Chapter 7: Storage Management

Zhaonian Zou

Massive Data Computing Research Center
School of Computer Science and Technology
Harbin Institute of Technology, China
Email: znzou@hit.edu.cn

Spring 2020

Outline[1]

1. Storage Media
2. Representation of Databases on Disks
   - Value Representation
   - Tuple Layout
   - Page Layout
     - Tuple-Oriented Page Layout
     - Log-Structured Page Layout
   - File Organization
3. System Catalogs
4. Buffer Management

1存储介质

2磁盘上数据库的表示形式
价值表示
元组布局
页面布局
面向元组的页面布局
日志结构页面布局
文件组织

3系统目录

4电池管理

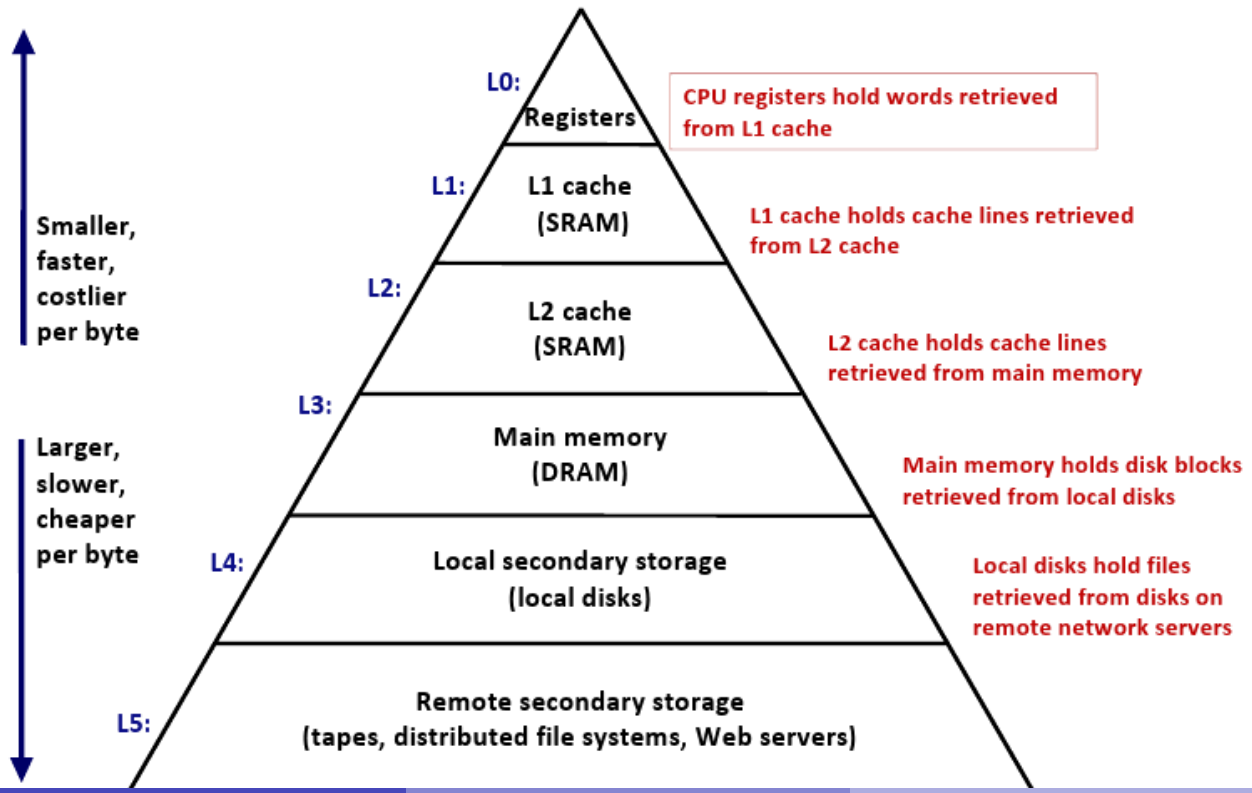
[1]Updated on March 24, 2020

Storage Media 存储介质

The Memory Hierarchy (存储层级)

内存层次结构（X®Bß）
计算机系统中的内存按层次结构排列

Memory in a computer system is arranged in a *hierarchy*



Primary Storage (主存储器)

CPU可以使用加载/存储直接操作主存储中的数据

Data in primary storage can be operated on directly by CPUs using `load/store`

- Registers (寄存器)
- Cache (高速缓存)
- Main memory (内存)

主存储

Primary storage is byte-addressable (按字节寻址)

Secondary Storage (二级存储器)

二级存储器中的数据不能直接由CPU处理； 必须首先使用读/写将其复制到主存储中

Data in secondary storage cannot be processed directly by CPUs; It must first be copied into primary storage using `read/write`

- Magnetic disks (磁盘)/Hard disk drives (HDD, 机械硬盘)
- Flash memory (闪存)/Solid state drives (SSD, 固态硬盘)

Secondary storage is block-addressable (按块寻址) and online (联机)

Tertiary Storage (三级存储器)

第三存储中的数据必须首先复制到第二存储中

Data in tertiary storage must first be copied into secondary storage

- Magnetic tape (磁带)
- Optical disks (光盘)
- Network Storage (网络存储)

三次存储

Tertiary storage is block-addressable and offline (脱机)

Access Time (访存时间)

Access time (ns)	Storage media
0.5	L1 cache
7	L2 cache
100	DRAM
150,000	SSD
10,000,000	HDD
30,000,000	Network storage
1,000,000,000	Tape

各级之间的数据传输

Data Transfer Between Levels

- Cache
  ↕ Unit: cache lines (缓存行), size: 64B
- Main memory
  ↕ Unit: blocks (块)/pages (页), size: 512B–16KB
- Secondary storage
  ↕ Unit: blocks/pages, size: 512B–16KB
- Tertiary storage

存储媒体类别

Categories of Storage Media

重新启动计算机时（关闭或崩溃后），易失性存储中的数据会丢失
主存储

Volatile Storage (易失存储器)

Data in volatile storage is lost when the computer is restarted (after a shutdown or a crash)

- Primary storage

重新启动计算机后，非易失性存储中的数据将保留

Non-volatile Storage (非易失存储器)

Data in non-volatile storage is retained when the computer is restarted

- Secondary storage
- Tertiary storage

非易失性主存储器正在兴起！

Non-volatile main memory is emerging!

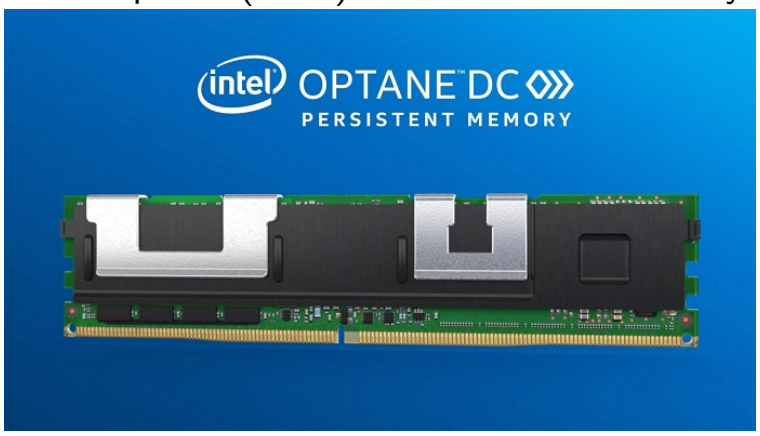
Non-volatile Main Memory (NVM, 非易失主存)

NVM is also known as persistent memory or storage-class memory (SCM)

- Byte-addressable
- Non-volatile (persistent)

NVM也称为持久性内存或存储类内存（SCM）
字节寻址
非易失性（持久）

Intel Optane (傲腾) DC Persistent Memory

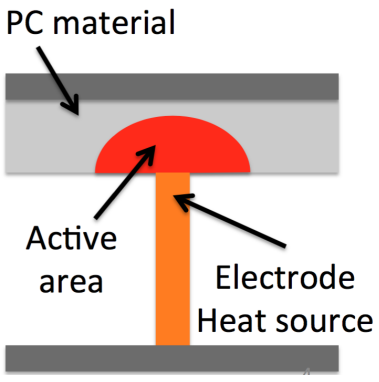


Persistent Memory Developing Toolkit (PMDK) (`http://pmem.io`)

Phase-Change Memory (PCM, 相变存储器)

- PCM stores data using phase-change material (相变材料)
  - Amorphous phase (非晶态): high resistivity $\longrightarrow$ 0
  - Crystalline phase (晶态): low resistivity $\longrightarrow$ 1
- Set phase via current pulse
  - Fast cooling $\rightarrow$ Amorphous
  - Slow cooling $\rightarrow$ Crystalline

通过电流脉冲设置相位
快冷却！ 非晶态
慢冷却！ 结晶



Characteristics of NVM

NVM的特征

Characteristics	DRAM	PCM	Memristor	MRAM	SSD	HDD
Volatility	Yes	No	No	No	No	No
Addressability	Byte	Byte	Byte	Byte	Block	Block
Volume	GB	TB	TB	TB	TB	TB
Read latency	60ns	50ns	100ns	20ns	25μs	10ms
Write latency	60ns	150ns	100ns	20ns	300μs	10ms
Energy per bit	2pJ	2pJ	100pJ	0.02pJ	10nJ	0.1J
Endurance	10^{16}	10^{10}	10^{8}	10^{15}	10^{5}	10^{16}

NVM将成为内存层次结构中的关键级别，并在计算机系统和DBMS中扮演重要角色

NVM will become a critical level in the memory hierarchy and plays important roles in computer systems and DBMS

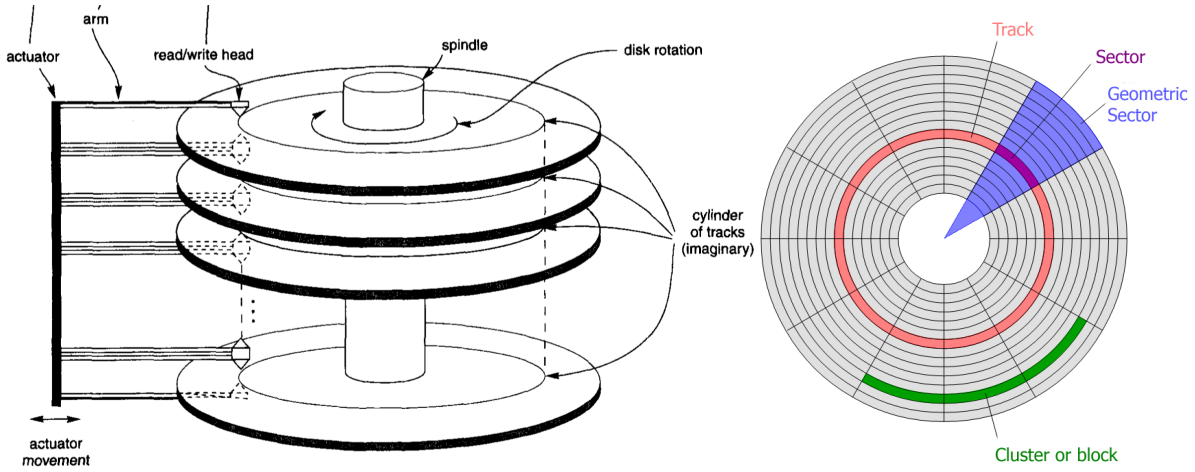
Magnetic Disks (磁盘)

磁盘由两个部分组成：

A magnetic disk is composed by two components:

磁盘装配：
磁头组件：

- Disk assembly: sectors (扇区) $\subset$ tracks (磁道) $\subset$ cylinders (柱面)
- Head assembly: disk heads (磁头) and disk arms (磁臂)



磁盘上的每个磁道都分为较小的扇区
·将磁道划分为扇区是硬编码在磁盘表面上的，无法更改
行业组织
·方法1：部门对向固定角度
·方法2：扇区保持统一的记录密度

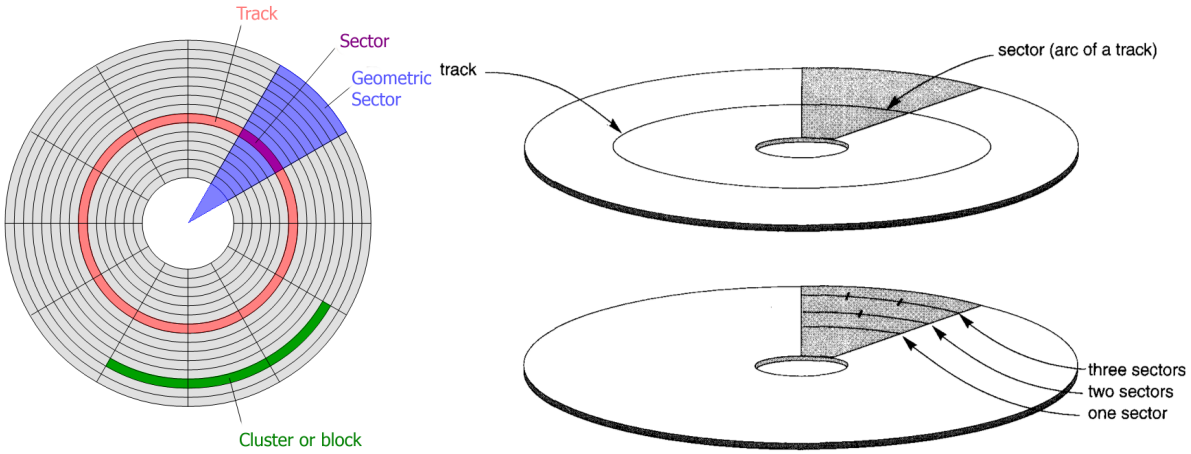
Sectors (扇区)

Every track on disks is divided into smaller sectors

- The division of a track into sectors is hard-coded on the disk surface and cannot be changed

Sector organizations

- Approach #1: Sectors subtend a fixed angle
- Approach #2: Sectors maintain a uniform recording density



Disk Blocks (磁盘块)/Pages (页)

操作系统在格式化磁盘时将磁道分成大小相等的磁盘块（或页面）
·磁盘块的大小可以设置为扇区大小的倍数
·块大小是在磁盘格式化期间固定的，不能动态更改
·典型的磁盘块大小为512B–4KB

The OS divides a track into equal-sized disk blocks (or pages) during disk formatting

- The size of a disk block can be set as a multiple of the sector size
- Block size is fixed during disk formatting and cannot be changed dynamically
- Typical disk block size is 512B–4KB

磁盘块的LBA是0到n-1之间的数字（假设磁盘的总容量为n个块）

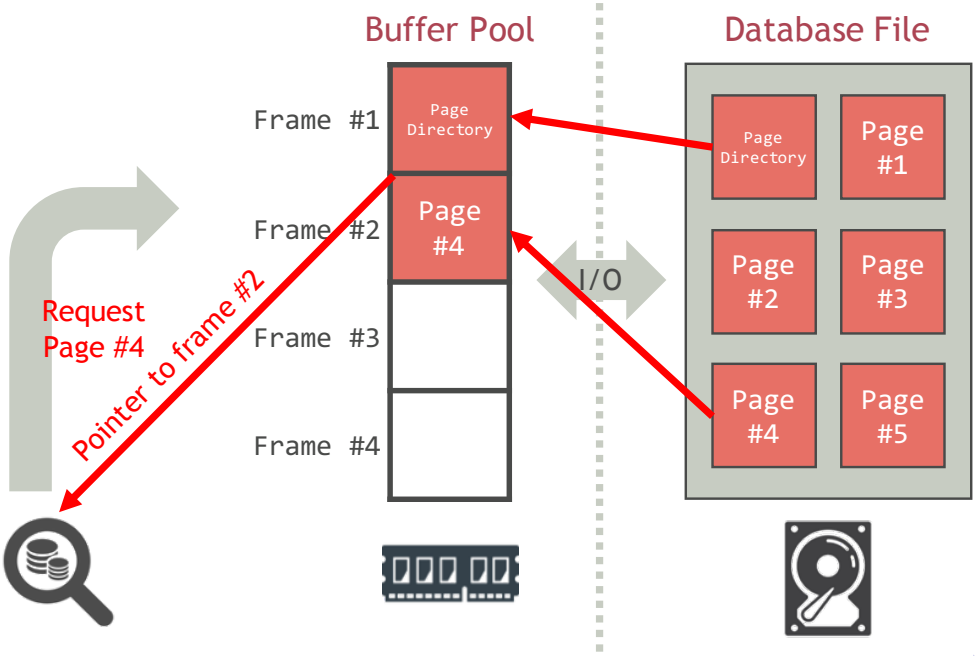
Logical Block Address (LBA, 逻辑块地址)

The LBA of a disk block is a number between 0 and $n-1$ (assuming the total capacity of the disk is n blocks)

面向磁盘的数据库存储
·数据永久存储在辅助存储中
·要读取或修改数据时将数据加载到主存储器中（但尚未在主存储器中找到）

Disk-Oriented Database Storage

- Data is persistently stored on secondary storage
- Data is loaded into main memory when it is going to be read or modified (but not yet found in main memory)



Disk-Oriented

面向磁盘的数据库存储（续）
设计目标
允许DBMS管理超出可用内存量的数据库
读/写磁盘非常昂贵，因此必须仔细管理，以避免大量停顿和性能下降

Design Goals

- Allow the DBMS to manage databases that exceed the amount of memory available
- Reading/writing to disk is expensive, so it must be managed carefully to a[illegible]

在磁盘上写页面的位置
目标是使经常一起使用的页面在磁盘上在物理上尽可能地靠近在一起（数据局部性）

Spatial Control (空间方面的控制)

- Where to write pages on disk
- The goal is to keep pages that are used together often as physically close together as possible on disk (data locality)

Temporal Control (时间方面的控制)

- When to read pages into memory, and when to write them to disk
- The goal is minimize the number of stalls from having to read data from disk

何时将页面读取到内存中以及何时将它们写入磁盘
目的是最大程度地减少必须从磁盘读取数据的停顿数

磁盘上数据库的表示形式

Representation of Databases on Disks

磁盘上数据库的表示形式
价值表示

Representation of Databases on Disks
Value Representation

Representation of Numbers

INT/INTEGER/BIGINT/SMALLINT/TINYINT (整数)

- C/C++ representation

FLOAT/DOUBLE/REAL (浮点数)

- IEEE-754 standard
- Typically faster than arbitrary precision numbers, but can have

通常比任意精度的数字快，但可能会有舍入误差

任意精度和规模
当舍入错误不可接受时使用
通常以精确的，可变长度的二进制表示形式存储，并带有其他元数据（如VARCHAR，但不存储为字符串）

NUMERIC/DECIMAL (定点数)

- Arbitrary precision and scale
- Used when rounding errors are unacceptable
- Typically stored in an exact, variable-length binary representation with additional meta-data (like a VARCHAR but not stored as a string)

Representation of Strings

CHAR(n)/BINARY(n) (定长字符串/字节串)

- an array of n bytes
- If the length of a string is shorter than n, the array is padded with a special null character
- Example: 'cat' is represented as | 'c' | 'a' | 't' | 0 | 0 | in CHAR(5)

如果字符串的长度小于n，则使用特殊的空字符填充数组

VARCHAR/VARBINARY/TEXT/BLOB (变长字符串/字节串)

- Header with length, followed by data bytes
- Example: 'cat' is represented as | 3 | 'c' | 'a' | 't' | in VARCHAR(5)
- C/C++ null-terminated string representation is not used

带长度的标题，后跟数据字节

不使用以null终止的字符串表示形式

BIT(n) (定长位串)

- an array of $\lceil n/8 \rceil$ bytes
- Example: BIT(12), 010111110011 → | 01011111 | | 00110000 |

Representation of Other Types of Values

TIME/DATE/DATETIME/TIMESTAMP（时间/日期）

- 32/64-bit integer of (micro)seconds since Unix epoch

自Unix时代以来的32/64位整数（微秒）

ENUM（枚举型值）

具有n个项目的枚举类型表示为BIT（dlog2 ne）

- An enumerated type with n items is represented as BIT($\lceil \log_2 n \rceil$)
- Example: ENUM(RED, GREEN, BLUE, YELLOW), RED $\rightarrow$ 00, GREEN $\rightarrow$ 01, BLUE $\rightarrow$ 10, YELLOW $\rightarrow$ 11

磁盘上数据库的表示形式
元组布局

Representation of Databases on Disks
Tuple Layout

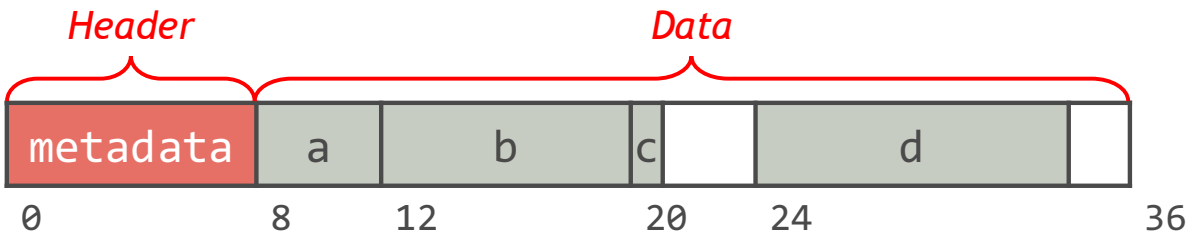
Tuple Layout

元组布局

元组本质上是字节序列
·将这些字节解释为属性类型和值是DBMS的工作
·DBMS的目录包含有关DBMS用于确定元组布局的表的模式信息

A tuple (元组) is essentially a sequence of bytes

- It is the job of the DBMS to interpret those bytes into attribute types and values
- The DBMS's catalogs (目录) contain the schema information about tables that the DBMS uses to figure out the tuple's layout

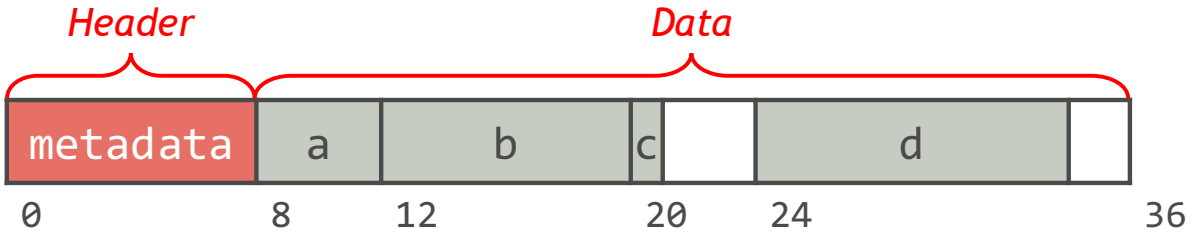


Tuple Header (头部)

每个元组都有一个标头，该标头包含有关它的元数据
·指向DBMS在哪里存储这种类型的元组的模式的指针
·元组的长度
·可见性信息（并发控制）
·用于NULL值的位图

Each tuple is prefixed with a header that contains meta-data about it

- A pointer to where the DBMS stores the schema for this type of tuple
- The length of the tuple
- Visibility info (concurrency control)
- A bitmap for `NULL` values



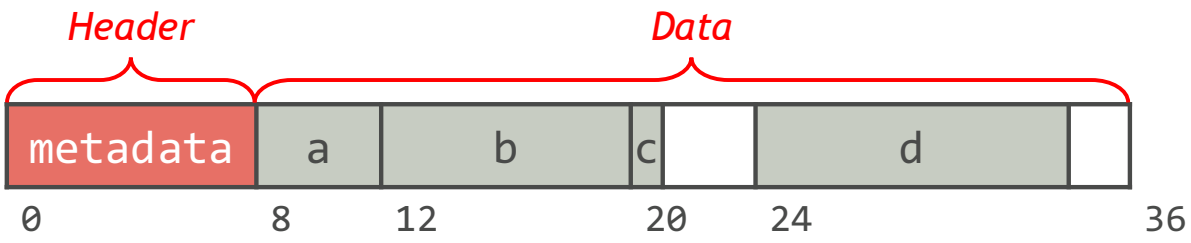
元组数据
元组数据部分将元组的所有属性值连接在一起
·属性通常按照创建表时指定的顺序存储
·每个属性值从元组的开头对齐一个偏移量，为4字节/ 8字节的倍数

Tuple Data

The tuple data part concatenates all attribute values of the tuple

- Attributes are typically stored in the order that you specify when you create the table
- Each attribute value aligns at an offset from the beginning of the tuple by a multiple of 4 bytes/8 bytes

```
CREATE TABLE Foo (
  a INT NOT NULL,
  b BIGINT NOT NULL,
  c CHAR NOT NULL,
  d VARCHAR(10) NOT NULL);
```



Representation of Databases on Disks
Page Layout

磁盘上数据库的表示形式
页面布局

Database Pages (数据库页面)

数据库页面

页面是固定大小的数据块（512B–16KB）
·它可以包含元组，元数据，索引，日志记录
·大多数DBMS不会混合页面类型
·一些DBMS要求页面是独立的

·每个页面都有一个唯一的标识符作为其页面ID
·DBMS使用间接层将页面ID映射到物理位置

A page is a fixed-size block of data (51

- It can contain tuples, meta-data, indexes, log records ...
- Most DBMSs do not mix page types
- Some DBMSs require a page to be self-contained

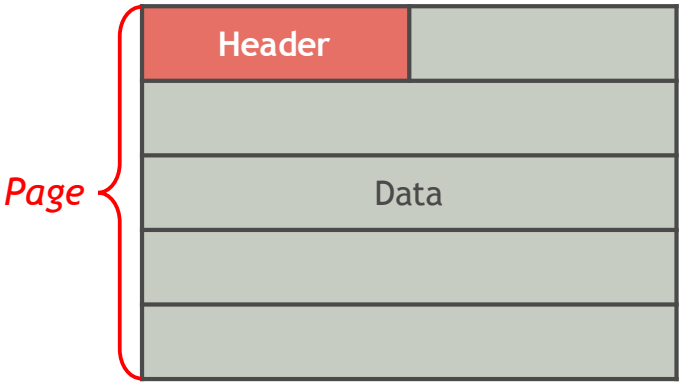
Each page is given a unique identifier as its page ID (页号)

- The DBMS uses an indirection layer to map page IDs to physical locations

Page Layout (页面布局)

页面由页面头部和页面数据组成

A page is comprised of the page header and the page data

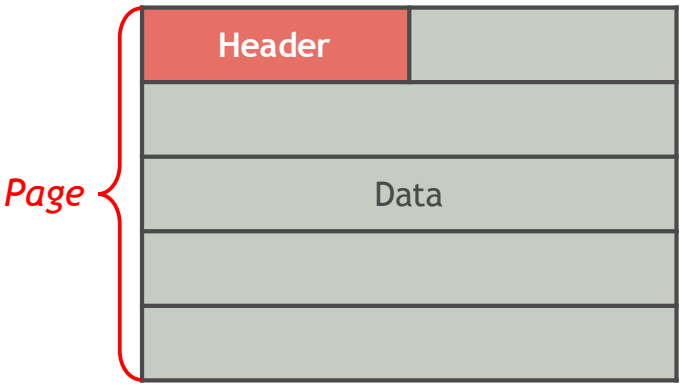


Page Header

Every page contains a header of metadata about the page's contents

- Page size
- Checksum (校验码)
- DBMS version
- Transaction visibility
- Compression information

页面标题
每个页面都包含有关页面内容的元数据标题
·页面大小
·校验码
·DBMS版本
·交易可见度
·压缩信息



Page Data

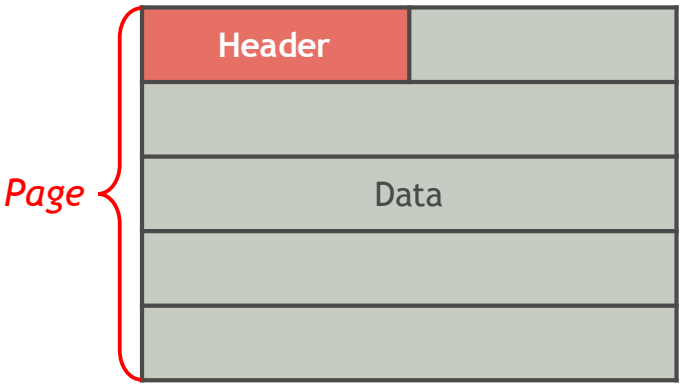
页面数据
在这里，我们仅考虑存储多个元组的页面

如何组织存储在页面内部的数据？
·方法1：以元组为导向
·方法2：日志结构

Here we only consider pages storing a number of tuples

How to organize the data stored inside of a page?

- Approach #1: Tuple-oriented
- Approach #2: Log-structured

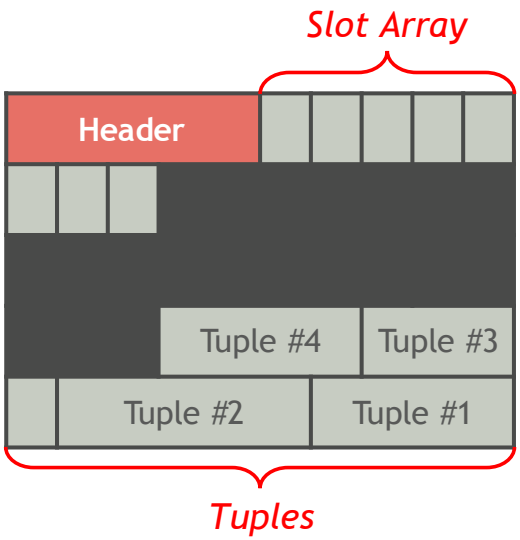


Tuple-Oriented Page Layout

面向元组的页面布局

最常见的页面布局方案称为分页页面（^˜ub）

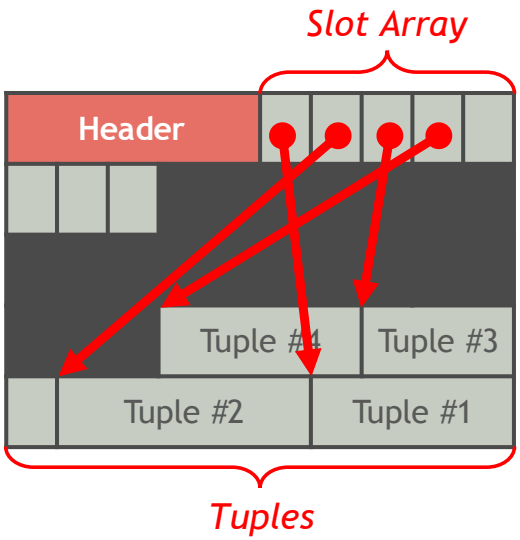
The most common page layout scheme is called slotted pages (分槽页面)



Slot Pages (分槽页面)

插槽数组将使用的“插槽”映射到元组的起始位置

The slot array maps the used “slots” to the tuples’ starting position offsets

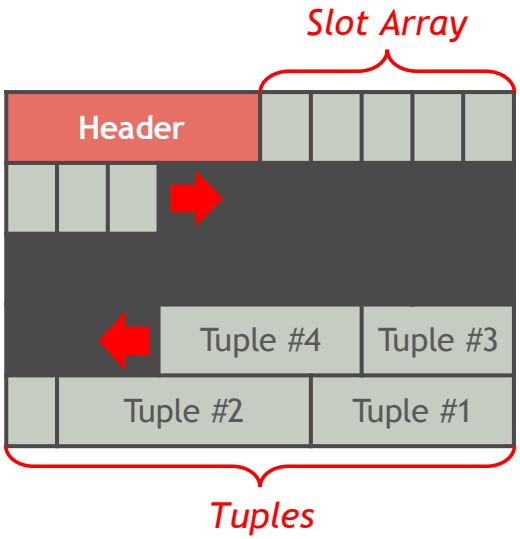


Slotted Pages (Cont'd)

版位页面的页眉跟踪
·已用插槽数
·最后使用的插槽的起始位置

The header of a slotted page keeps track of

- The # of used slots
- The offset of the starting location of the last slot used



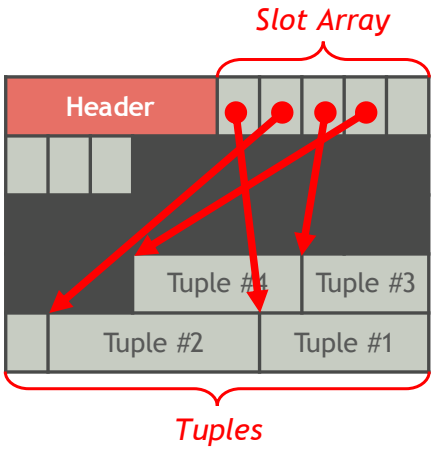
Record

记录ID（∞U∽）
为了跟踪单个元组，DBMS为每个元组分配一个唯一的记录标识符
最常见的记录ID是（PageID，Slot#）
·PageID：包含元组的页面的ID
·Slot#：元组在页面中存储的插槽号

To keep track of individual tuples, the DBMS assigns each tuple with a unique record identifier

The most common record ID is `(PageID, Slot#)`

- `PageID`: the ID of the page containing the tuple
- `Slot#`: the # of the slot where the tuple is stored in the page



Large Values

·大多数DBMS不允许元组超过单个页面的大小
·要存储大于页面的值，DBMS使用单独的溢出存储页面

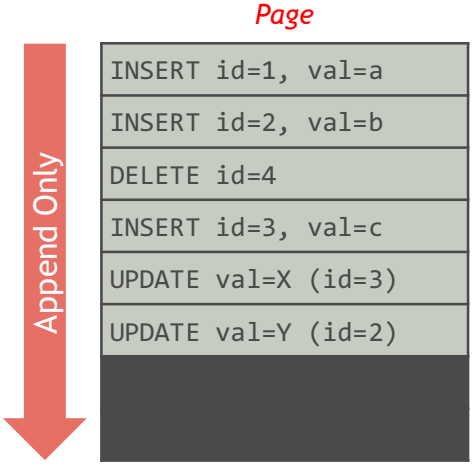
- Most DBMSs don't allow a tuple to exceed the size of a single page
- To store values that are larger than a page, the DBMS uses separate overflow storage pages

日志结构页面布局
DBMS不在页面中存储元组，而是仅将日志记录追加到如何修改数据库的文件中
·插入存储整个元组
·删除将元组标记为已删除
·更新仅包含已修改属性的增量

Log

Instead of storing tuples in pages, the DBMS only appends log records (日志记录) to the file of how the database was modified

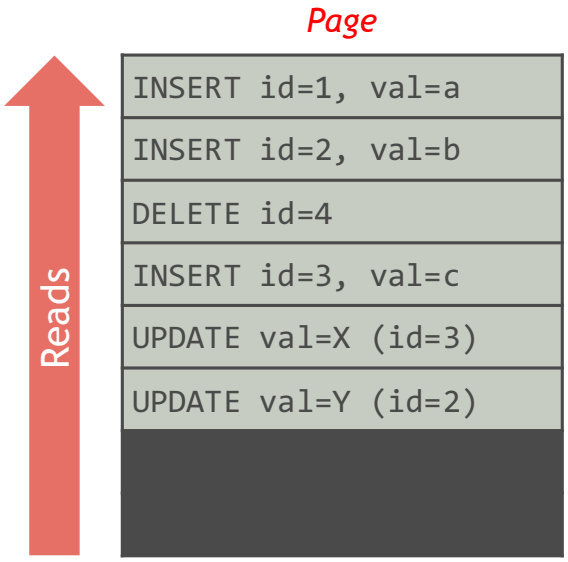
- Inserts store the entire tuple
- Deletes mark the tuple as deleted
- Updates contain the delta of just the attributes that were modified



日志结构的页面布局（续）
为了读取元组，DBMS向后扫描日志并“重新创建”元组以找到所需的内容

Log-Structured Page Layout (Cont'd)

To read a tuple, the DBMS scans the log backwards and "recreates" the tuple to find what it needs



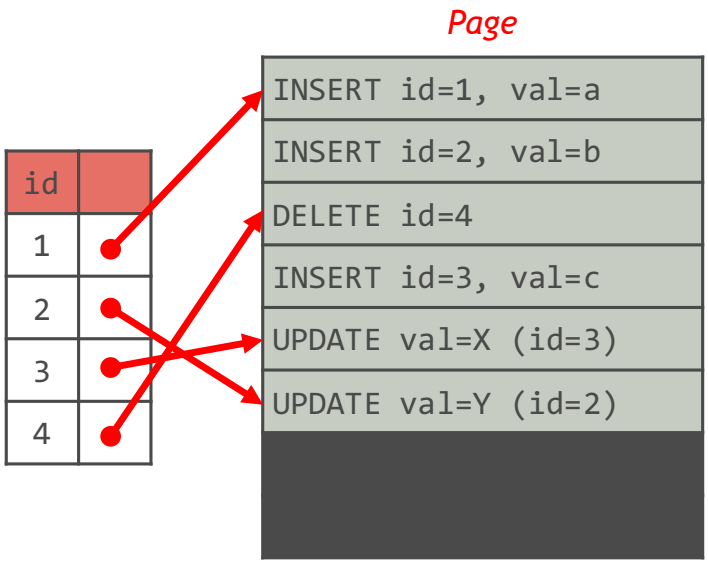
(id=1, val=a); (id=2, val=Y); (id=3, val=X); id=4 deleted;

Log-Structured Page Layout (Cont'd)

日志结构的页面布局（续）

构建索引以允许DBMS跳至日志中的位置

Build indexes to allow the DBMS to jump to locations in the log



Log-Structured Page Layout (Cont'd)

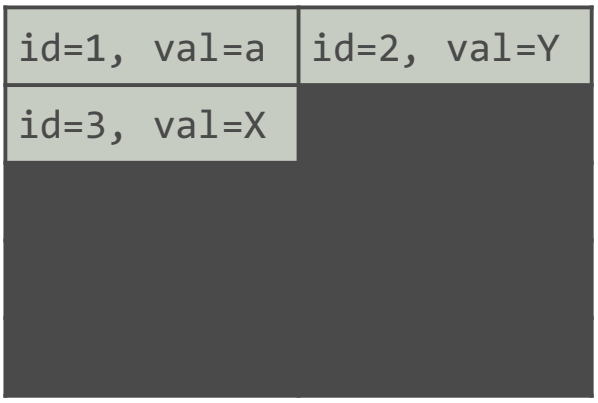
日志结构的页面布局（续）

定期压缩日志

压缩页

Periodically compact the log

Compacted Page



磁盘上数据库的表示形式
文件组织

Representation of Databases on Disks
File Organization

File Storage

档案储存

DBMS将数据库存储为磁盘上的一个或多个文件
·操作系统对这些文件的内容一无所知

存储经理负责维护数据库文件。 它将文件组织为页面的集合。
·它跟踪读取/写入页面的数据
·跟踪可用空间

The DBMS stores a database as one or more files on disk

- The OS doesn't know anything about the contents of these files

The storage manager is responsible for maintaining a database's files. It organizes the files as a collection of pages.

- It tracks data read/written to pages
- It tracks the available space

Page Storage Architecture

页面存储架构

不同的DBMS以不同的方式管理磁盘上文件的页面
·方法1：堆文件的组织
·方法2：顺序/排序的文件组织
·方法3：哈希文件组织（第8章）

Different DBMSs manage pages in files on disk in different ways

- Approach #1: Heap File Organization
- Approach #2: Sequential/Sorted File Organization
- Approach #3: Hashing File Organization (Chapter 8)

Heap File Organization (堆文件组织)

堆文件是无序的页面集合，其中元组以随机顺序存储
·创建/获取/写入/删除页面
·还必须支持对所有页面的迭代

A heap file (堆文件) is an unordered collection of pages where tuples that are stored in random order

- Create/get/write/delete pages
- Must also support iterating over all pages

Need meta-data to keep track of what pages exist and which ones have free space

需要元数据来跟踪存在哪些页面以及哪些页面具有可用空间

Two ways to represent a heap fi

表示堆文件的两种方法
·方法1：链接列表
·方法2：页面目录

- Approach #1: Linked lists
- Approach #2: Page directory

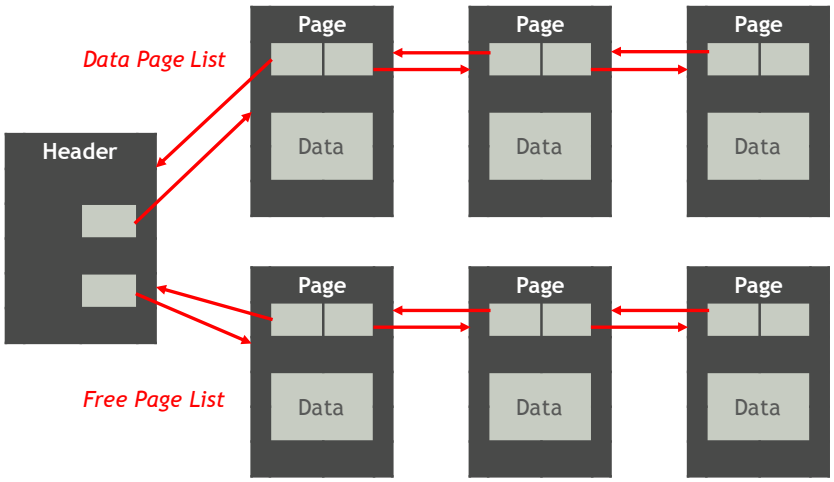
Heap Files: Linked Lists (链表法)

Maintain a header page at the beginning of the file that stores two pointers

在文件的开头维护一个包含两个指针的标题页
·指向空闲页
·指向数据页
每个页面都跟踪其自身的可用插槽数

- The head of the free page list
- The head of the data page list

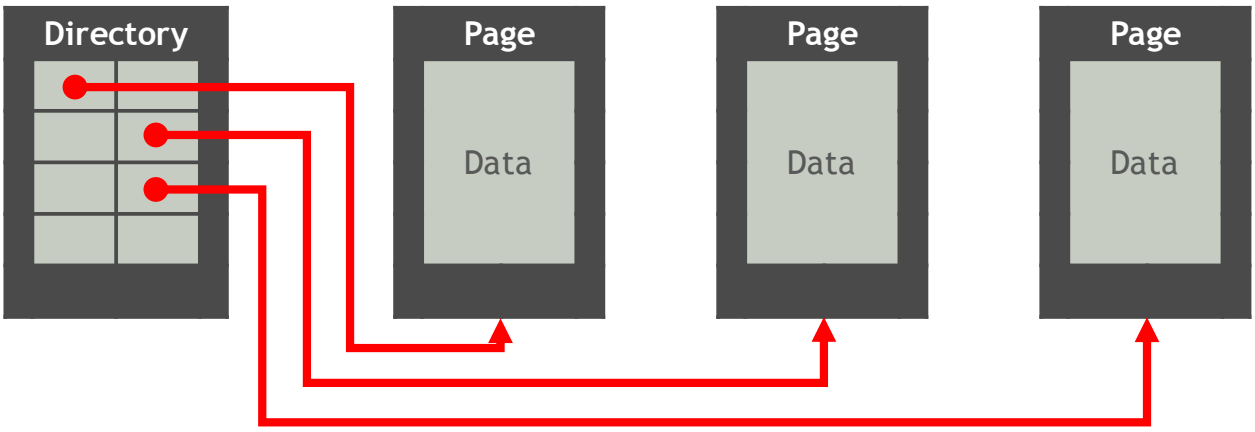
Each page keeps track of the number of free slots in itself



·DBMS维护特殊页面，这些页面可跟踪数据文件在数据库文件中的位置
·该目录还记录每页的可用插槽数
·DBMS必须确保目录页面与数据页面同步

Heap Files: Page Directory (页目录法)

- The DBMS maintains special pages that tracks the location of data pages in the database files
- The directory also records the number of free slots per page
- The DBMS has to make sure that the directory pages are in sync with the data pages



Sequential/Sorted File Organization (顺序/有序文件组织)

顺序/排序文件是页面的有序集合，其中元组以排序键顺序存储
除非使用主索引，否则很少使用有序文件

A sequential/sorted file is an ordered collection of pages where tuples that are stored in sorted key order

- Ordered files are rarely used unless a primary index is used

系统目录

System Catalogs

System Catalogs (系统目录)

系统目录

DBMS在其内部目录中存储有关数据库的元数据
·表，列，索引，视图
·用户权限
·内部统计

几乎每个DBMS都会自己存储数据库目录

A DBMS stores meta-data about databases in its internal catalogs

- Tables, columns, indexes, views
- Users, permissions
- Internal statistics

Almost every DBMS stores their a database's catalog in itself

Syst

您可以查询DBMS的内部INFORMATION SCHEMA目录以获取有关数据库的信息
·ANSI标准的只读视图集，可提供有关所有
·数据库中的表，视图，列和过程

DBMS还具有非标准的快捷方式来检索此信息

You can query the DBMS's internal `INFORMATION_SCHEMA` catalog to get info about the database

- ANSI standard set of read-only views that provide info about all of the tables, views, columns, and procedures in a database

DBMSs also have non-standard shortcuts to retrieve this information

Example (MySQL System Catalogs)

- List all the tables in the current database: `show tables;`
- List the schema of a table: `describe student;`

列出当前数据库中的所有表：
列出表格的模式：描述学生；

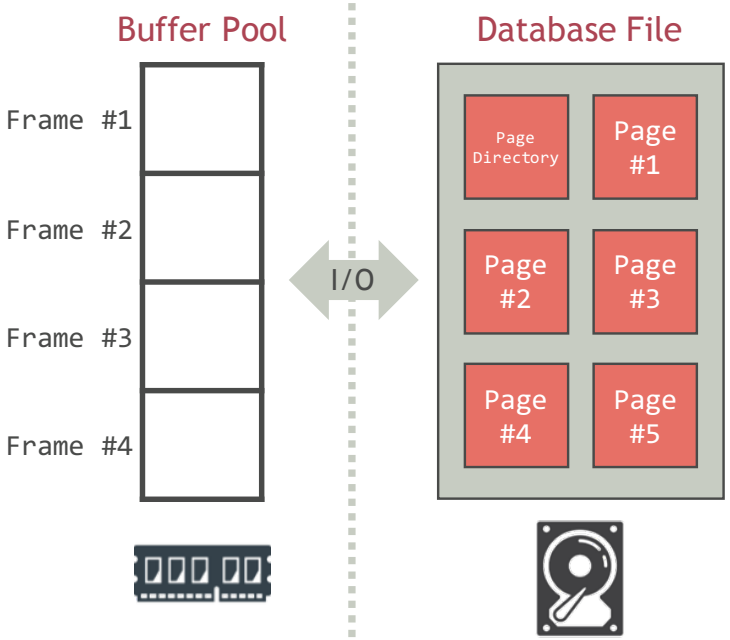
缓冲区管理

Buffer Management

可用内存区域被划分为固定大小的页面数组，这些页面统称为缓冲区池（◆≤`）
缓冲池中的页面称为页框

Buffer Pool (缓冲池)

- The available memory region is partitioned into an array of fixed-size pages, which are collectively called the buffer pool (缓冲池)
- The pages in the buffer pool are called frames (页框)

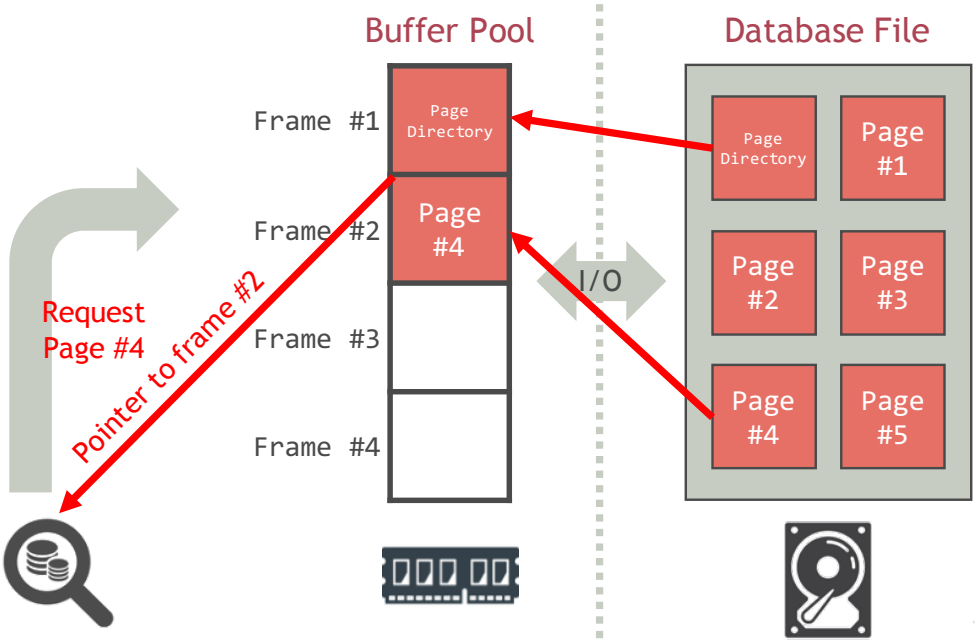


买方经理负责根据需要将页面带入买方池
买主管理器决定替换买主池中的现有页面以为新页面腾出空间（如果买主池已

Buffer Manager (缓冲区管理器)

The buffer manager is responsible for bringing pages into the buffer pool as needed

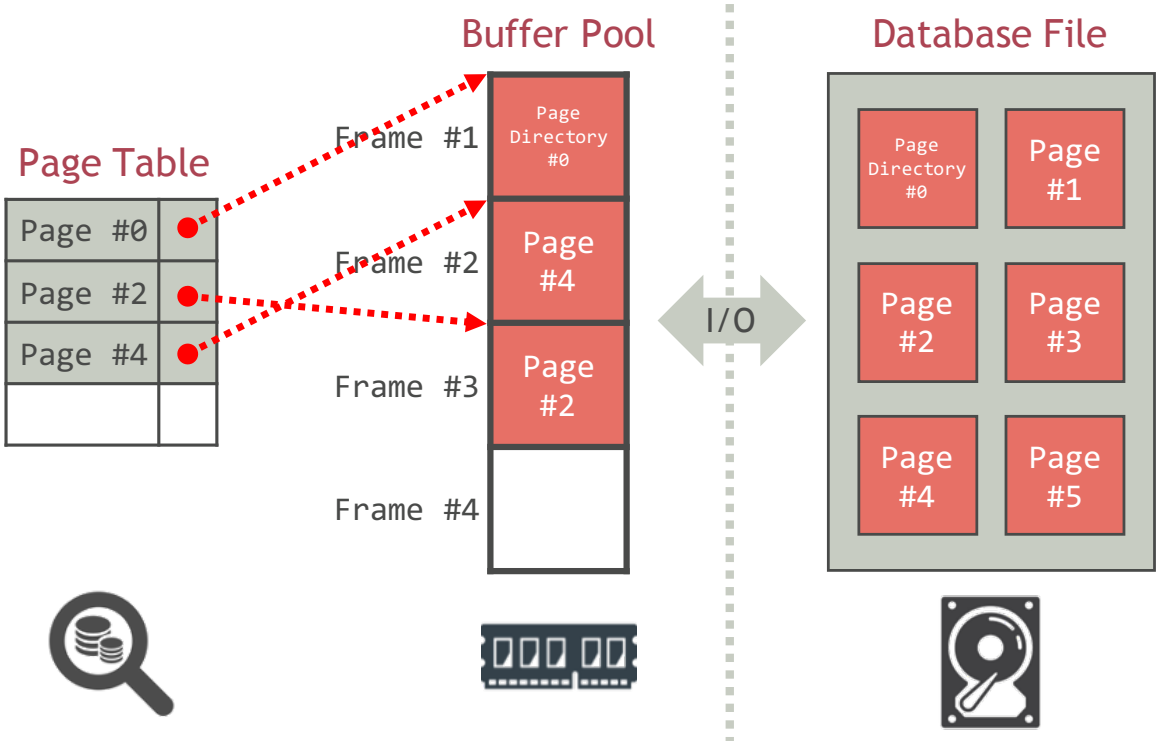
- The buffer manager decides what existing page in the buffer pool to replace to make space for the new page (if the buffer pool is full)



缓冲池内部结构：页表（uh）
页面表跟踪当前缓冲区池中的页面

Buffer Pool Internals: Page Table (页表)

The page table keeps track of pages that are currently in the buffer pool



页面表与页面目录
页面目录是从页面ID到数据库文件中页面位置的映射。
·所有更改都必须记录在磁盘上，以便DBMS在重新启动时能够找到
页面表是从页面ID到页面副本的映射。
缓冲池框架
·这是一种内存中的数据结构，不需要存储在磁盘上

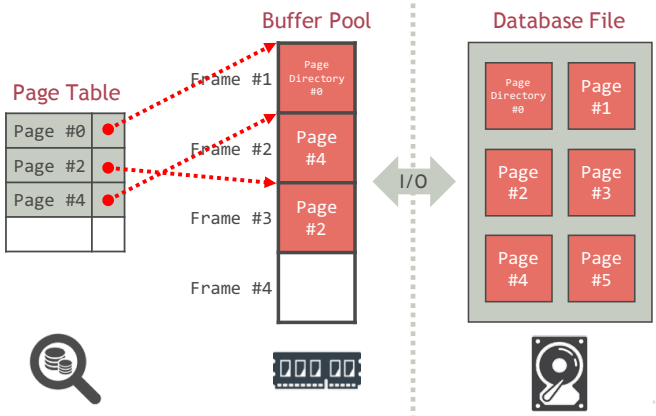
Page

The page directory is the mapping from page IDs to page locations in the database files

- All changes must be recorded on disk to allow the DBMS to find on restart

The page table is the mapping from page IDs to a copy of the page in buffer pool frames

- This is an in-memory data structure that does not need to be stored on disk

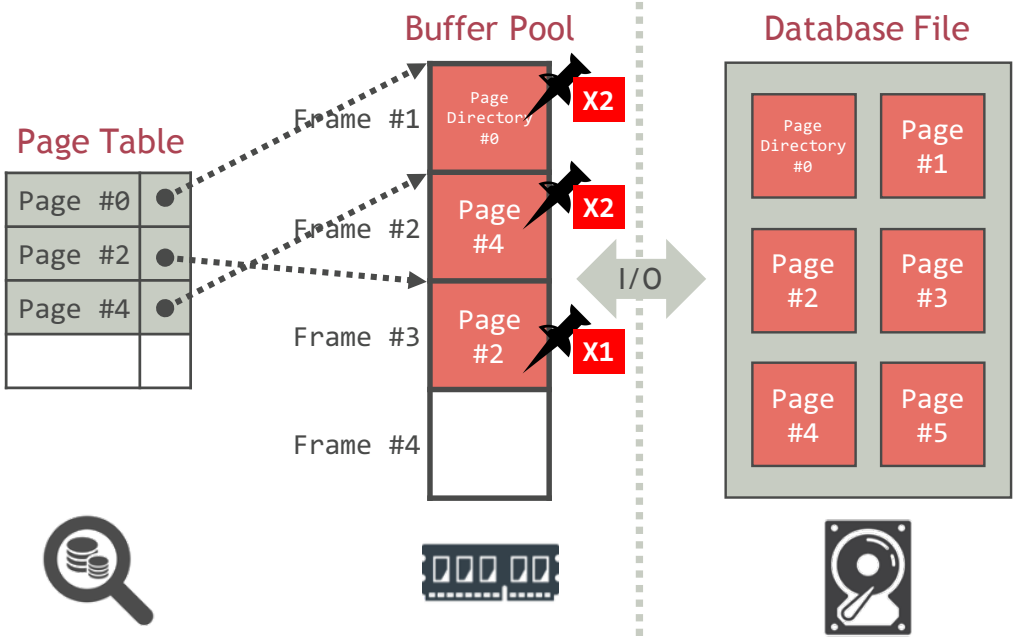


缓冲池内部：框架的元数据
缓冲区管理器为每个帧维护两个变量
·pin count：请求当前框架中的页面但未被释放的次数，即页面的当前用户数

Buffer Pool Internals: Frame's Meta-Data

The buffer manager maintains two variables for each frame

- `pin_count`: the number of times that the page currently in the frame has been requested but not released, i.e., the number of current users of the page

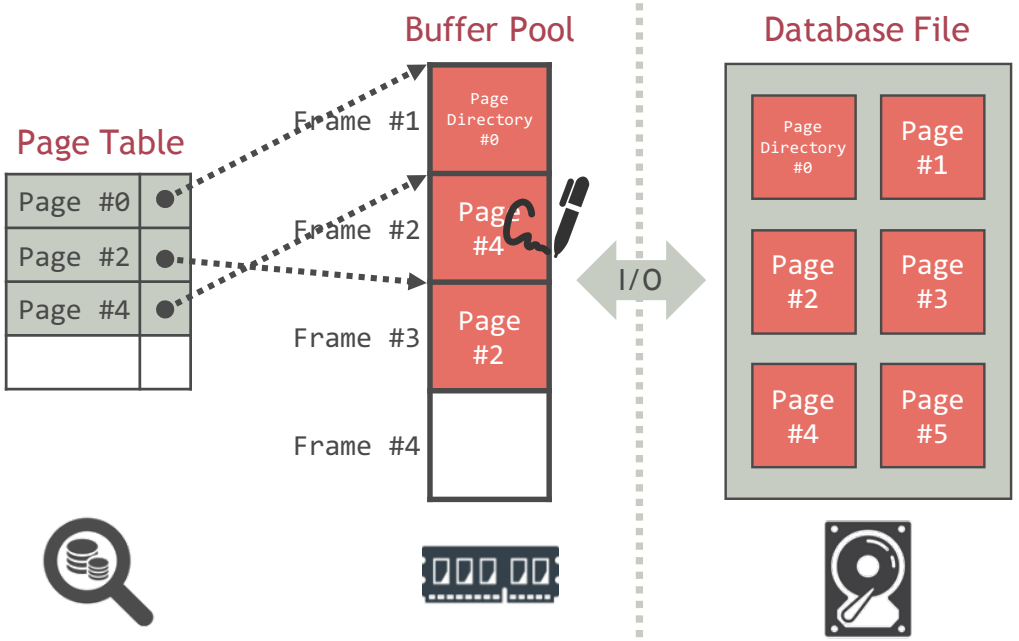


缓冲池内部：框架的元数据
缓冲区管理器为每个帧维护两个变量
·dirty：框架中的页面自进入缓冲池以来是否已被修改的状态

Buffer Pool Internals: Frame's Meta-Data

The buffer manager maintains two variables for each frame

- `dirty`: the status whether the page in the frame has been modified since it was brought into the buffer pool

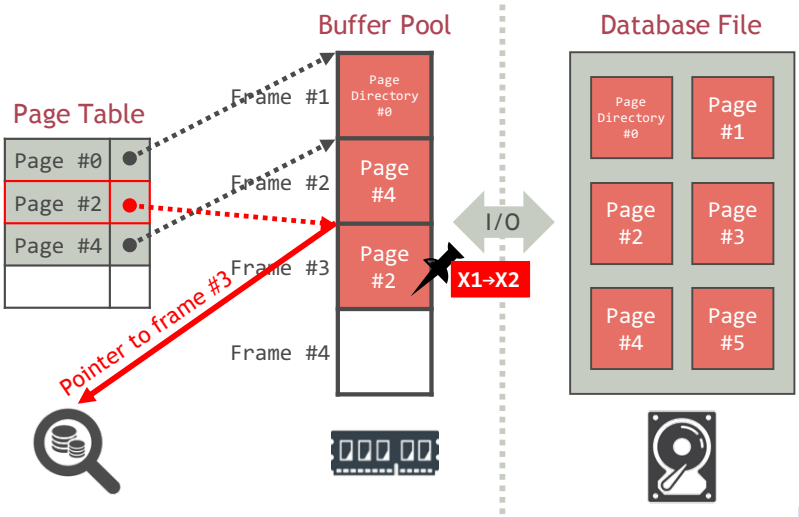


1检查页表以查看某帧是否包含请求的页P
2如果P在缓冲区池中，则在P页上固定引脚，即增加包含P的帧的引脚数
3返回包含P的帧的指针

Page Requests (请

1. Check the page table to see if some frame contains the requested page P
2. If P is in the buffer pool, pin page P, i.e., increment the `pin_count` of the frame containing P
3. Return the pointer of the frame containing P

Example: Request page #2



Page Requests (Cont'd)

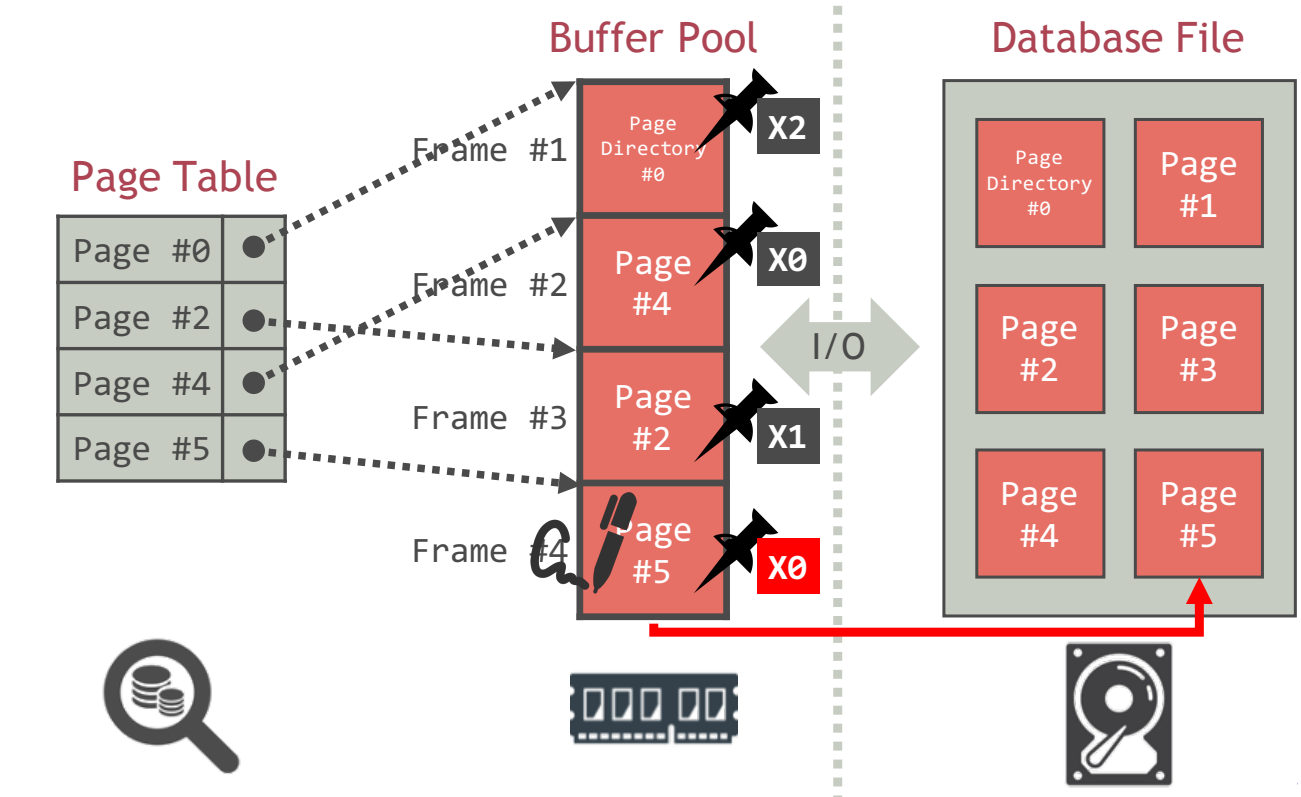
如果请求的页面不在缓冲区池中，
1使用替换策略选择针数= 0的帧进行替换，并增加其针数
2如果替换框架的脏位打开，则将其包含的页面写入磁盘
3将请求的页面读入替换框中

If the requested page is not in the buffer pool,

1. (选替换页) Chooses a frame with `pin_count` $= 0$ for replacement, using the replacement policy, and increments its `pin_count`
2. (写回脏页) If the `dirty` bit for the replacement frame is on, writes the page it contains to disk
3. (读请求页) Reads the requested page into the replacement frame

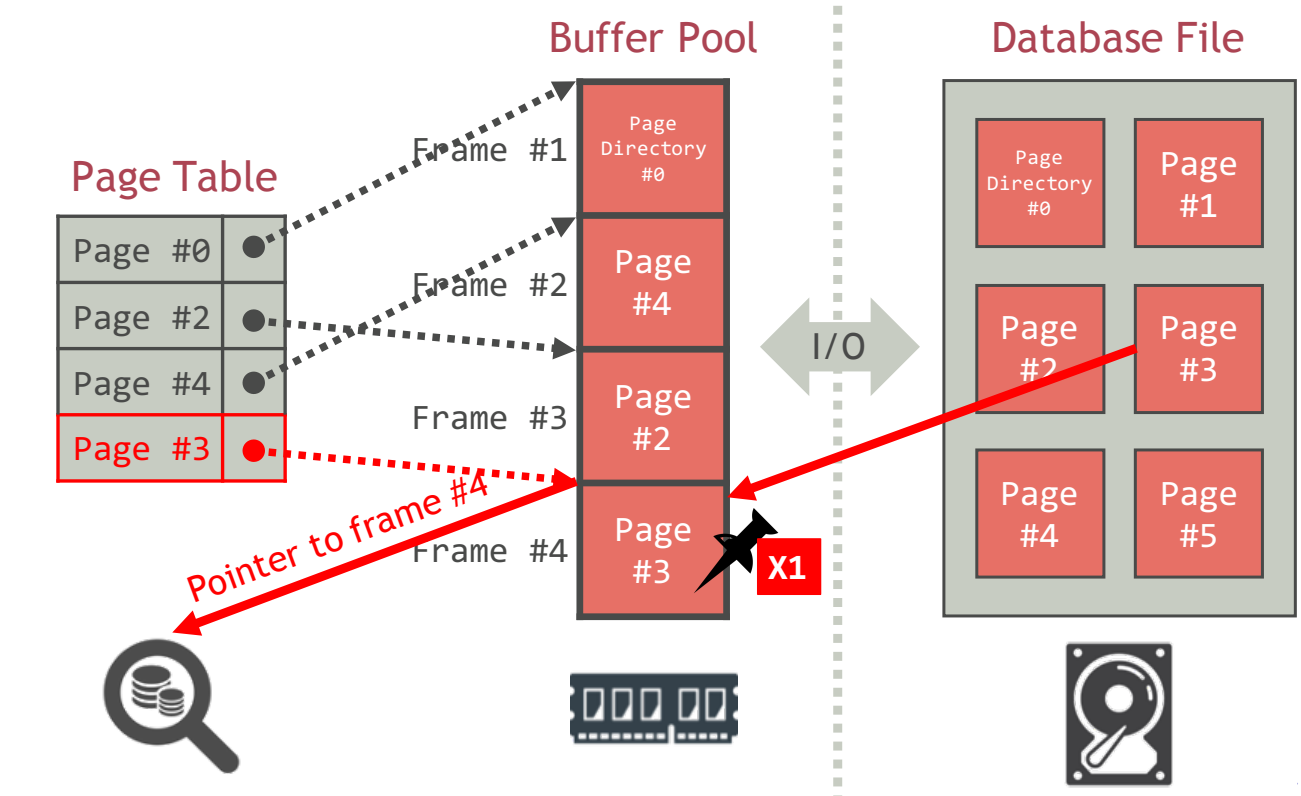
Page Requests (Cont'd)

Example: Request page #3



Page Requests (Cont'd)

Example: Request page #3

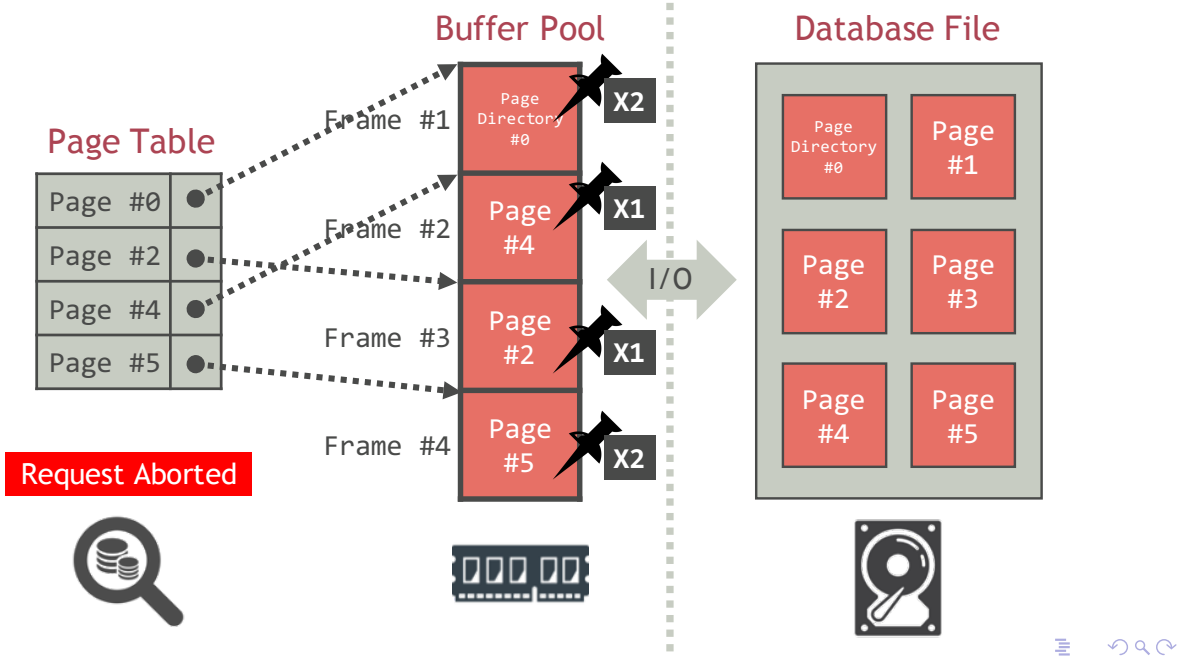


页面请求（续）
如果缓冲区池中没有页面的引脚数= 0，则缓冲区管理器必须等待某个页面被释放，然后才能响应该页面请求
实际上，请求页面的事务可以简单地中止

Page Requests (Cont'd)

If no page in the buffer pool has `pin_count` = 0, the buffer manager must wait until some page is released before responding to the page request

In practice, the transaction requesting the page may simply be aborted

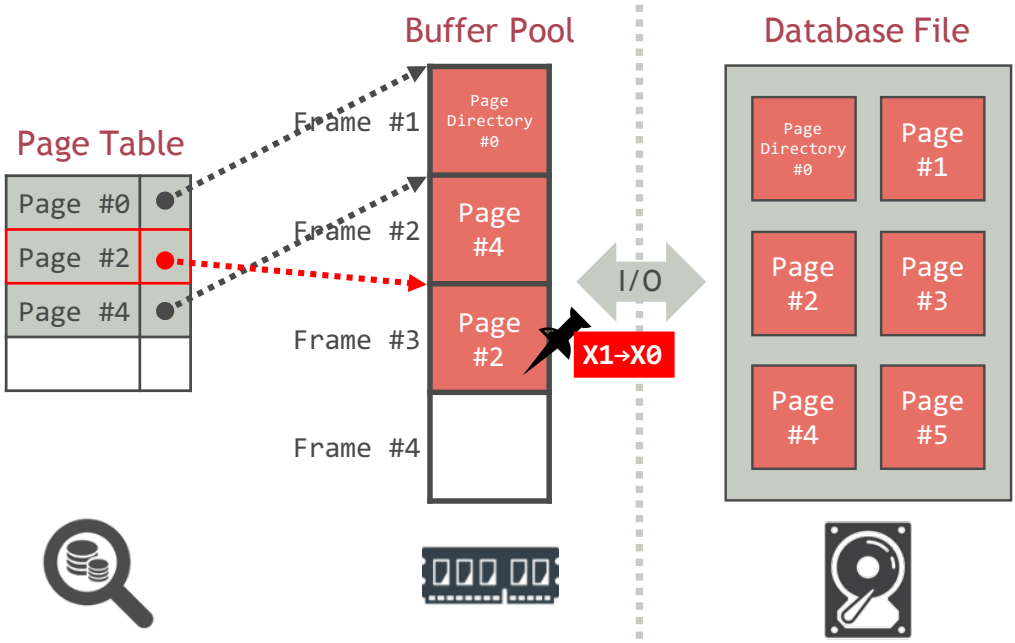


Page Releases (页释放)

Unpin the released page, i.e., decrement the `pin_count` of the frame containing the page

取消固定释放的页面，即减少包含页面的框架的固定针数

Example: Request page #2

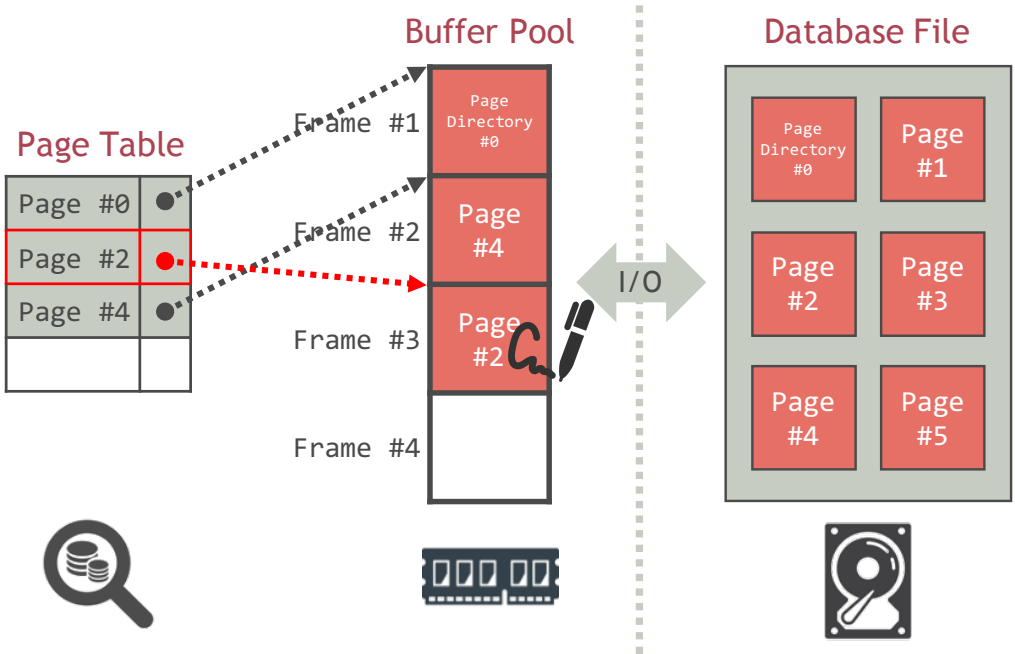


Page Modifications (页修改)

Set the `dirty` bit for the frame containing the modified page

Example: Modify page #2

设置包含修改后页面的框架的脏位

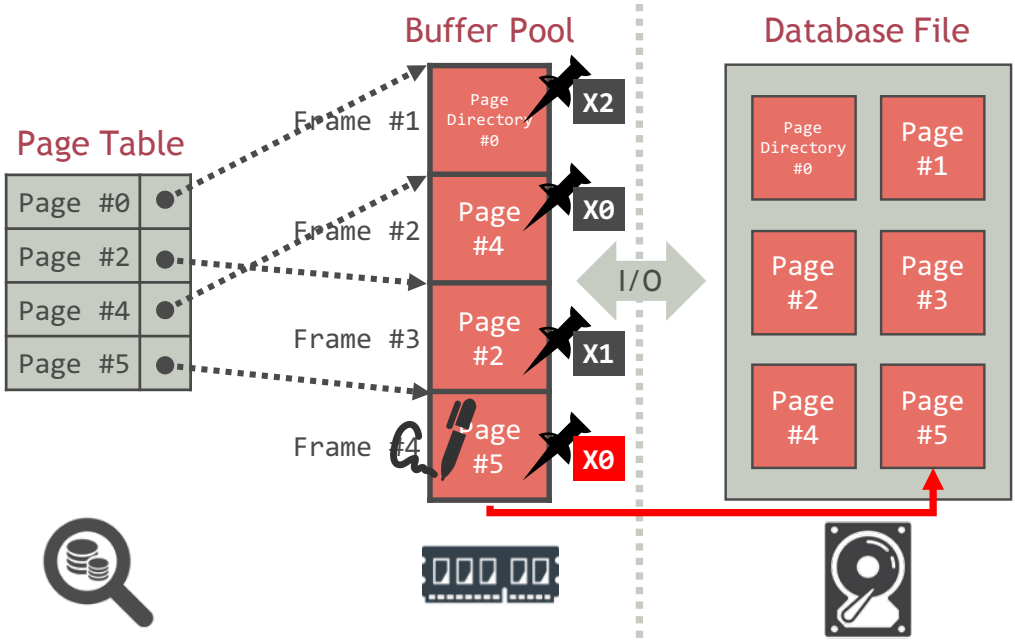


当缓冲区管理器需要释放一个框架以腾出空间容纳新页面时，它必须决定从缓冲区池中逐出哪一页
页面替换策略会大大影响数据库操作所花费的时间

Page Replacement Policies (页替换策略)

When the buffer manager needs to free up a frame to make room for a new page, it must decide which page to evict from the buffer pool

- The page replacement policy can affect the time taken for database operations considerably



Least Recently Used (LRU) Relpacement Policy

最近最少使用（LRU）淘汰政策

维护每个页面上次访问的时间戳

当DBMS需要逐出页面时，请选择时间戳最旧的页面
使页面保持排序，以减少逐出的搜索时间

Maintain a timestamp of when each page was last accessed

When the DBMS needs to evict a page, select the one with the oldest timestamp

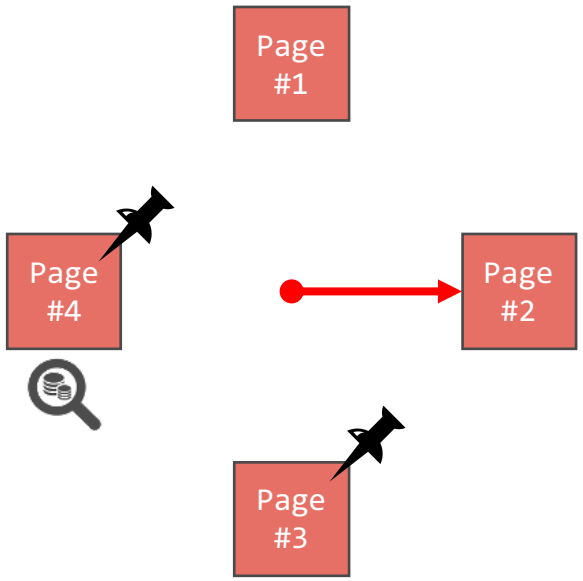
- Keep the pages in sorted order to reduce the search time on eviction

时钟替换策略是LRU的近似值，不需要每页单独的时间戳记
·每个帧都有一个参考位，可以是0或1
·将页面读入框架或访问框架中的页面时，其参考位设置为1

Clock Replacement Policy (时钟替换策略)

The clock replacement policy is an approximation of LRU without needing a separate timestamp per page

- Each frame has a reference bit, which is either 0 or 1
- When a page is read into a frame or when the page in a frame is accessed, its reference bit is set to 1

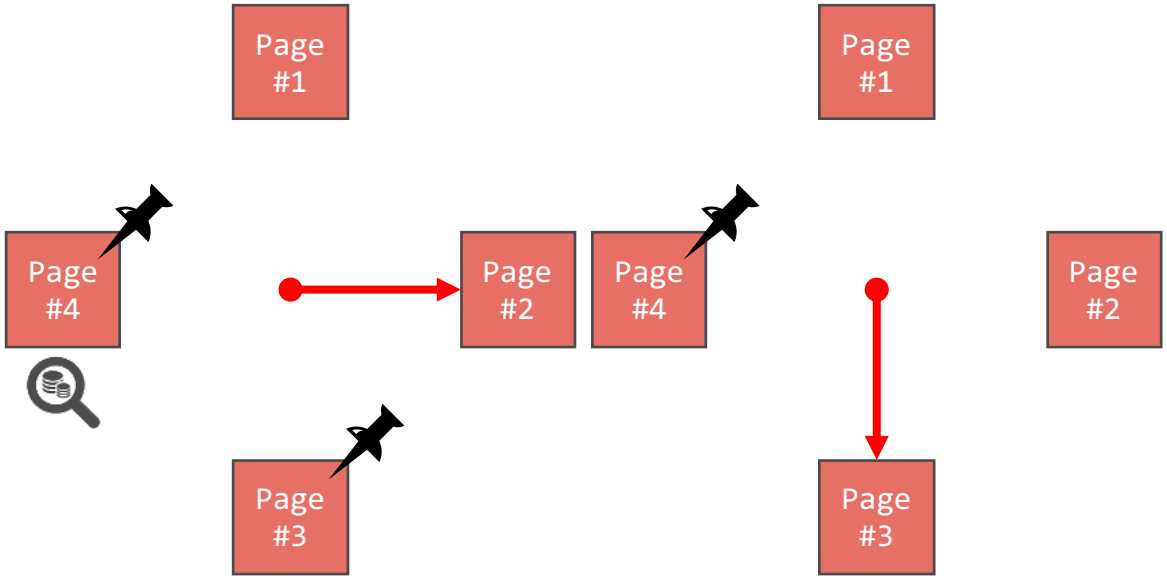


时钟更换政策（续）
框架采用圆形表圈和“时针”进行组织
时钟指针顺时针旋转
扫描后，检查页面的参考位是否设置为1
如果是，则将参考位设置为0

Clock Replacement Policy (Cont'd)

The frames are organized in a circular buffer with a "clock hand"

- The clock hand rotates clockwise
- Upon sweeping, check if a page's reference bit is set to 1
- If yes, set the reference bit to 0

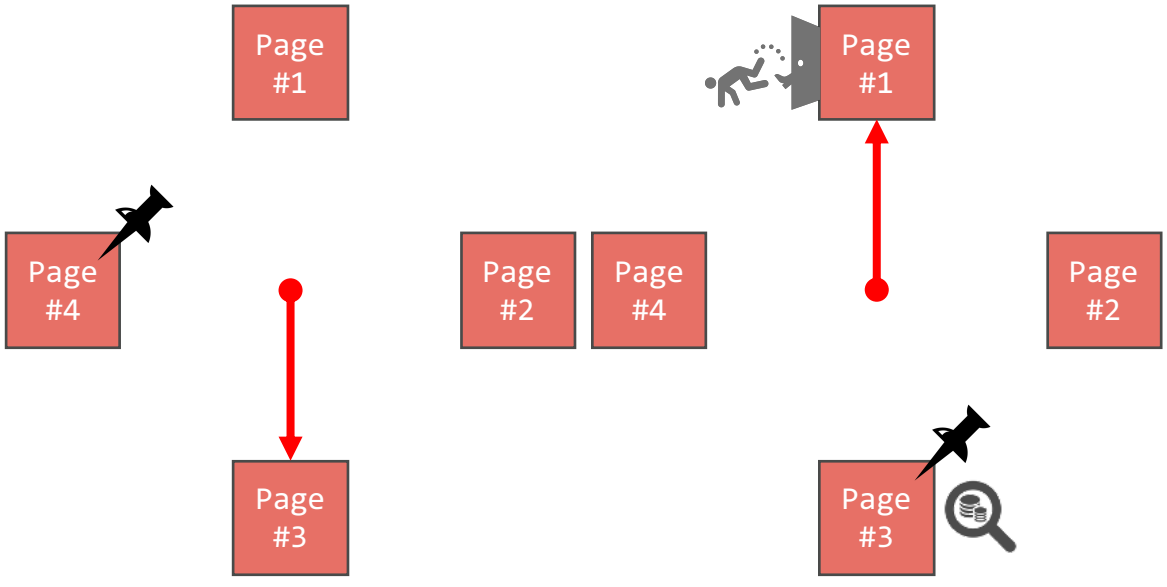


当缓冲区管理器需要新页面的框架时，
·用参考位0查找第一个帧
·逐出框架中的页面

Clock Replacement Policy (Cont'd)

When the buffer manager needs a frame for a new page,

- Look for the first frame with reference bit 0
- Evict the page in the frame



缓冲池VS虚拟内存

Buffer Pool VS Virtual Memory

相似之处
相似的目标：两者都提供对无法装入主存储器的更多数据的访问
相似的操作：两者都根据需要将页面从磁盘移入主存储器，替换不再在主存储器中需要的页面
为什么我们不能使用操作系统的虚拟内存功能来构建DBMS?

Similarities

- Similar goal: Both provide access to more data that cannot fit in main memory
- Similar operations: Both bring in pages from disk to main memory as needed, replacing pages no longer needed in main memory

Why can't we build a DBMS using the virtual memory capability of an OS?

Buffer Management VS Virtual Memory (Cont'd)

原因1（页面参考模式预测）
与操作系统环境中的典型情况相比，DBMS通常可以更准确地预测页面的访问顺序或页面引用模式。

Why can't we build a DBMS using the virtual memory capability of an OS?

Reason 1 (Page reference pattern prediction)

A DBMS can often predict the order in which pages will be accessed, or page reference patterns much more accurately than is typical in an OS environment

意义
页面替换：预测页面参考模式的能力使您可以更好地选择要替换的页面
页面预取：能够预测页面参考模式，使bu↵buerer管理员可以预测接下来的几个页面请求，并在请求页面之前将相应的页面获取到内存中

Implication

- Page Replacement: The ability to predict page reference patterns allows for a better choice of pages to replace
- Page Prefetching: Being able to predict page reference patterns enables the buffer manager to anticipate the next several page requests and fetch the corresponding pages into memory before the pages are requested

原因2（强制页面写入）
DBMS需要具有将页面显式强制到磁盘的功能，即确保使用内存中的副本来更新磁盘上的页面副本。
可以通过实质上记录写请求并推迟磁盘副本的实际修改来实现将页面写到磁盘的OS命令。 如果在此期间系统崩溃，则对DBMS的影响可能是灾难性的

Buffer Management vs Virtual Memory (Cont'd)

Why can't we build a DBMS using the virtual memory capability of an OS?

Reason 2 (Forcing page write)

- A DBMS requires the ability to explicitly force a page to disk, that is, to ensure that the copy of the page on disk is updated with the copy in memory
- The OS command to write a page to disk may be implemented by essentially recording the write request and deferring the actual modification of the disk copy. If the system crashes in the interim, the effects can be catastrophic for a DBMS

Implication

A DBMS must be able to ensure that certain pages in the buffer pool are written to disk before certain other pages to implement the Write Ahead Logging (WAL) protocol for crash recovery

意义
DBMS必须能够确保将缓冲池中的某些页面写入磁盘，然后再执行某些其他页面以实现预写日志记录（WAL）协议以进行崩溃恢复

Summary

1. Storage Media
2. Representation of Databases on Disks
   - Value Representation
   - Tuple Layout
   - Page Layout
     - Tuple-Oriented Page Layout
     - Log-Structured Page Layout
   - File Organization
3. System Catalogs
4. Buffer Management

1个存储介质

2磁盘上数据库的表示形式
·价值表达
·元组布局
·页面布局
·面向元组的页面布局
·日志结构页面布局
·文件组织

3系统目录

4电池管理