meta data页描述了每个普通页存储的数据的数据结构，但如果想实现self-contain，即自解释，那么就需要舍弃一部分data的空间，用来记录页存储的数据的格

式，即每个页都有meta data

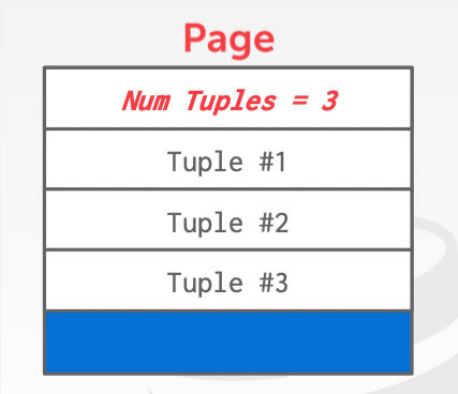
# 关于Page Layout

关于如何组织每个页内部的数据，如果是存储一堆tuple的话，可以有如下的方法

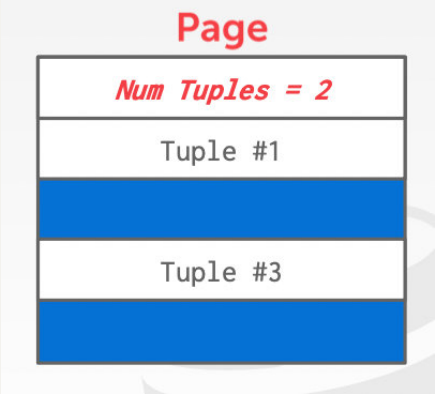
一个比较简单的方法就是如下所说的*Strawman Idea*

> **Strawman Idea:** Keep track of the number of tuples in a page and then just append a new tuple to the end.

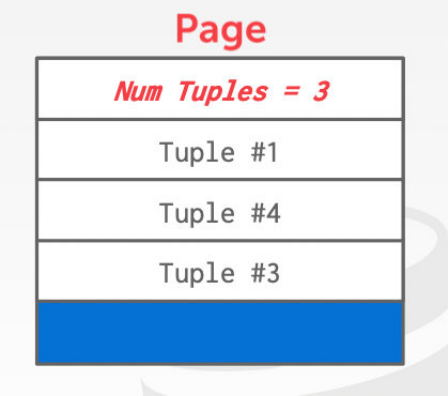
如果每个tuple变长，并且我们有可能删除之前插入的tuple的话，会破坏天然的顺序，也有可能导致类似于操作系统内存管理中的“外部碎片”。浪费存储空间，比如下面这种情况




Next->

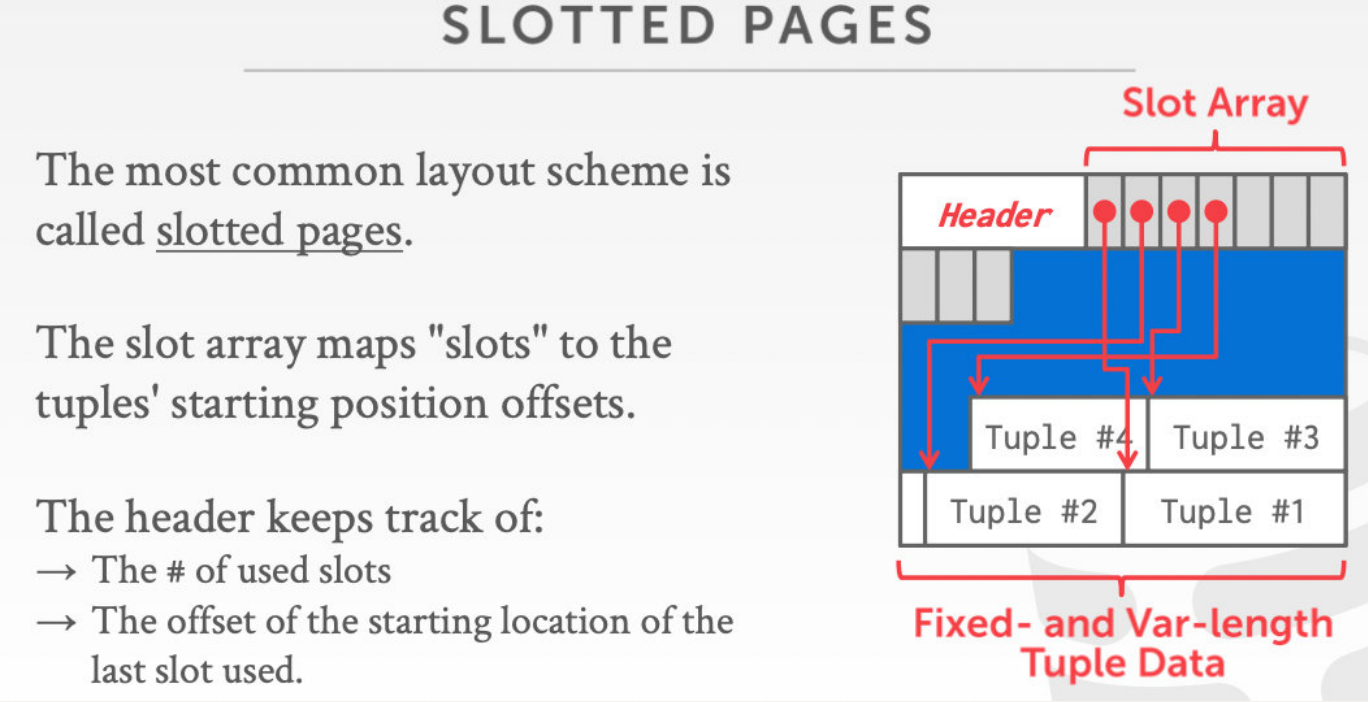



Next->

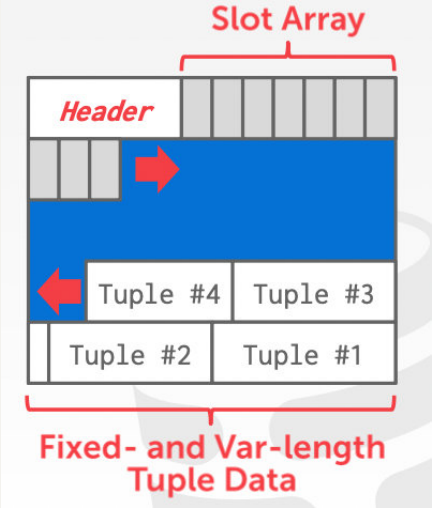



比较常见的做法是slotted pages策略

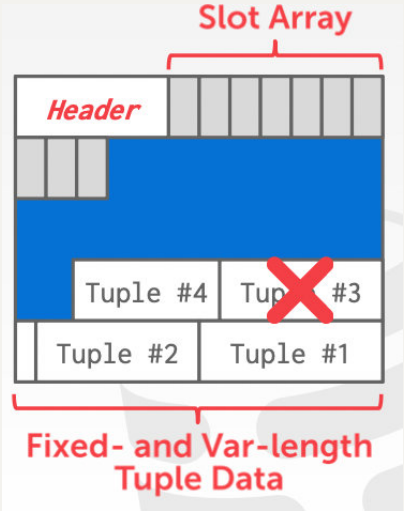
在data区域的最前面有一个slot数组，其中的每个元素和一个tuple构成一对一的映射关系，每个slot元素里面会记录它所映射到的tuple在页的哪个位置，



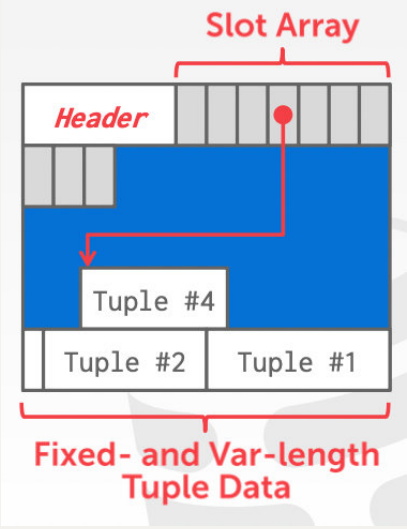

slot是从页的data区域的最前面往后存，tuple是从data区域的最后面往前存，这样可以最大化的利用这一片区域，



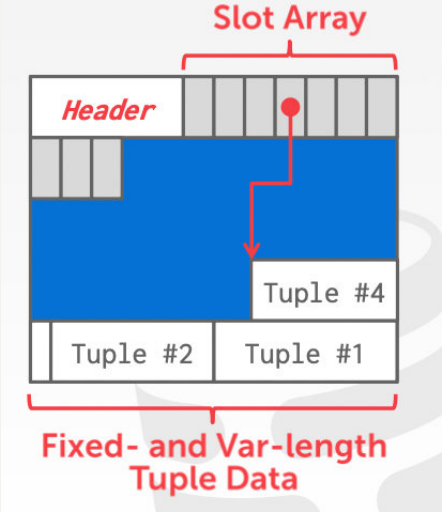

并且tuple之间可以是乱序的，而且slot的数据可以动态更新，这使得我们可以在删掉某个tuple之后移动其他tuple的位置，从而整理data区域，让tuple之间紧凑，这个过程如下所示



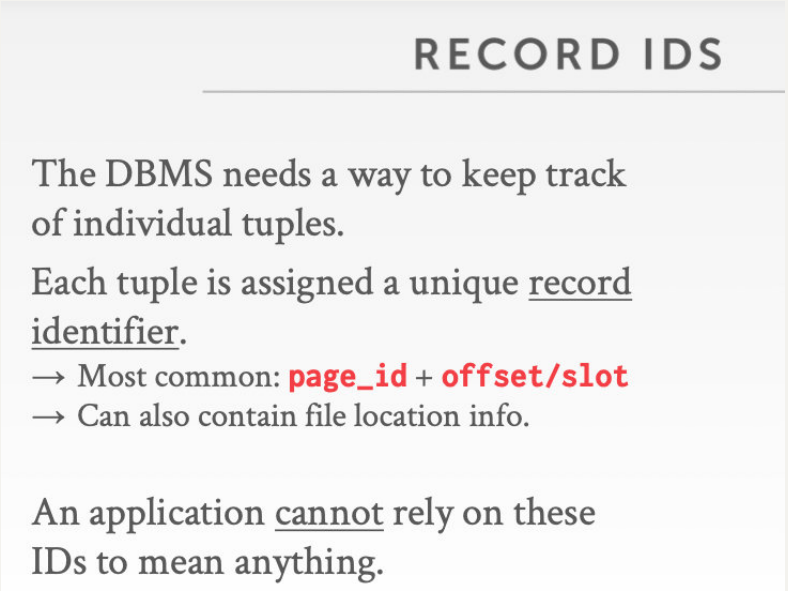

Next->




Next->




关于tuple的Record ID




"An application cannot rely on these IDs to mean anything."说的便是，从程序员/数据库使用者的角度，不能通过某个tuple的record id去索引它，因为有可能文件经过某种整理之后，某个页的页号会发生变化，还会有其他的因素导致信息的变化

# 关于Tuple Layout

## TUPLE HEADER

Each tuple is prefixed with a header that contains meta-data about it.
→ Visibility info (concurrency control)
→ Bit Map for NULL values.

We do not need to store meta-data about the schema.


每个tuple也有header，里面会有一些用于并发控制/事务相关的数据

不管数据库的页是不是前面的“自解释”型，都不会在tuple里面记录数据库的表的格式，这样就太浪费存储空间了，顶多会像前面说的，在页的data区域里花一点空间来记录

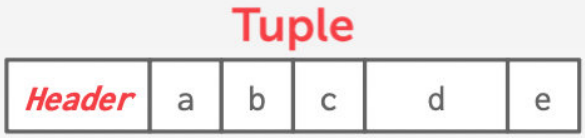
每个tuple的attribute data内部没有分隔符，因此需要header里面的bitmap来标注tuple的attribute data里哪些字段是NULL，从而防止误读，字段间没有分隔符的attribute data的结构如下所示

## TUPLE DATA

Attributes are typically stored in the order that you specify them when you create the table.

This is done for software engineering reasons (i.e., simplicity).

However, it might be more efficient to lay them out differently.




```
CREATE TABLE foo (
  a INT PRIMARY KEY,
  b INT NOT NULL,
  c INT,
  d DOUBLE,
  e FLOAT
);
```

Reference:
https://www.bilibili.com/video/BV1km4y1X7Tq/?spm_id_from=333.788