CnosDB

数据库系统 | Database Systems

第四课：存储–2

Lecture 4: **Storage-2**

Lecturer: Harbour

Date: 2021.11.29

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

00

本节大纲

- 页的存储结构
- **堆文件**
- 页布局
- **日志**结构
- **元**组

页的存储结构

- 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

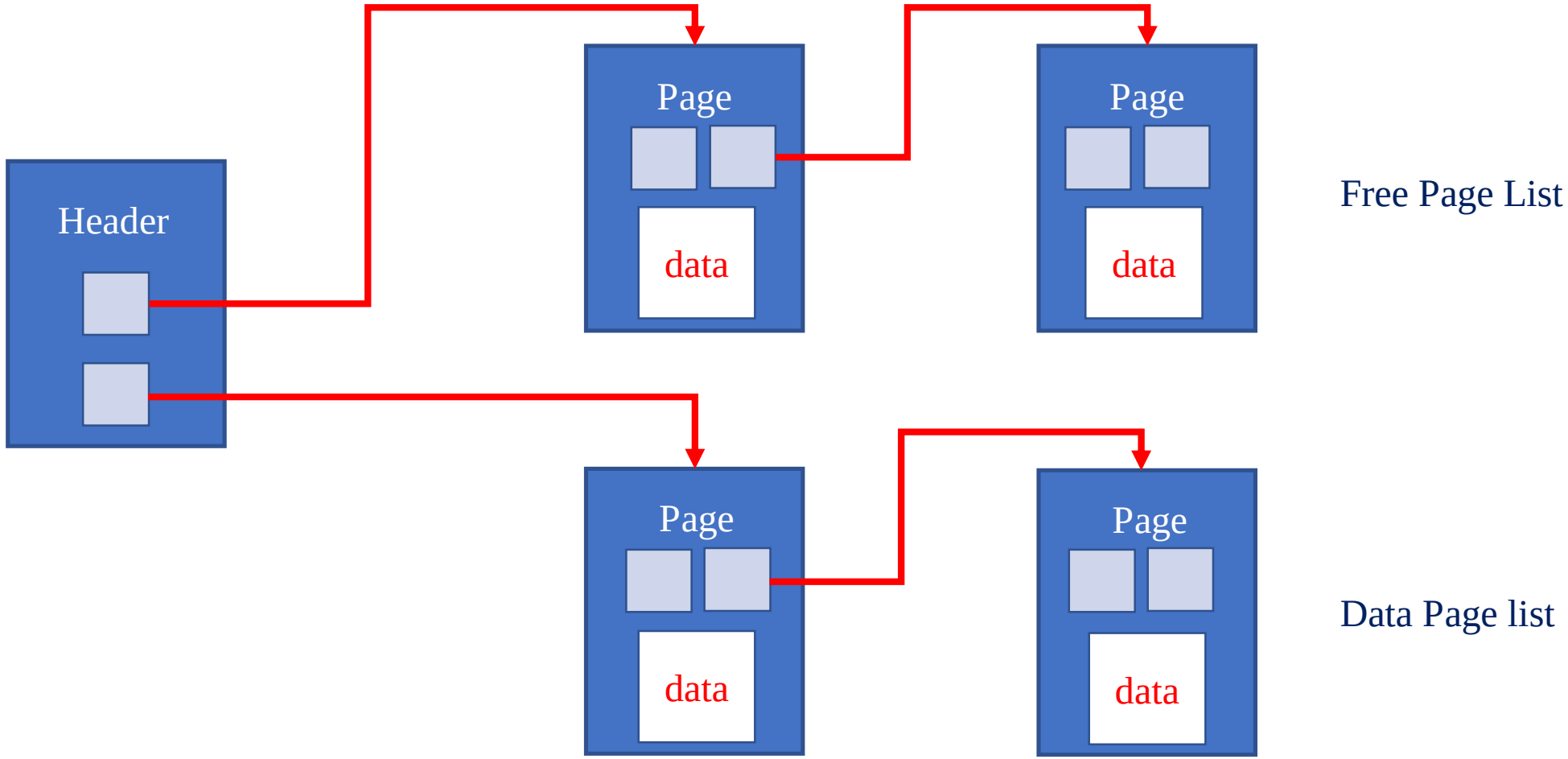
- **堆文件是一个无序的**页面集合，其中元组以随机顺序存储。

  CGWD Pages

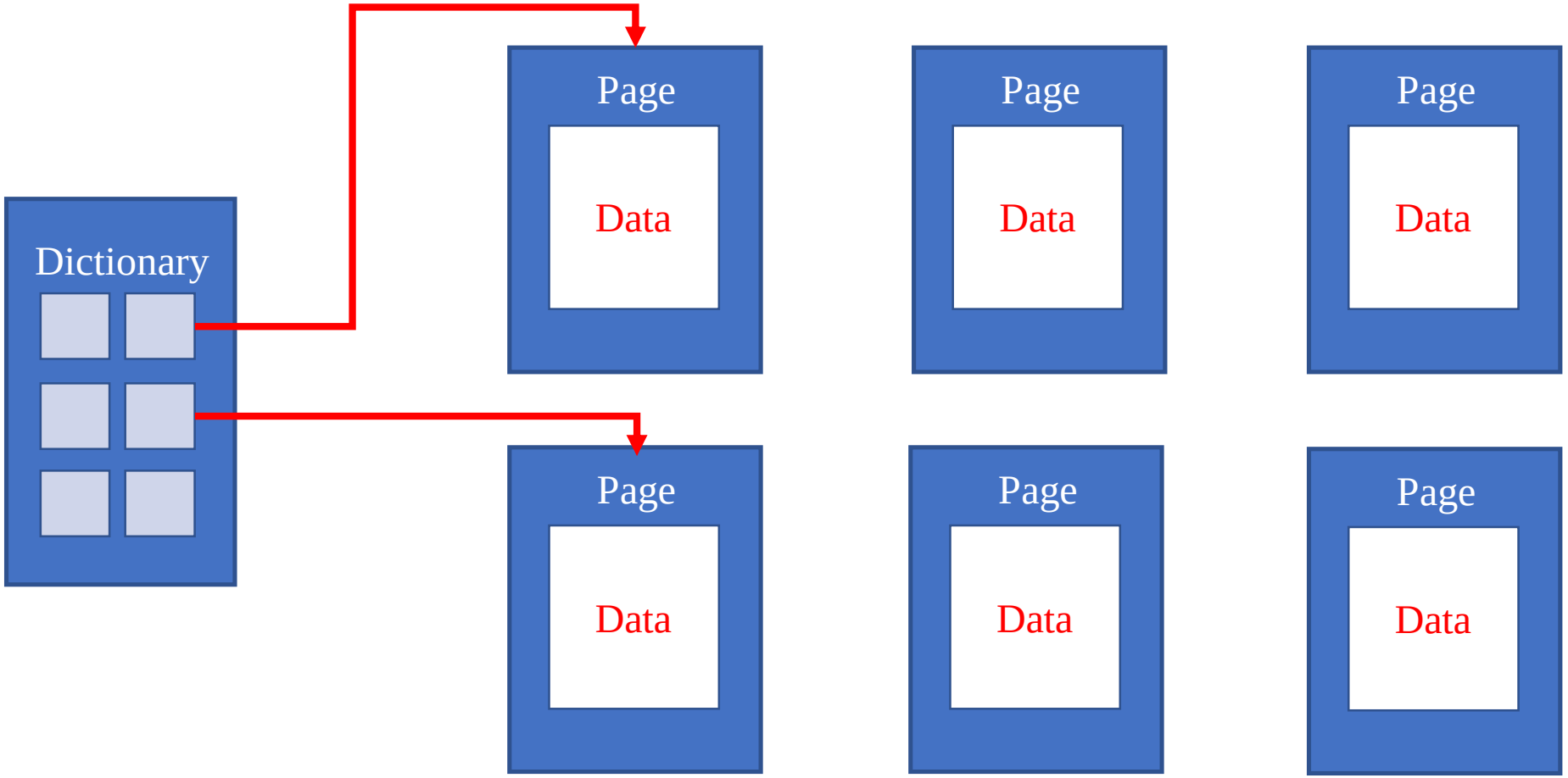
  支持遍历所有的Pages

- **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。

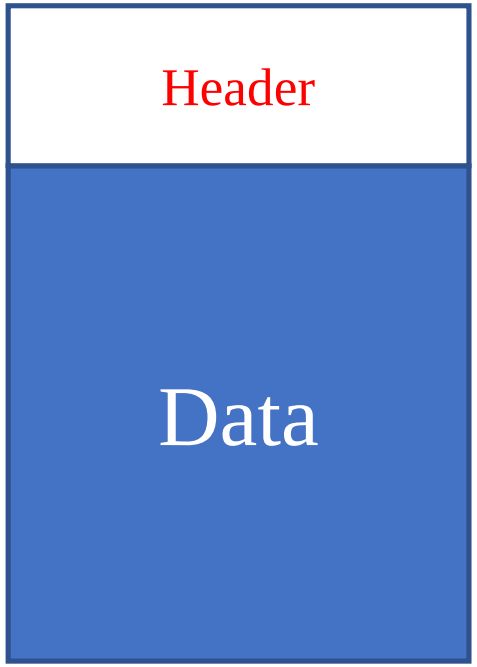
堆文件-链表


Header
Page
data
Page
data
Page
data
Page
data
Free Page List
Data Page list

堆文件-页字典


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

页布局



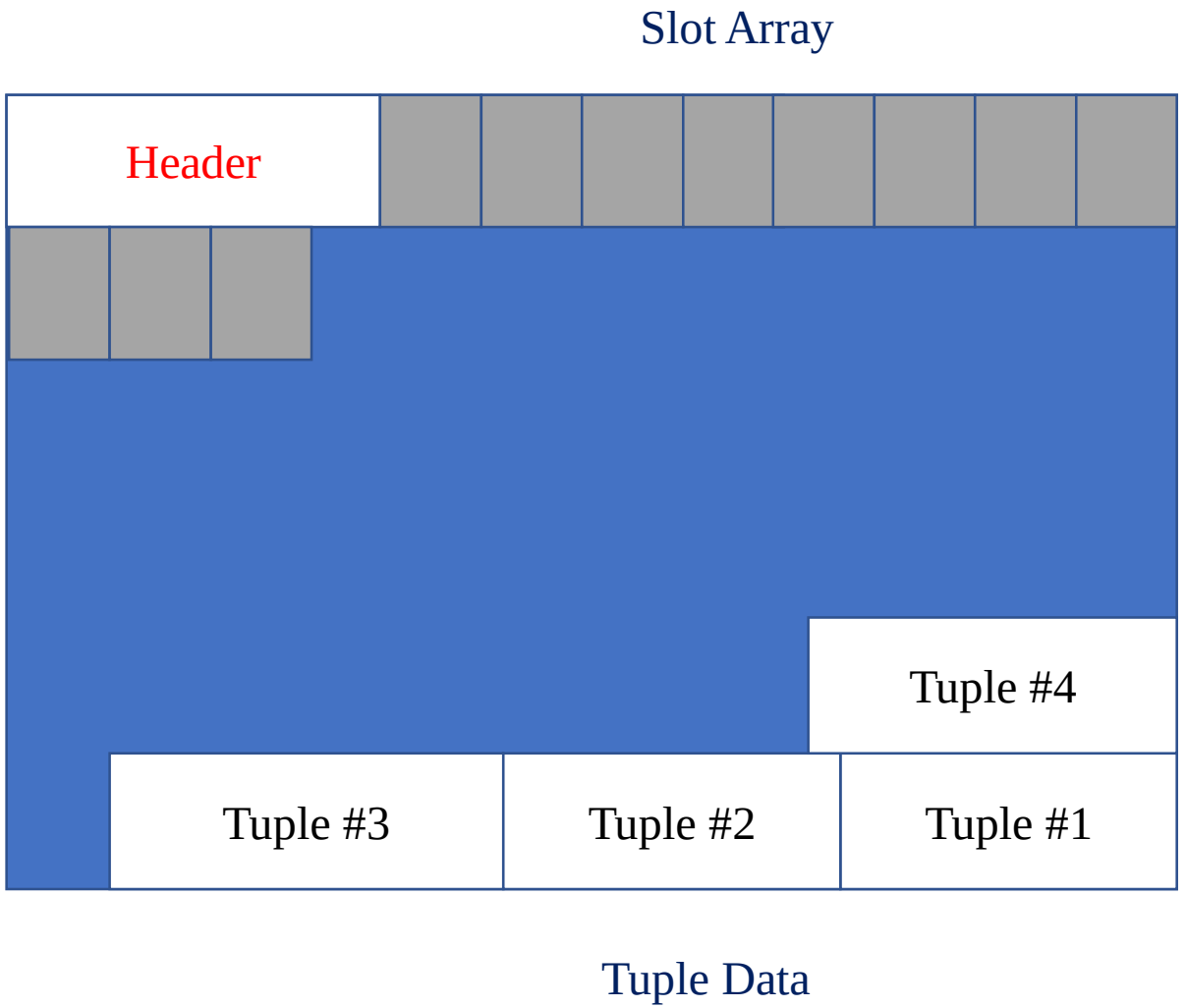
元数据

- 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**

页布局-槽页

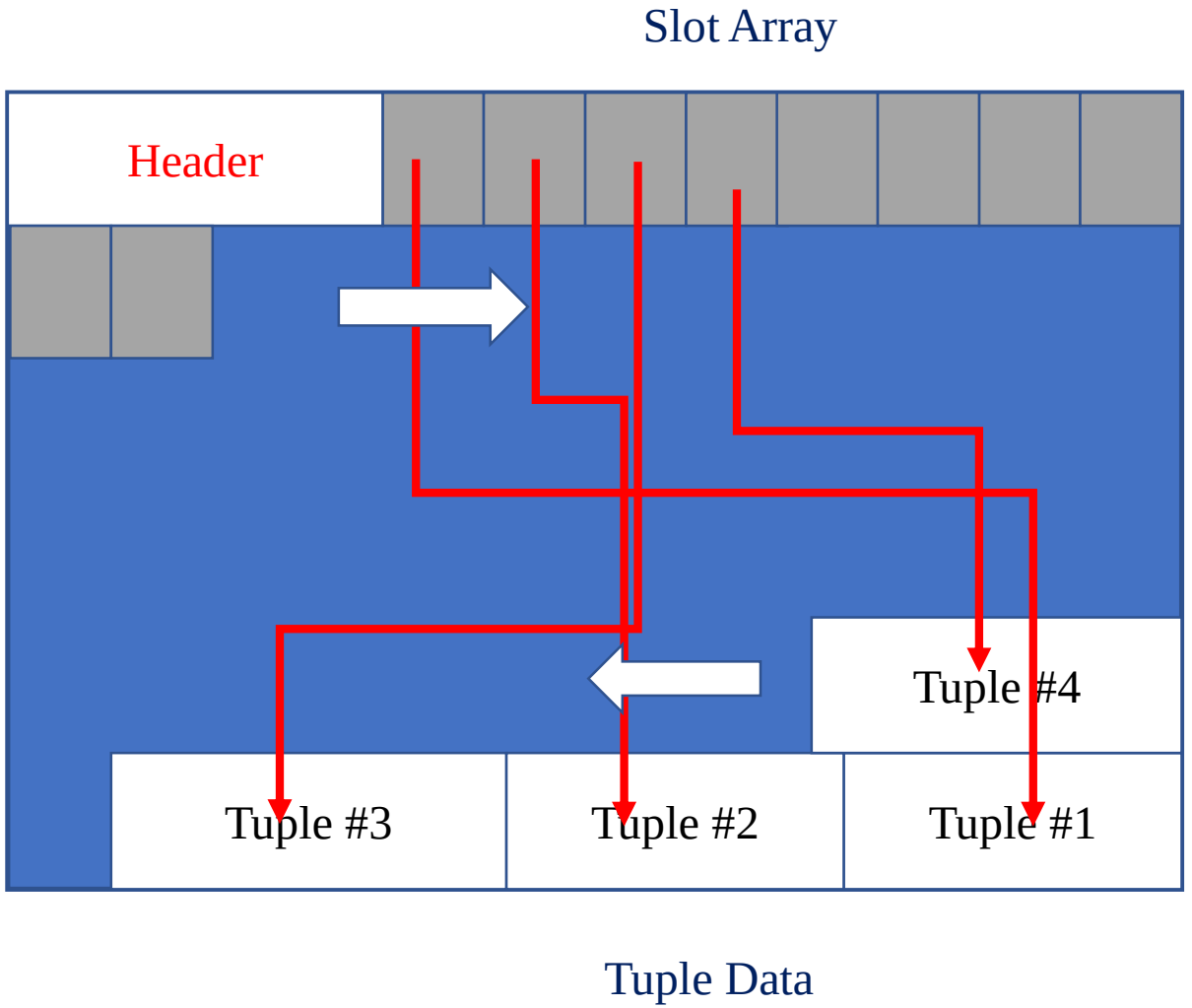
slot 数组将“slots”**映射到元**组的起**始位置偏移量。**

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。



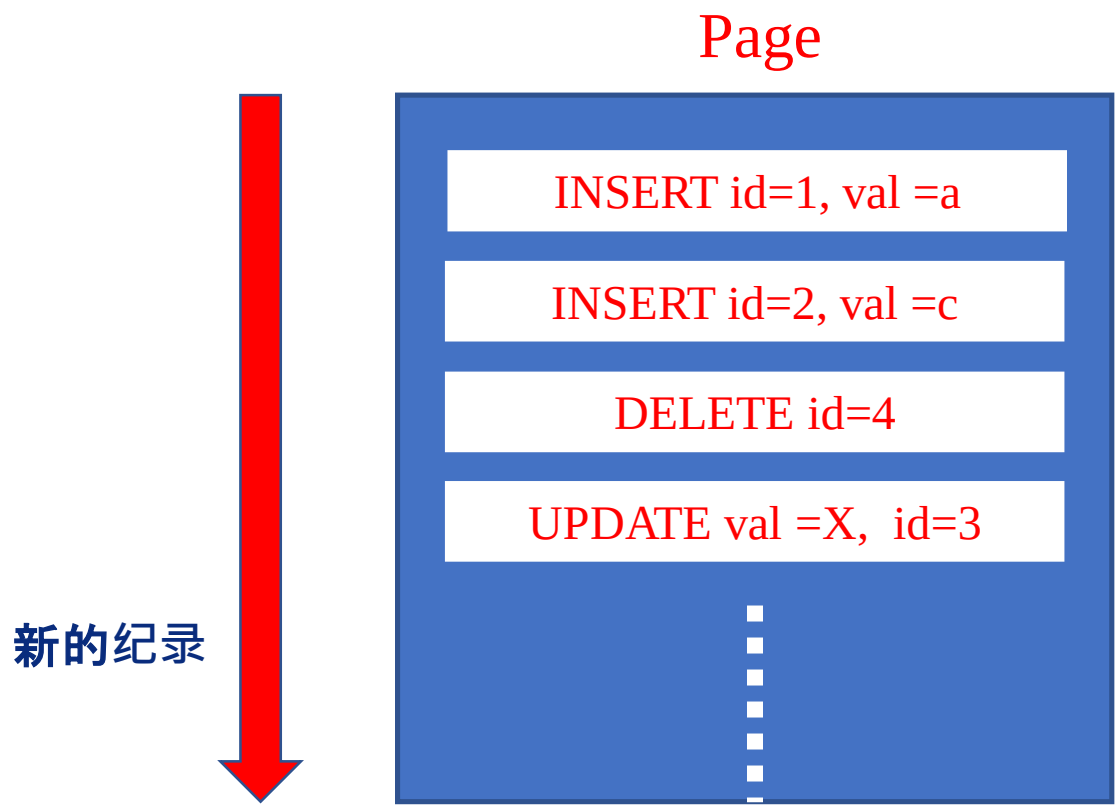
页布局-槽页

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

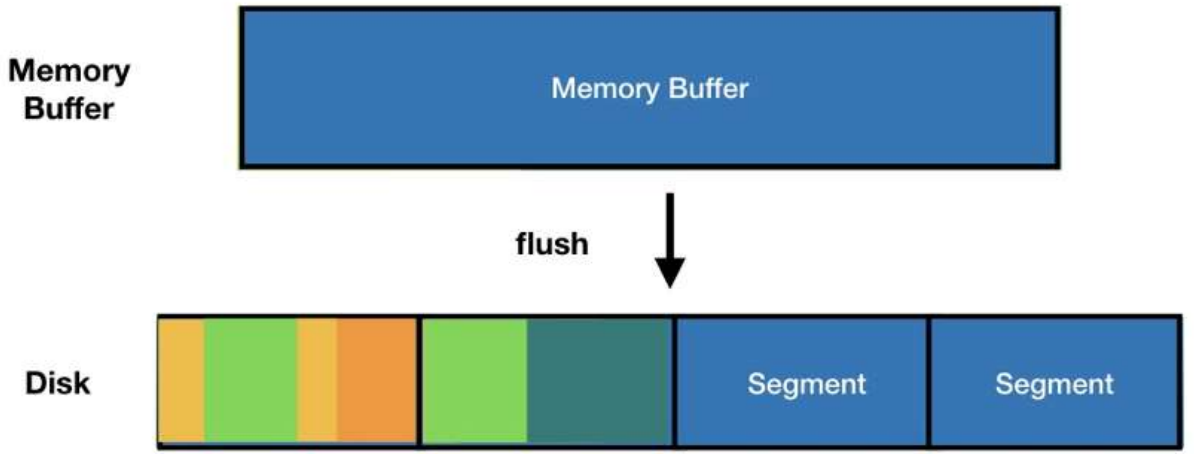
- DBMS **只存**储日志记录，而不**是在**页面中存储元组。
- **系**统将日志记录附加到数据库**修改方式的文件中**：
  - **插入存**储整个元组。
  - 删除将元组标记为已删除。
  - **更新**仅包含被修改的属性的增量。



日志结构

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

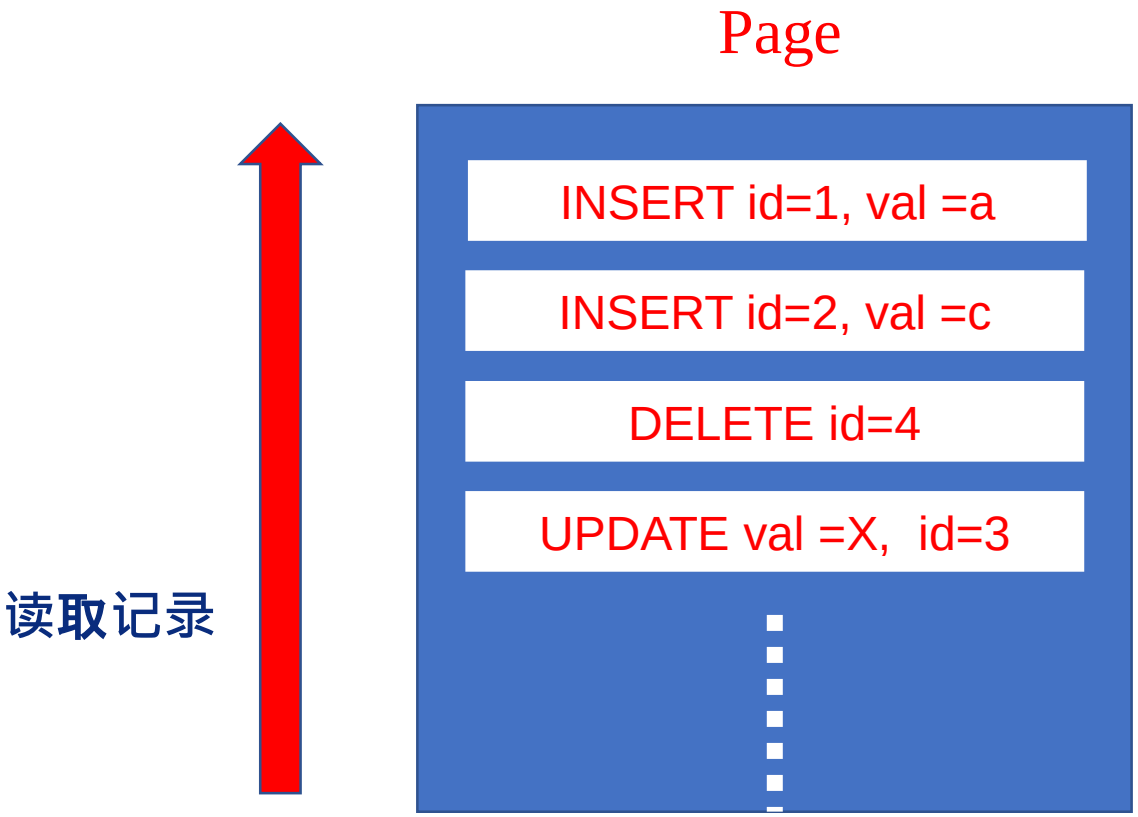
硬盘被分割成固定大小的段；写**操作首先会被写入到内存中**；**当内存中**缓存的数据超过段的大小**后，LFS 将数据一次性写入到空**闲的段中。



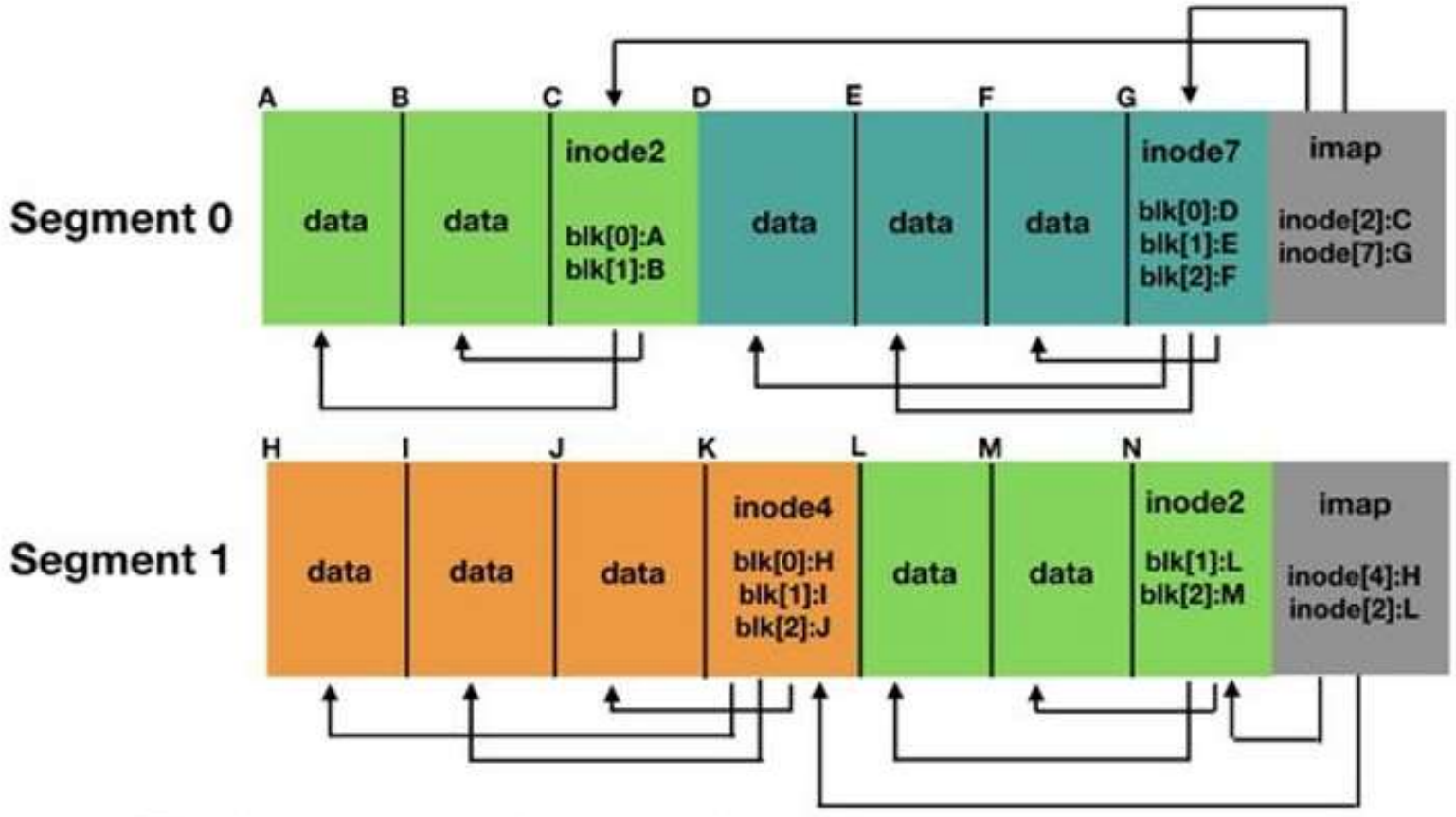
日志结构

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。

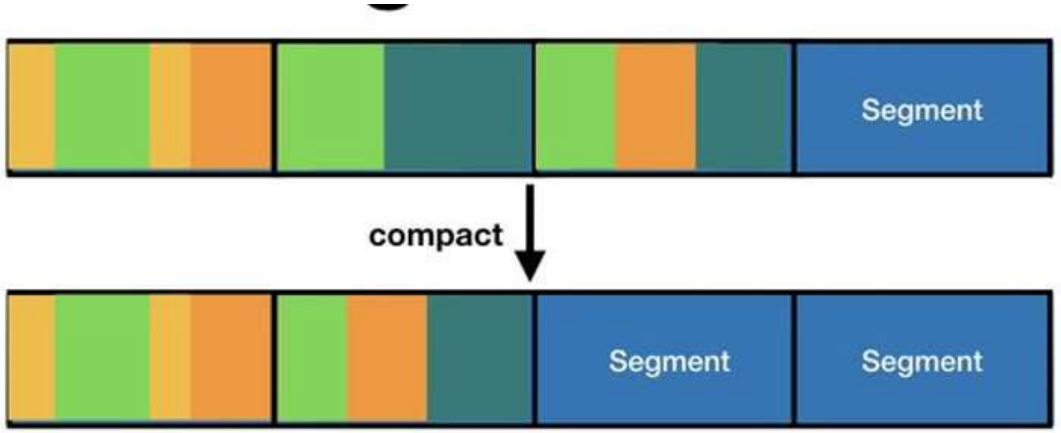
构建索引以允许它跳转到日志中的位置。

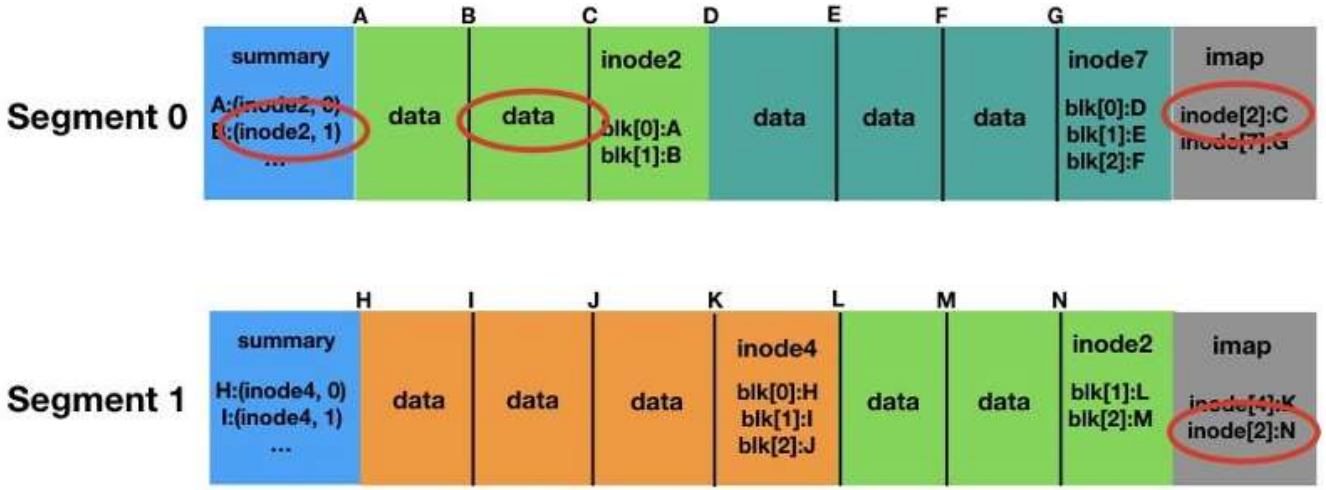


日志结构


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

CnosDB

- **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）。

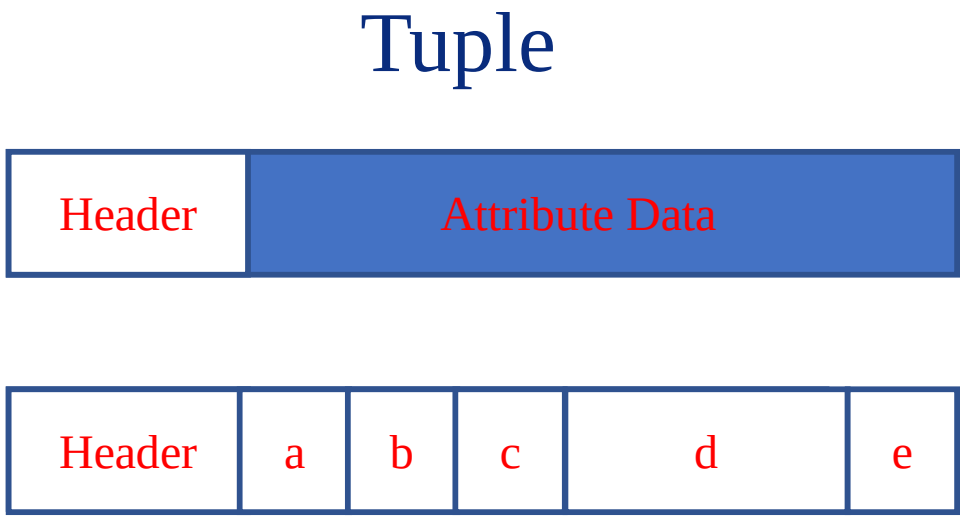
  标记NULL的位图（bitmap）。

- 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。

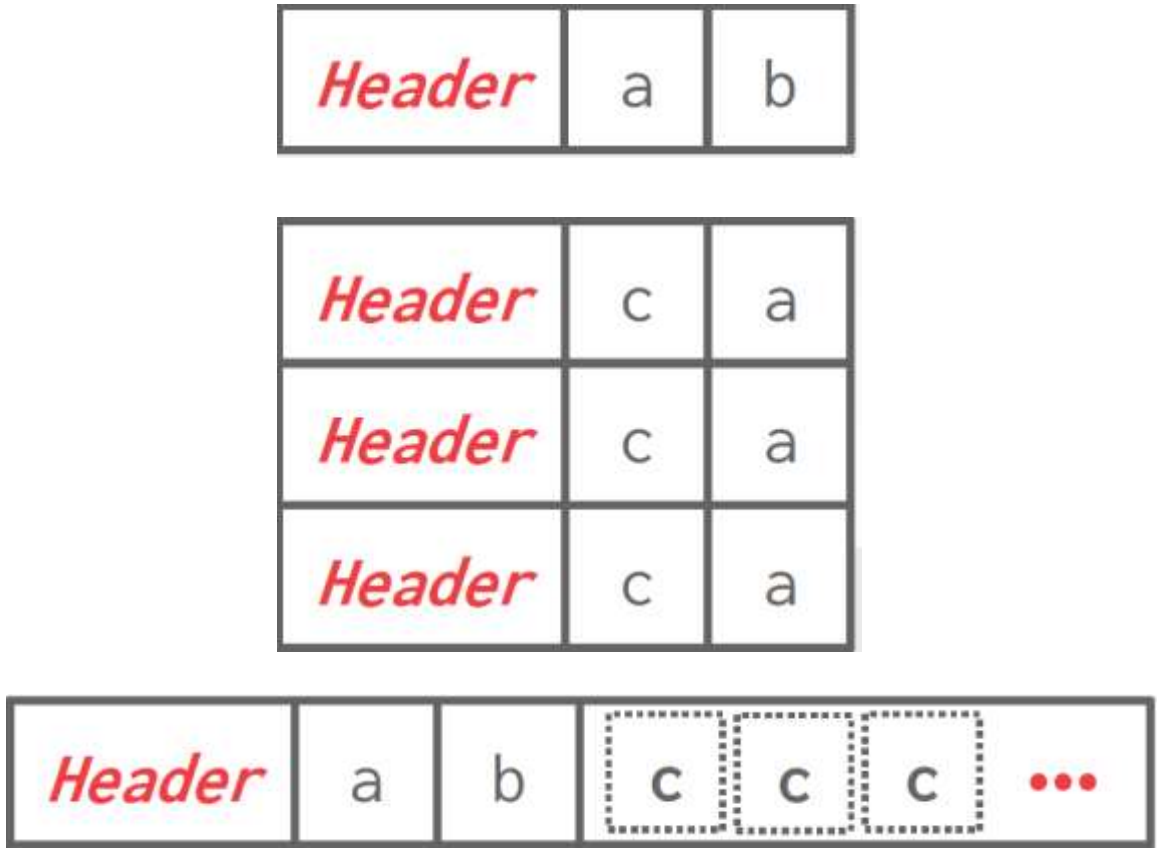


元组-规范化的元组数据

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