ELE2742 / ENG1467 - PUC-Rio, 2019.1

Lecture 3: Database Storage I

DATE

Lecturer: Alexandre Street *Scribe: Kaifeng Lin*

1 STORAGE HIERARCHY

In this course, we mainly classify the storage as volatile and non-volatile.

1.1 Difference

The general definition is that if the storage will lose data when the power is cut off, we call it volatile. Otherwise, it is called non-volatile.

1.2 Volatile

In this course, the only volatile storage we are going to consider is Dram, which is also referred as memory. Since it is above the non-volatile in the hierarchy, it is much faster than them, which also means it doesn't contain much storage compared to them.

1.3 Non-volatile

In this course, we refer them as disk. That is, there are many layers, with different storage, runtime. However, we don't really distinguish among them in this course. We just need to know that disk is non-volatile.

2 WHY NOT USE THE OS?

Introduction: the idea of the database system management will need to use the idea of virtual memory. This is easy to understand because it saves the runtime if two data of the same table live together. However, it is well-known that the OS has implemented the virtual memory very well. Then why don't we just use the OS?

2.1 mmap

The idea of mmap is pretty much the function between virtual pages to disk. You use some function to images them. This is the function being used in OS, not database management system!

2.2 Reason 1. Multi-thread problem

In operation system, there is no something called multi-thread, everything will be executed by the system sequentially. Therefore, the OS is designed to be not "thread-safe" because there is only one thread. However, in the database management system, it is unavoidable to use multi-thread, because there are a lot of data to handle, and multi-thread has already been a "must".

The reason why we can't use OS is that, when there are multiple writers, there will be a race condition, and since mmap doesn't care about thread safe, so it is very dangerous.

3 File storage

This is the top level hierarchy because we consider all data are stored in files. Therefore, at this point, we don't need to care about how what is inside the page. We only care about how we organize the pages.

3.1 Types

Typically, there are three types of files storage, heap file, sequential file and hash file.

3.2 Heap file

Definition: the heap file storage means we don't care about the order of each record. Therefore, they can appear in anywhere in the file. There are mainly two kinds of heap files storage. One is linked list and the other one is page directory.

Linked list Similar to the malloc design in CS241, here we use two head pointers (or sentinels) to denote the heads. One head point to the existed files, and the other head pointer pointing to those "freed" files. In this setting, each metadata is stored in each sub file. Therefore, to find a file, we basically need to iterate through all the files, which could be time-consuming.

Page directory Instead of recording the metadata separately, we record the information in each page to a page directory, so that the page directory has all the information of all pages. Therefore, whenever we modify any of the page, we will need to update the page directory. Notice that the page directory controls everything, and therefore will need to use another way to do the protection!

4 Page layout

Introduction: in this section, we come to the layer below page, so we are focusing the content within a page(file). The problem now becomes how we record the data in a page.

4.1 Types

There are mainly two types to store data within a single page: tuple-oriented and log-structure

4.2 Tuple-oriented

This means we store each data as a tuple, the first element is a metadata of the content, and the second element is the content. Nowadays, slotted array is the most popular structure in this setting. The slotted array means all the metadata are in the front of the page, and all the real content are in the end of the page. We have each metadata pointing to its content.

4.3 Log-structure

The basic idea is that we have each record as just a sentence. We don't actually record the data, but we just record the commands, which modify the page. In this setting, the writing commands will be extremely fast, which takes only constant time. However, the read in this setting is somehow unfriendly.

5 Tuple layout

Introduction: this is the most fundamental layer of database management system. The previous layer is page layout, which takes the layout of the tuple as granted. Essentially, tuple layout is just the way we store data. Notice that data are just bytes in computer, so we are just storing bytes!

5.1 Tuple header

This is one of the components in a tuple. It records the metadata for the corresponding tuple, for example the Visibility information for concurrency control.

5.2 Tuple data

The real information about the record, which is the content of the tuple.