CnosDB

数据库系统 | Database Systems

第三课：存储-1

Lecture3: **Storage 1**

Lecturer: Harbour

Date: 2021.11.22

Scan CC WeChat to Join the Community

添加CC好友，接受进群邀请

Welcome to follow the GitHub repo

欢迎关注我们的代码仓库

https://github.com/**cnosdatabase/cnosdb**

GitHub

00

计划教学内容

CnosDB

必讲

DB/DBMS

关系模型与关系代数

数据库存储

散列索引（哈希）

B+树

查询处理

并发控制

如果有兴趣

SQL

查询优化

恢复系统

分布式OLTP/OLAP

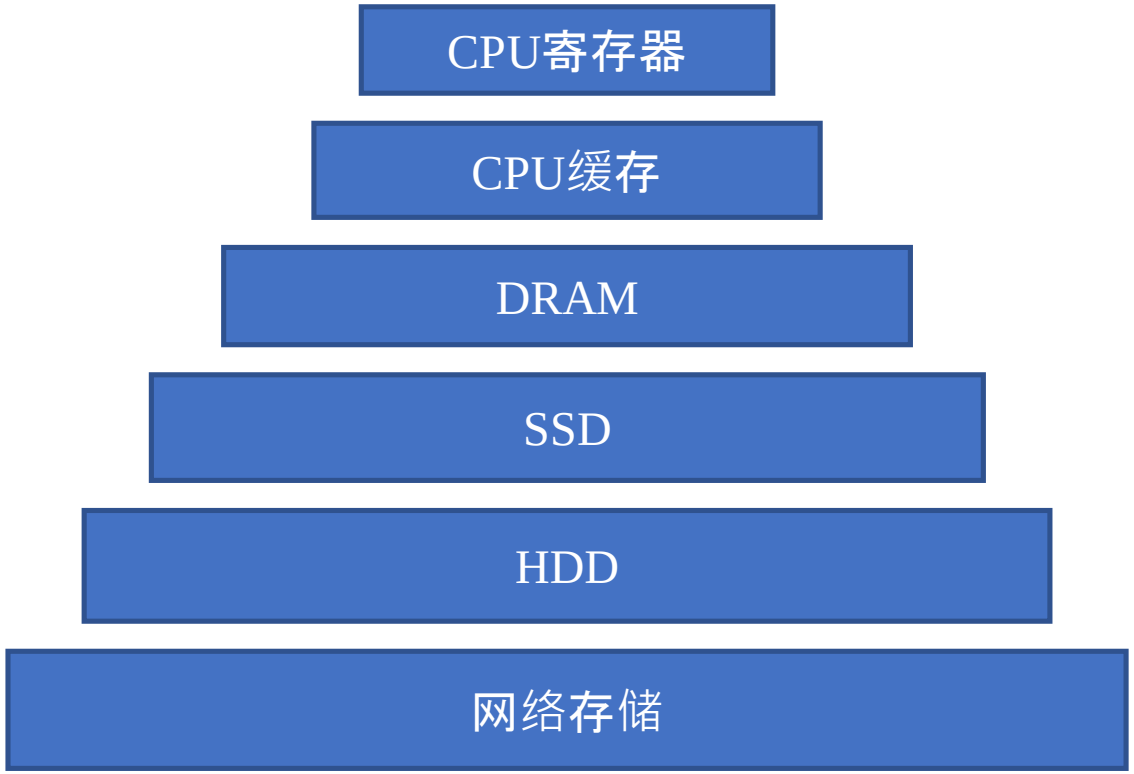
01

面向磁盘的存储逻辑

CnosDB

- DBMS 假定主存储数据库的位置在非易失性磁盘上。
- DBMS 的组件管理非易失性和易失性存储之间的数据交换。

存储层级



访问时间

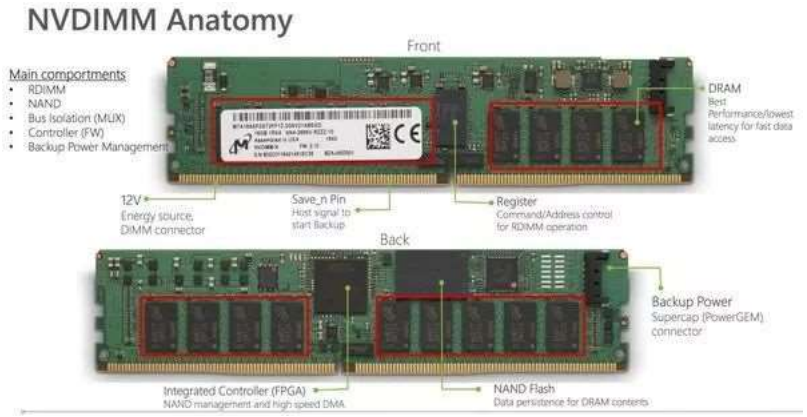
CnosDB

0.5 ns	L1 Cache Ref
7 ns	L2 Cache Ref
100 ns	DRAM
150,000ns	SSD
10,000,000 ns	HDD
~30,000,000 ns	Network Storage
1,000,000,000 ns	Tape Archives

拓展阅读

美光发布新款NVDIMM-N非易失性内存

NVDIMM是一种比较普遍的易失性内存设计方案，
其原理便是在内存上集成了非易失的颗粒，
NVDIMM-N意味着它是N类型，使用NAND闪存也
就是本次美光发布的产品



拓展阅读

CnosDB

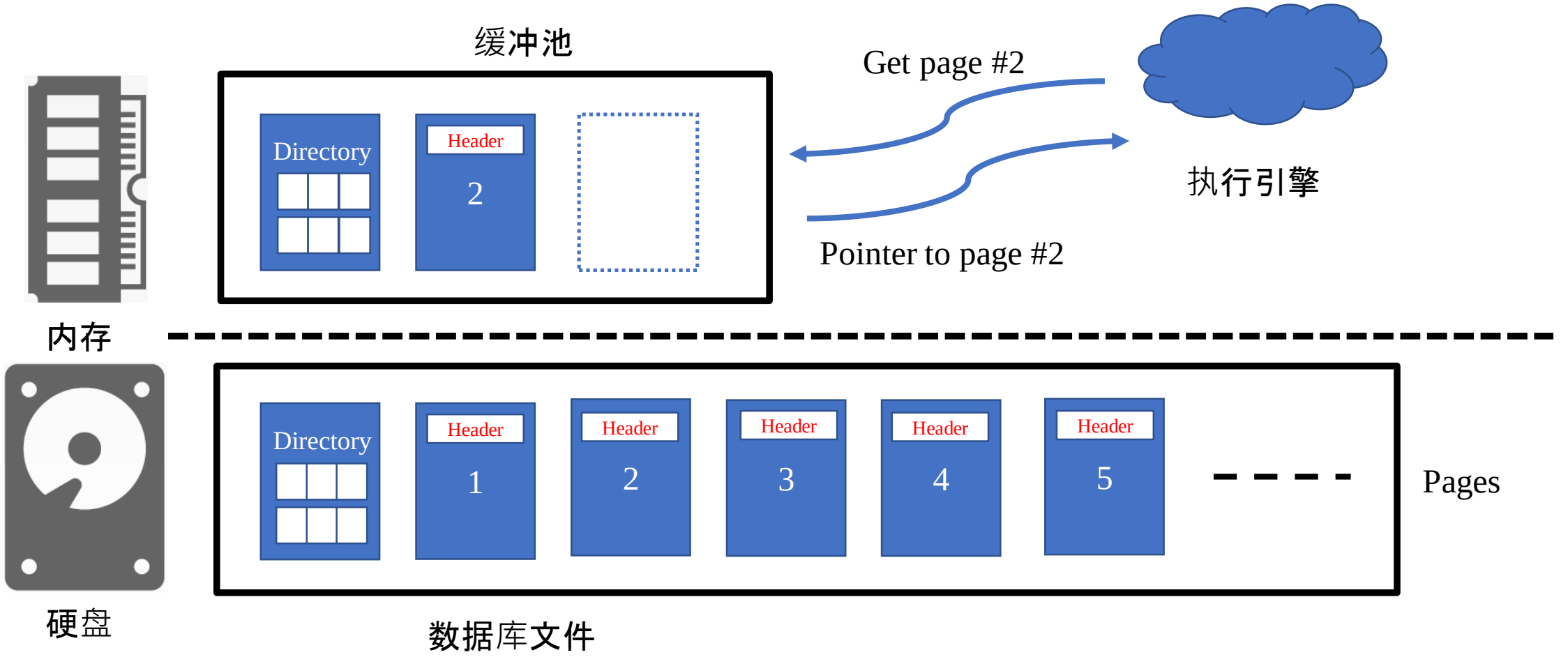
为什么顺序读写比随机读写快？

1. 寻道时间
2. 旋转延迟
3. 传送时间

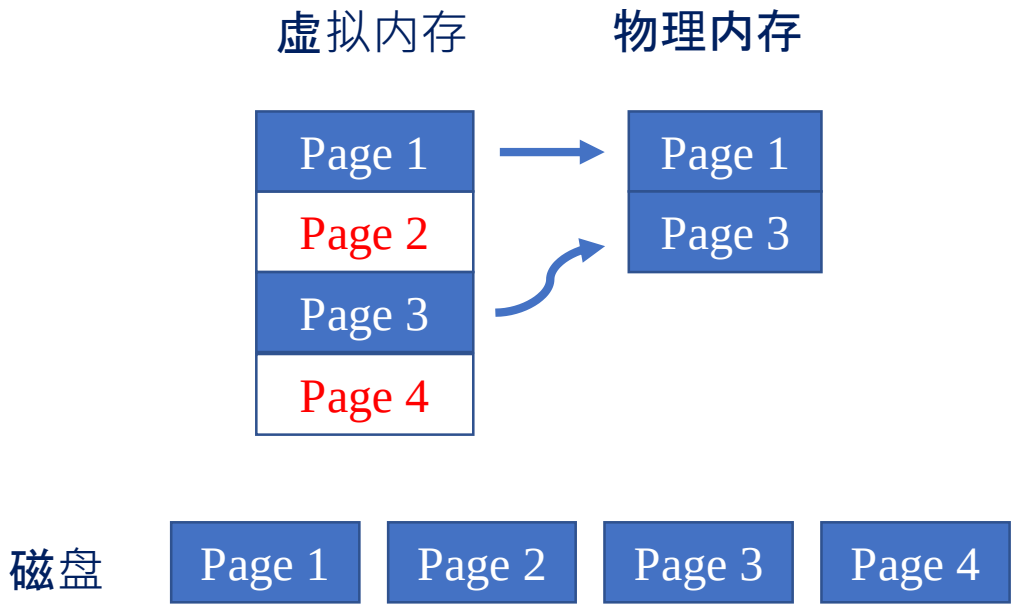
系统设计目标

- 允许 DBMS 管理超出可用内存量的数据库。
- 对磁盘的读/写开销很大，因此必须谨慎管理以避免大停顿和性能下降。

面向硬盘的数据库管理系统



数据库vs操作系统



实现虚拟内存的一种方式是使用mmap**去映射一个文件的内容到一个内存地址空**间，**操作系**统负责从硬盘和内存之间移动页。**但不幸的是，如果**mmap**遇到**页错误，进**程将被阻塞。**

数据库vs操作系统

CnosDB

➢ OS可以做的

- madvise：告诉操作系统什么时候你计划读取特定的页。
- mlock：告诉操作系统不要将内存交换到硬盘上。
- msync：告诉操作系统同步内存的内容到硬盘上。

➢ DB更擅长做的

- **以正确的**顺序将脏页刷新到磁盘。
- 专门的预读取。
- 缓冲区的替换策略。
- 线程/进程调度。

文件存储

- 在最基本的形式中，数据库将数据存成文件。
- OS不必了解，只有DBMS知道。

数据库存储管理器

CnosDB

➢存储管理器负责维护数据库的文件。

- 改善Pages的空间和时间局部性。

➢将文件组织为一组Pages。

- 跟踪读取/写入Pages的数据。
- 跟踪可用空间。

数据库的页

CnosDB

- Page就是固定大小的数据块（block）
  - 元组tuples，元数据meta-data，indexes索引，log records日志记录
  - 不会混合这些类型
  - self-contained 自包含

- 每个页面都有一个唯一的标识符。
- DBMS 使用中间层将page id 映射到物理位置。

数据库的页

CnosDB

- 使用固定长度的页，而不是可变长度的页，为什么？

这里有3个数据库中page的概念：

1. 硬件的页（通常为4KB）
2. 操作系统的页（4KB）
3. 数据库的页（1-16KB）

数据库的页

什么是“故障安全写入”的级别？

页的存储结构

CnosDB

- DBMS**以不同的方式管理磁**盘文件中的页面。

  顺序文件组织

  堆文件组织

  哈希文件组织

  ISAM(Indexed sequential access method)

  B+

  Cluster

堆文件

CnosDB

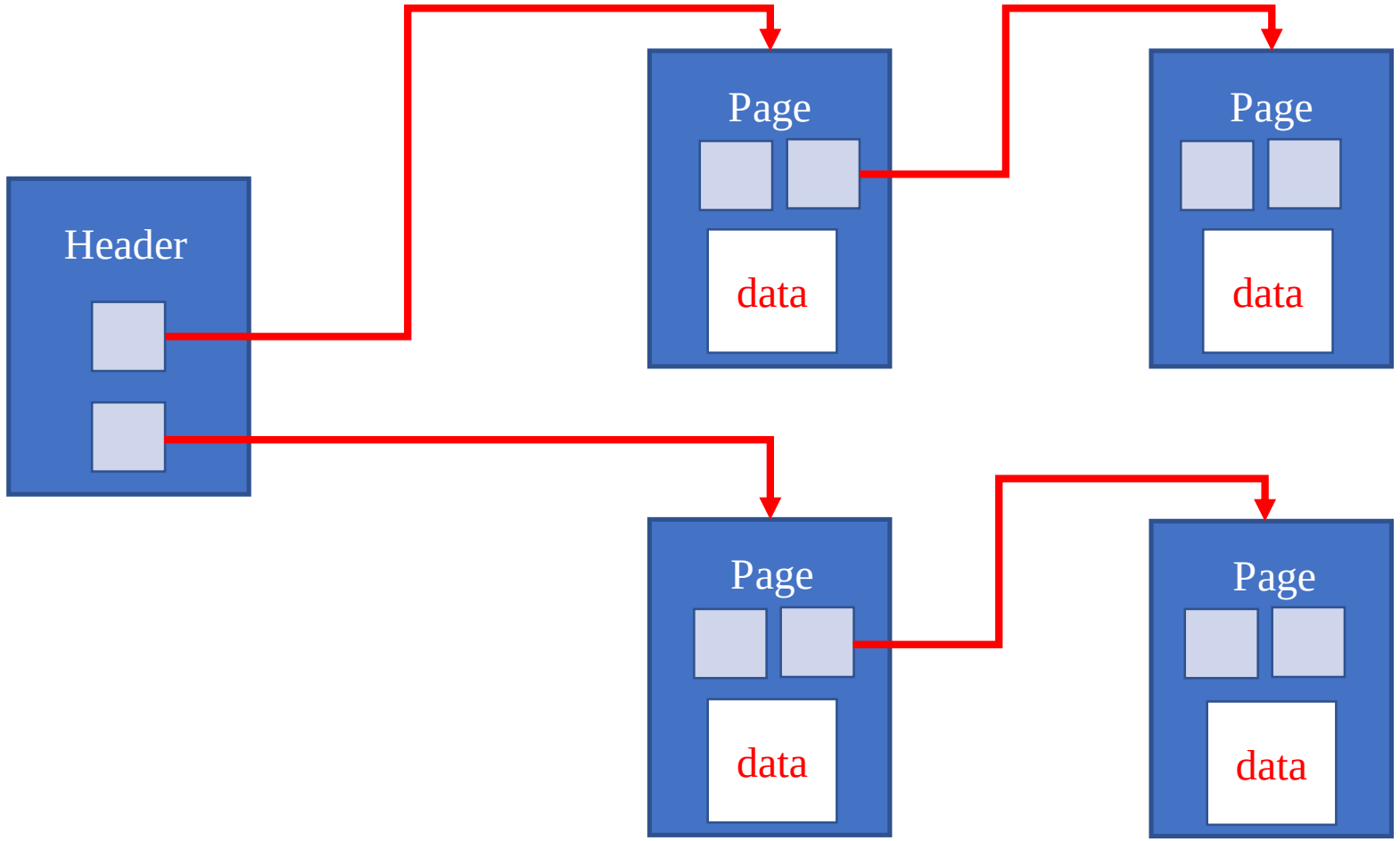
- **堆文件是一个无序的**页面集合，其中元组以随机顺序存储。

  CGWD Pages

  支持遍历所有的Pages

- **需要元数据来跟踪哪些**页面存在以及哪些页面具有可用空间。

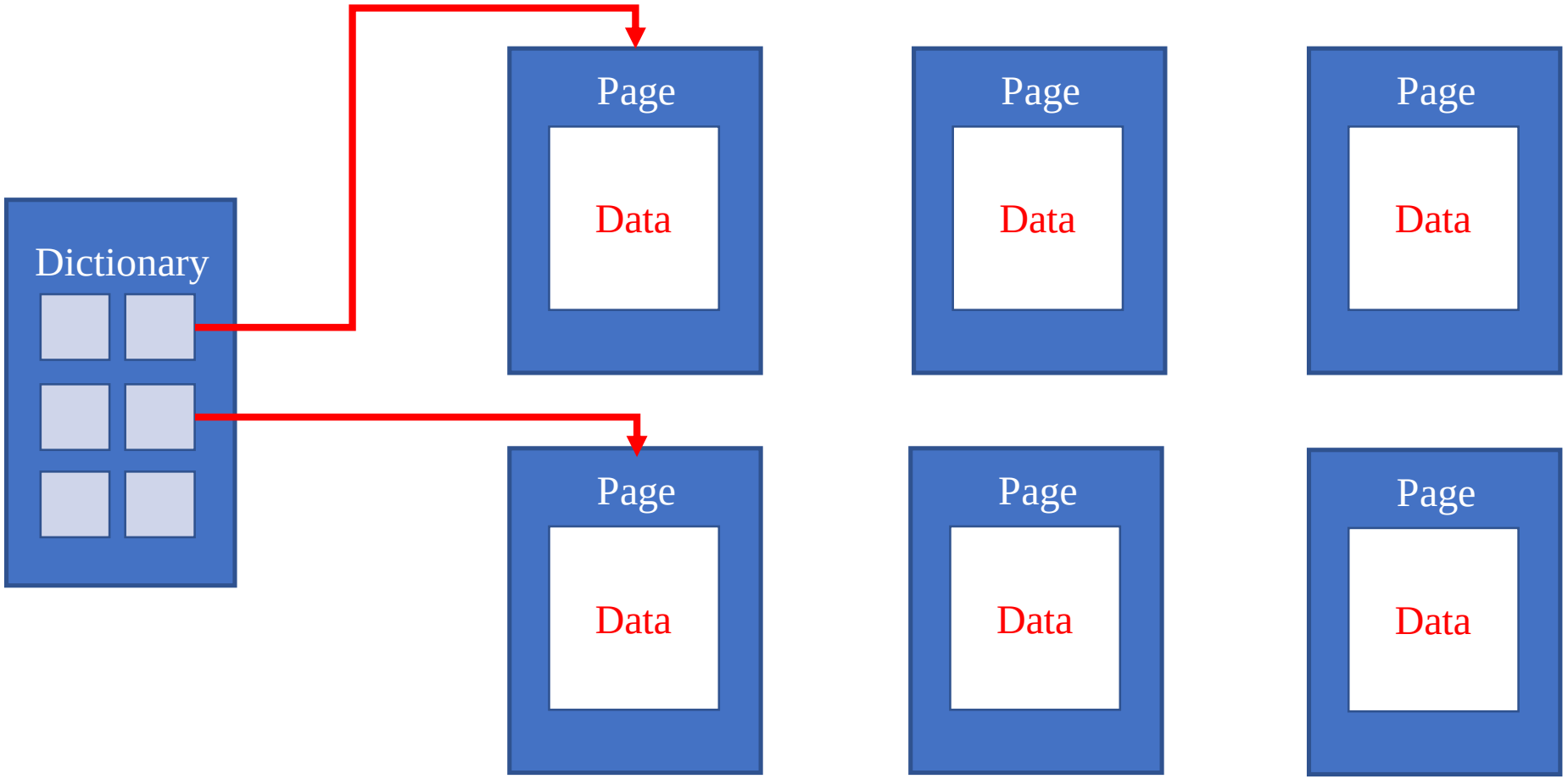
链表



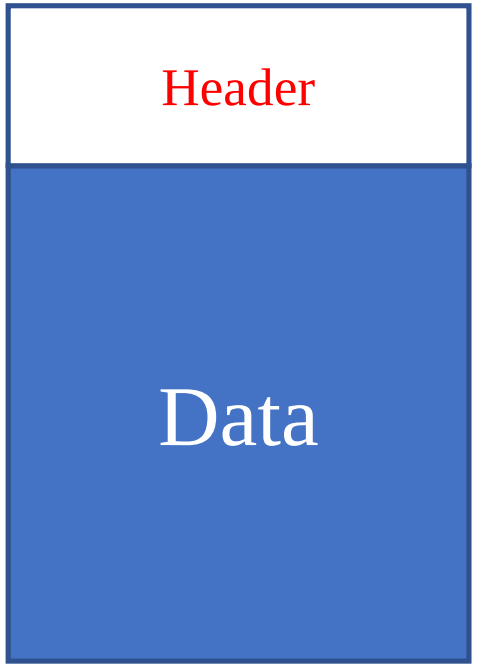
Free Page List

Data Page list

页字典


Dictionary
Page
Data
Page
Data
Page
Data
Page
Data
Page
Data
Page
Data

页布局



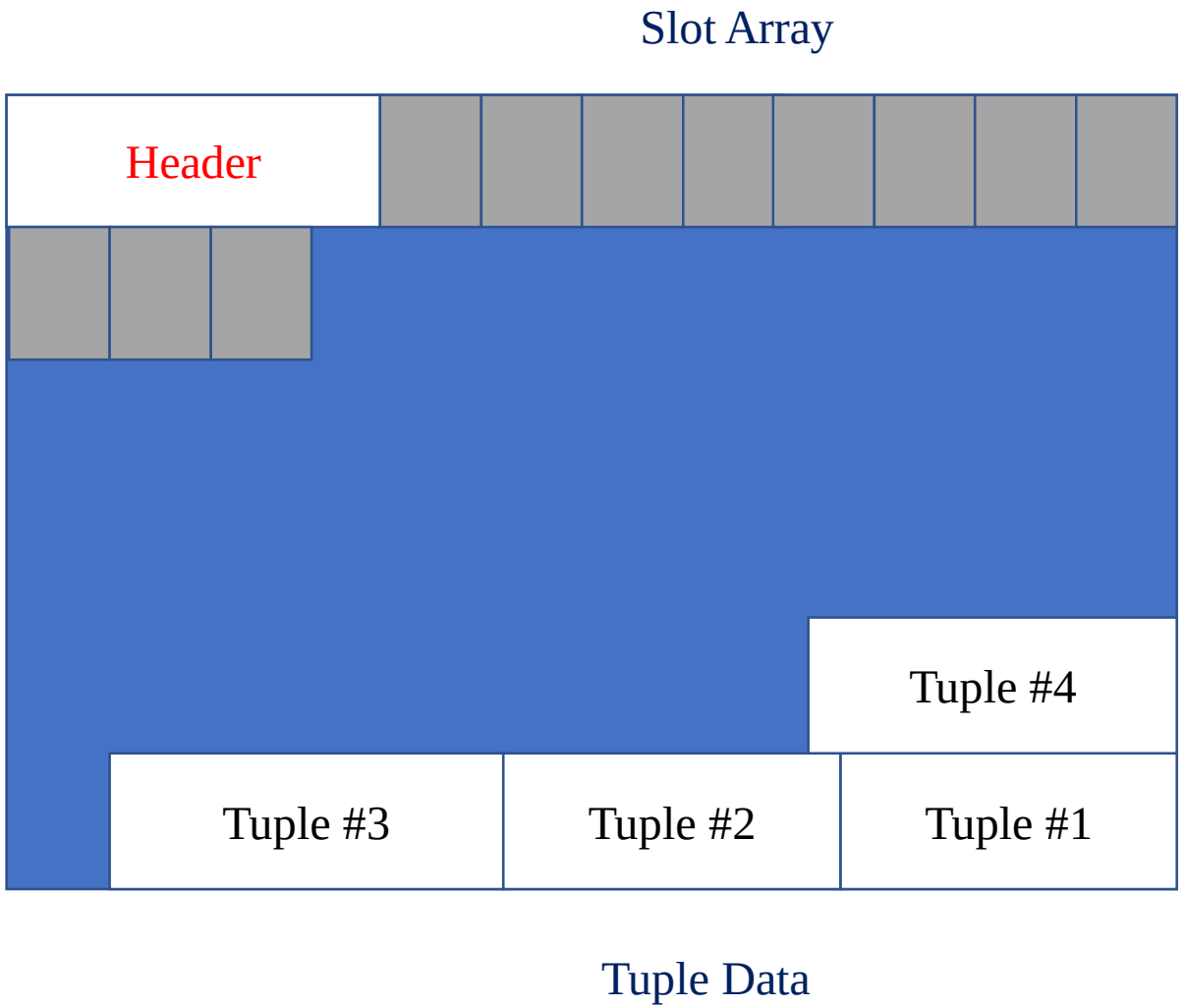
元数据

- 页面大小
- **校**验和
- **数据**库管理系统版本
- **事**务可见性
- 压缩信息
- **是否自包含**

槽页

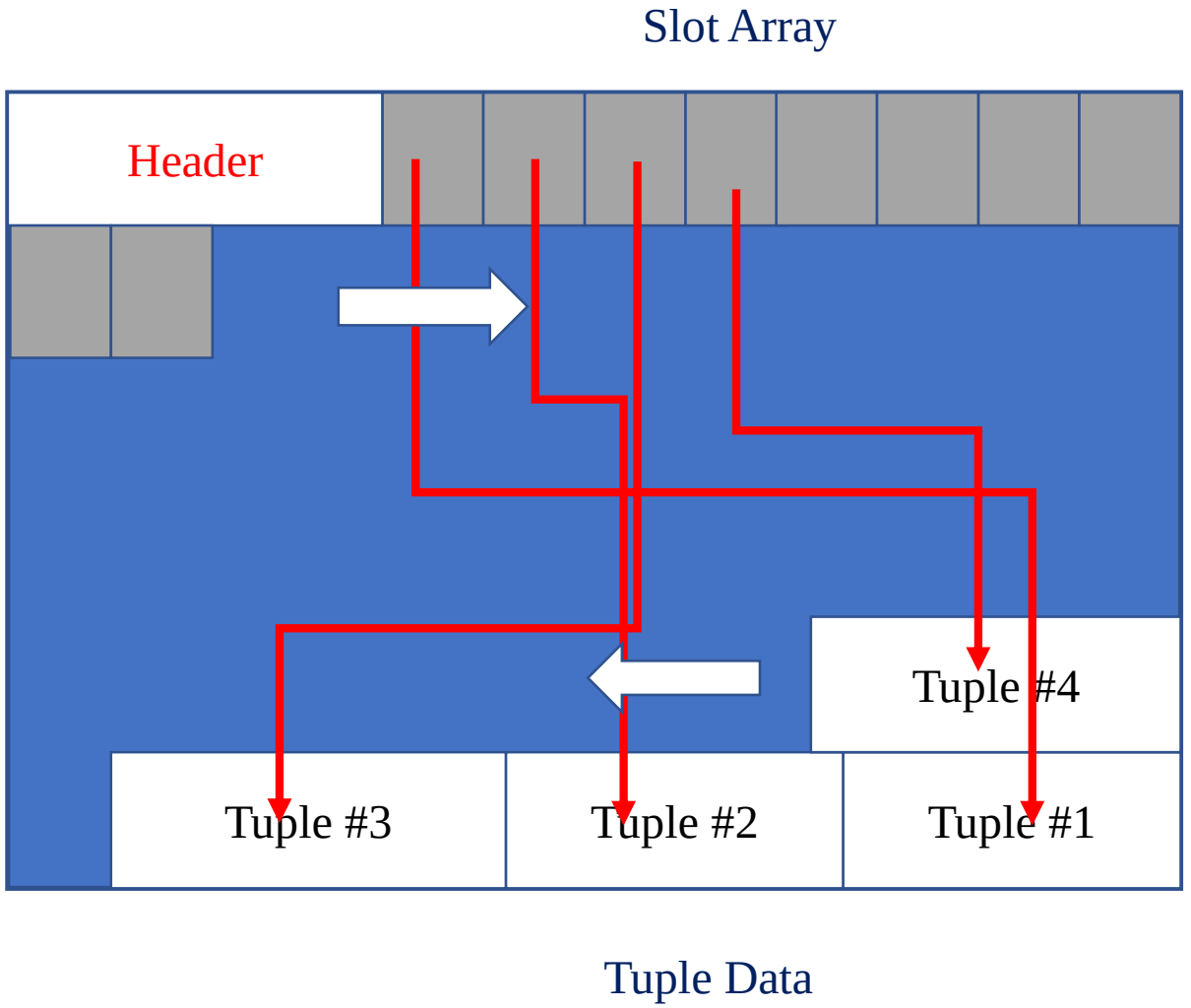
slot 数组将“slots”**映射到元**组的起**始位置偏移量。**

头记录使用槽的数量、最后使用的**槽的起始位置和一个跟踪了每个槽的起始位置的槽数**组。



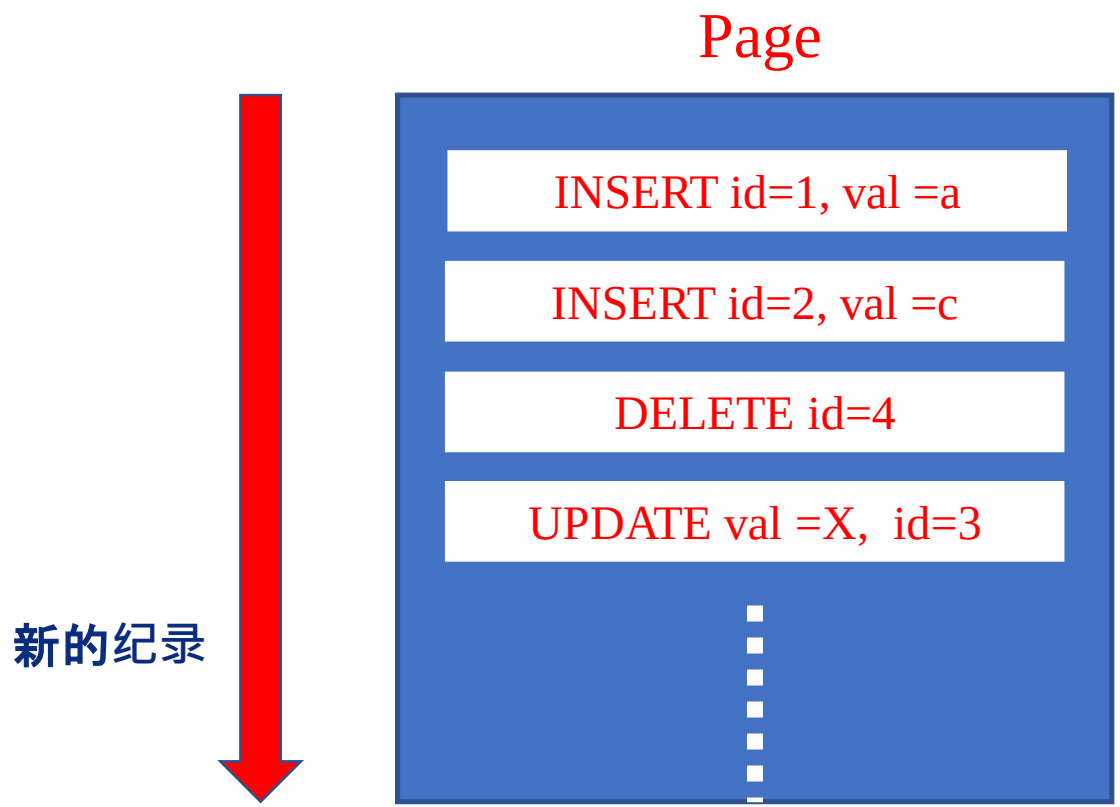
槽页

当插入一个元组的时候，槽数组将从前往后插入，元组数据**将从后往前插入。当槽数**组和**元**组数据相遇的时候说明当前页已满。



日志结构

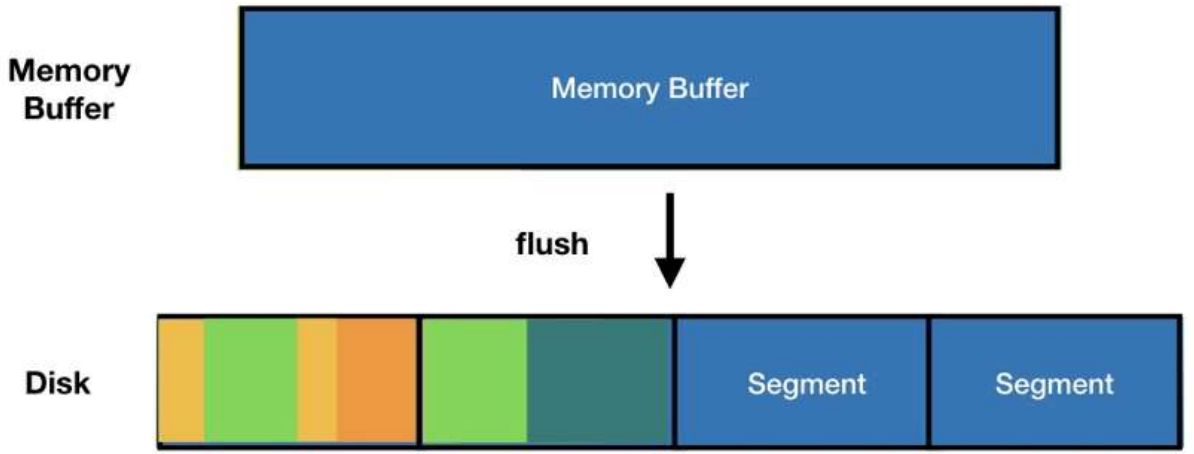
- DBMS **只存**储日志记录，而不**是在**页面中存储元组。
- **系**统将日志记录附加到数据库**修改方式的文件中**：
  - **插入存**储整个元组。
  - 删除将元组标记为已删除。
  - **更新**仅包含被修改的属性的增量。



日志结构

在 LFS 中，**空余空**间是用固定**大小的段(Segment)来管理的**：

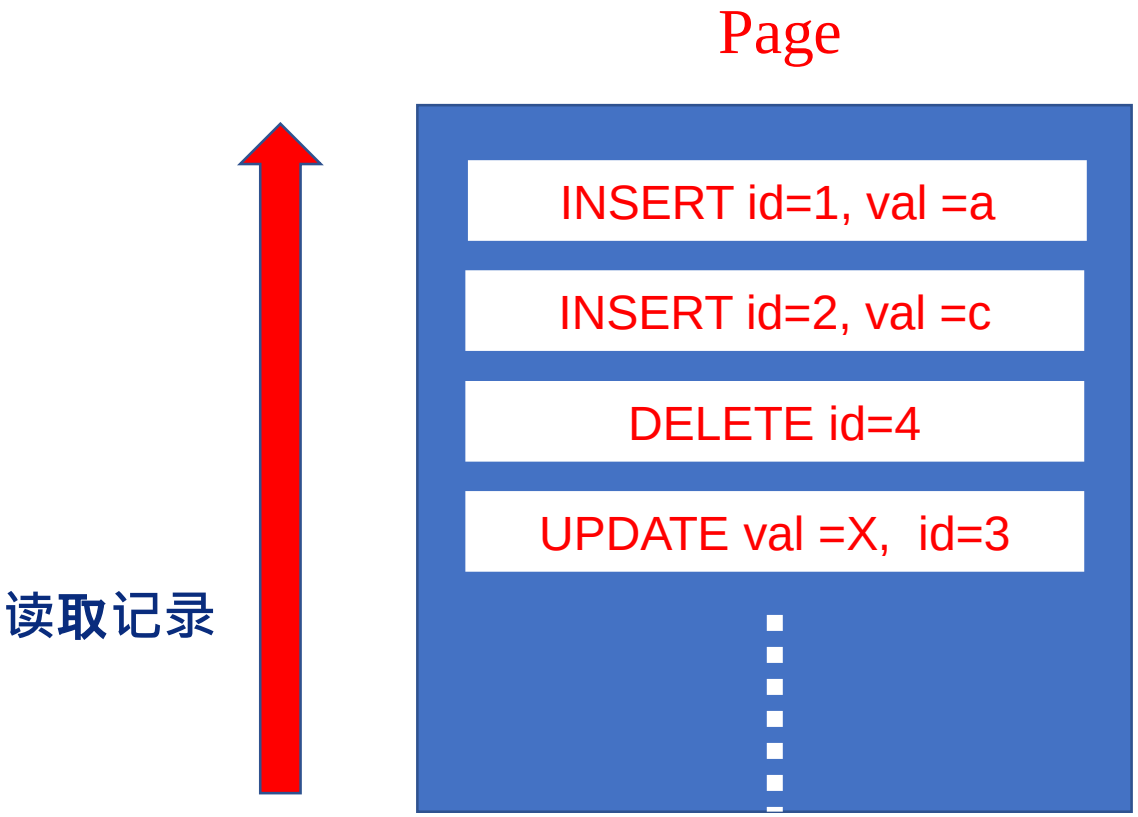
硬盘被分割成固定大小的段；写**操作首先会被写入到内存中**；**当内存中**缓存的数据超过段的大小**后，LFS 将数据一次性写入到空**闲的段中。



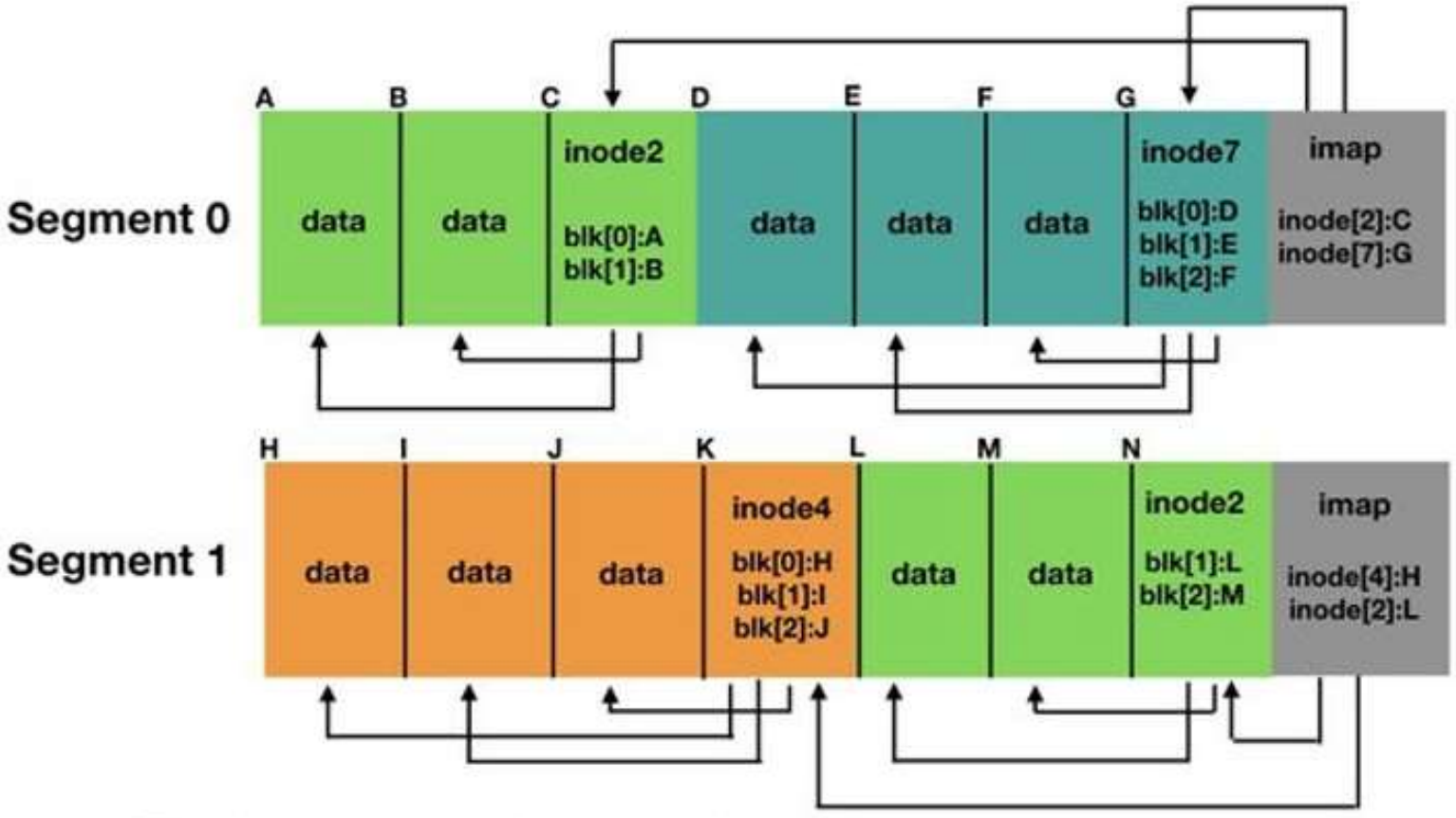
日志结构

为了读取记录，DBMS 向后扫描日志并“重新创建”元组以找到它需要的内容。

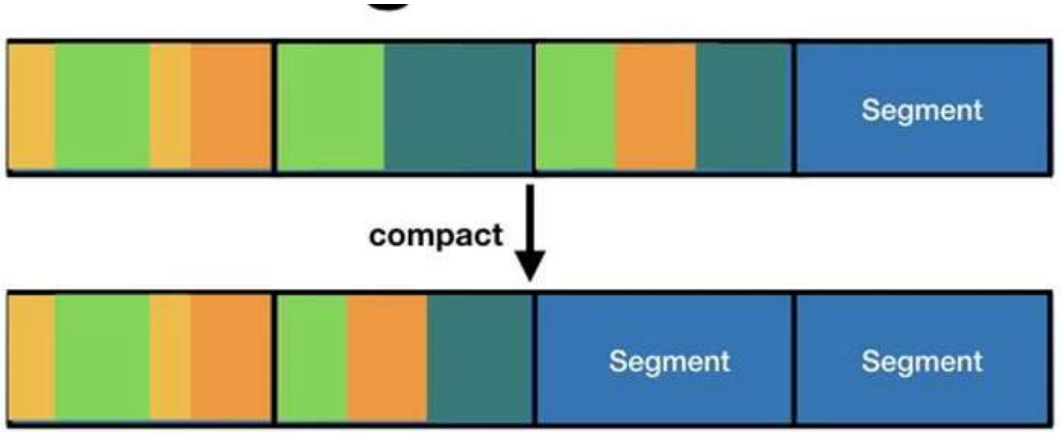
构建索引以允许它跳转到日志中的位置。

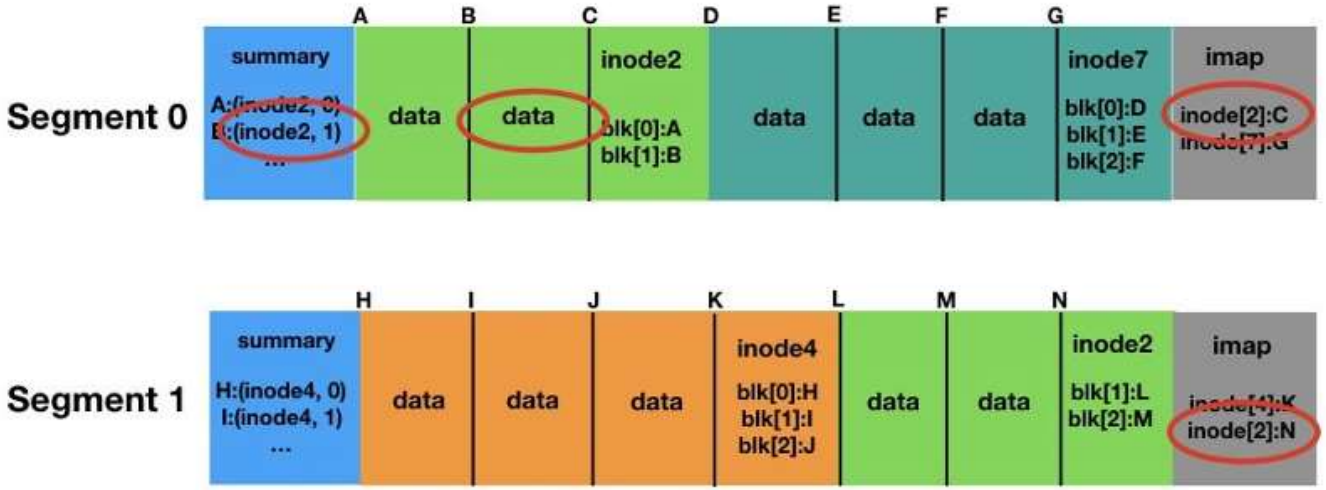


日志结构


A
B
C
D
E
F
G
Segment 0
data
data
inode2
blk[0]:A
blk[1]:B
data
data
data
inode7
blk[0]:D
blk[1]:E
blk[2]:F
imap
inode[2]:C
inode[7]:G
H
I
J
K
L
M
N
Segment 1
data
data
data
inode4
blk[0]:H
blk[1]:I
blk[2]:J
data
data
inode2
blk[1]:L
blk[2]:M
imap
inode[4]:H
inode[2]:L

日志结构


Segment
compact
Segment
Segment


A
B
C
D
E
F
G
Segment 0
summary
A:(inode2, 0)
B:(inode2, 1)
...
data
data
inode2
blk[0]:A
blk[1]:B
data
data
data
inode7
blk[0]:D
blk[1]:E
blk[2]:F
imap
inode[2]:C
inode[7]:G
H
I
J
K
L
M
N
Segment 1
summary
H:(inode4, 0)
I:(inode4, 1)
...
data
data
data
inode4
blk[0]:H
blk[1]:I
blk[2]:J
data
data
inode2
blk[1]:L
blk[2]:M
imap
inode[4]:K
inode[2]:N

元组

- **一个元**组的本质是一连串的字节。数据库将这些字节解释成属性类型和相应的值。

元组

- **Tuple Header（元组头部）**：包含元组的元数据

  一些关于数据库并发控制的信息（例如哪个事务创建/修改这个元组）。

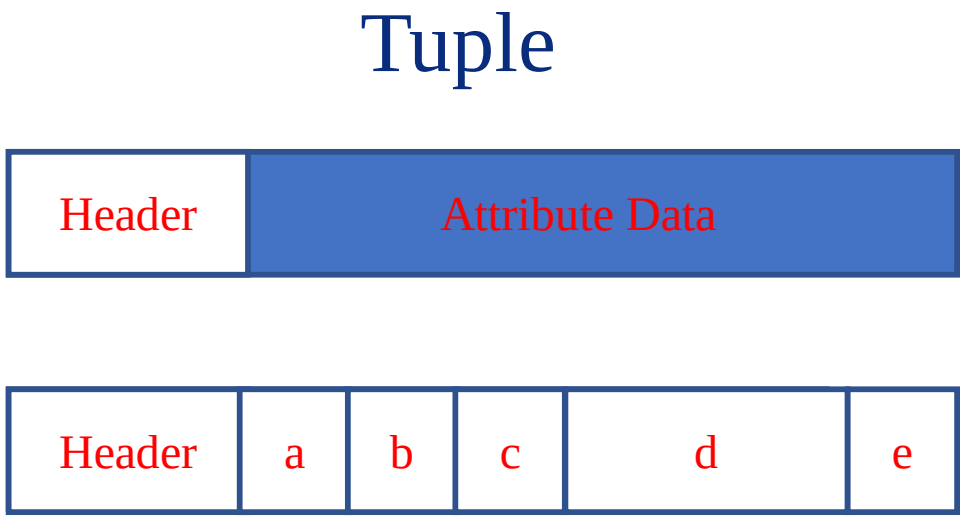
  标记NULL的位图（bitmap）。

- 数据库不需要在这里存储数据库模式的元数据。

- **Tuple Data（元组数据）**：每个属性真实的数据。

- 属性通常按照你创建表的时候的顺序存储。

- 大多数数据库不允许一个元组超过页的大小。

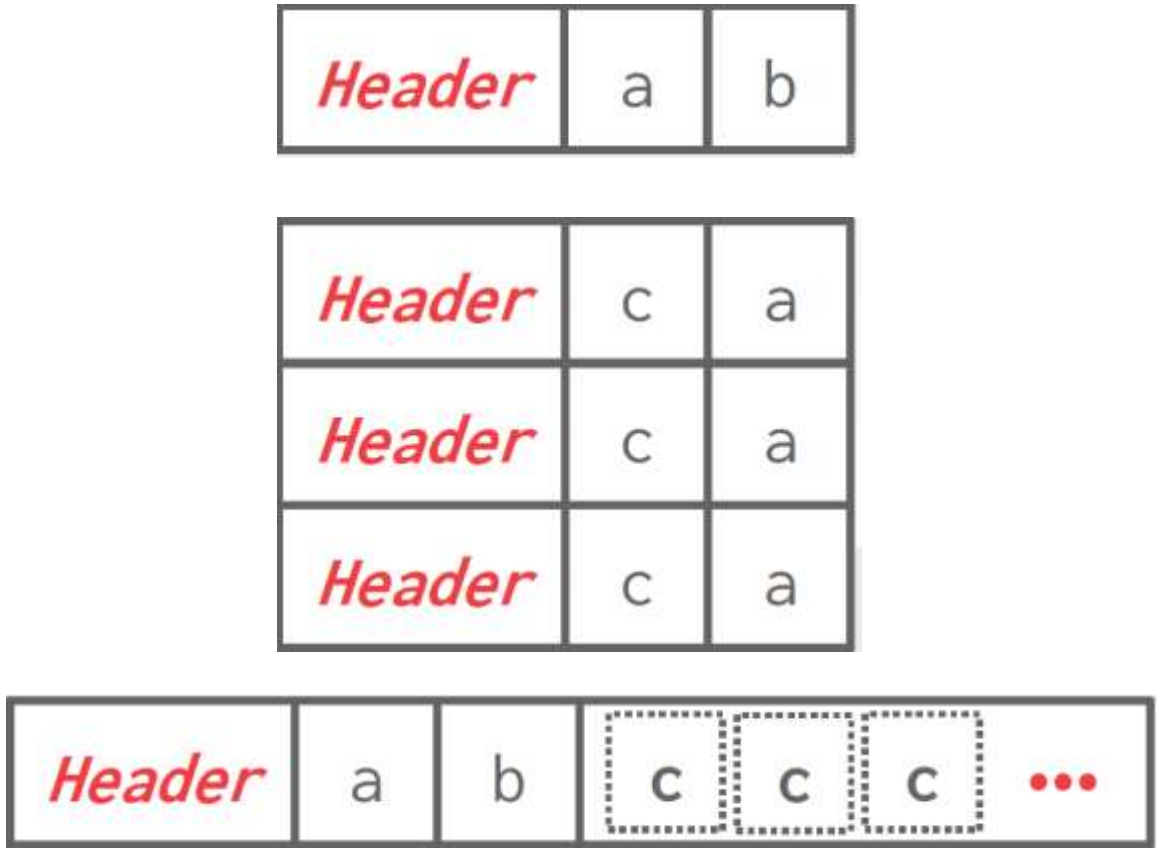


规范化的元组数据

CnosDB

```
CREATE TABLE foo (
a INT PRIMARYKEY,
b INT NOT NULL,
);

CREATE TABLE bar (
c INT PRIMARYKEY,
a INT
REFERENCES foo (a),
);
```



CnosDB

Q&A

Scan CC WeChat to Join the Community

添加CC好友，接受进群邀请

Welcome to follow the GitHub repo

欢迎关注我们的代码仓库

https://github.com/**cnosdatabase/cnosdb**

GitHub