ELE2742 / ENG1467 - PUC-Rio, 2019.1

Lecture 5: Buffer pool

DATE

Lecturer: *Scribe: Kelfin Lin*

1 Buffer pool organization

Introduction: the idea of buffer pool is very similar to the virtual memory in OS. So in this section, we will need to do some comparisons, and it is also a good way to memorize these concepts by comparisons.

1.1 Components

Recall from CS61C, the components of a virtual memory system are TLB (associative cache), page table, and disk.

In database system, TLB is TBD, not sure about that. But we do have page table and disk as well. Besides, we also have a new component- buffer pool. So, at least three components are needed in this system.

1.2 Page table

Recall that in OS, the page table is used to map the virtual memory address to the physical memory address. Note that the components inside the page tables are just the addresses of pages! That is to say, the page tables only contain the ids of the pages, they don't really store data.

In database management system, it is very similar. We use the page table to do the mapping from a page id, to the actual frame in buffer pool. In other words, the page table in this system is between the virtual address to the buffer pool. As for the usage of buffer pool, refer to the next subsection.

1.3 Buffer pool

This is a completely new component from OS. Note that this is a structure stored in the memory, or say DRAM. It stores the actuall data, instead of addresses. (The page tables only store the address) Whenever the page table gives an address, the buffer pool will look up at its table, and see if it processes the corresponding data. If it has, it returns the data, otherwise, it will fetch data from the disk.

For further references see Link

or see the url : `https://www.reddit.com/r/Database/comments/oxtvd3/need_some_clarification_on_buffer_pool_and_page/`

Summary: this is basically like a in-memory cache, it stores the data in memory. This makes the lookup faster.

1.4 Disk

In OS, the disk is the place where we store all the data, so is in this system. The only difference between these two systems in disk is probably the layout. But at this layer, we don't really care about the layout of each structure. So, we can think of them as the same.

2 Locks and latches

Introductions: when implementing a database management system, we will probably need to use parallel structure. Here we distinguish locks and latches because these two concepts are basically equal in other "systems", but they are totally different concepts in database management system, and that's the reason why we distinguish them in an early phase.

2.1 Locks

This is a high level concept. A lock is used when you want to protect some of your database from modification. This is done when all your fundamental data structures are built successfully.

2.2 Latches

This is low level concept. A latch is used within the data structure. Recall that in CS241, we implement a queue, which is thread safe. A latch is used when we are trying to implement a thread-safe data structure and mutex is just one of the latches, though in CS241, we call mutex a kind of locks.

3 Page table and page directory

Introduction: page table and page directory are totally different things in database system. Refer to the previous notes if you have no ideas what these concepts are.

3.1 Differences

A page table is used in memory while page directory is used in page layout. Therefore, table table is a higher level concept, and it is built on the top of the page directory. Recall from the previous lecture that page directory records all the information of all the pages. Therefore, the page directory should be stored in the disk!

4 Multiple buffer pools

Introductions: recall from the definition of buffer pools, it is somewhat like a in-memory caches. In database system, we really care about the concurrency. Therefore, if we have multiple buffer pools, we can just have different buffer pools storing different kinds of data. For example, a database system may have 5 tables. We can then create five buffer pools, with each caching one table. This is useful when we will need to handle multiple tables at the same time.

5 Pre-fetching

Introduction: The concept is pretty simple as the name suggests. Before implementing any queries, that is when the database system is loaded to memory, we will caches some data. There are two situations. One is that those pre-fetched data is not used, then we can just replace the caches. This takes basically the same amount of time if the cache is empty initially. The other case is that the data we pre-fetched is used, this is pretty good because we can now load the data immediately from the cache.

For further references see Link

6 Scan sharing

Introduction: this concept is also pretty simple and straightforward. Let us take a look at an example, say query one and query two both need the data from table A. If we are doing these in parallel, we will have query one pointing to table A in buffer pool. However, since table A is being occupied by query one, query two can only be waiting. This is done by the pin. However, this could be really inefficient if there are multiple queries all trying to accessing one table. Therefore, we can have scan sharing.

6.1 Definition

A scan sharing basically means that there are multiple queries pointing to the same table. In this way, the table can be read by several queries.

7 Buffer pool bypass

We mentioned before that the database will not use the mmap from OS system because it always tries to control everything on its own. However, the database is run on the OS anyway, because this is a program. therefore, some database management system will also make use of the OS. The professor only mentions about Mango DB. In our implementation, we will not rely on any other systems or caches anyway.

8 Buffer replacement policies

Introduction: in the OS, we know that the page table entries will be replaces once there is a page fault. This idea is also used in database management system. However, to make the decision is not easy, or to make a good decision is not easy. There is no such "best" solution, it really depends on the structure of each database system. But we will introduce several commonly used replacement policies here.

8.1 LRU

Recall that this is asked in the tencent interview. This is to say, we are trying to replace the least recently used entry with the newest one. In the implementation, we use a variable (or say attribute) ref to denote whether this variable is recently used. Recall from CS61C, we also use this idea. Say there are seven entries, and we label them from 1 through 7, each time we update the table, regardless of the implementation, we will update the ref variable. So that each time we need to replace an old entry, we find the entry with label 1.

8.2 Priority hints

Instead of using the frequency as a primitive, we can use the priority. For example, we may think the id is the most important one, and we will seldom want to replace it from the caches.

9 Allocation Policies

Introduction: this is to say that the database manament system will only allocate a specific amount of spaces for each query, and a single query can't access to the whole buffer pool.

10 Dirty page

Introduction: this idea is also used in OS. In page table, there is a dirty bit, indicating that whether the current record has been updated in disk. In buffer pool, there is a variable called dirty page, indicating whether the current page is polluted or not. If the page is not dirty and will need to be replaced, we can safely drop it. However, if the page is dirty, we will then need to update the disk before we do any modification to the page.

10.1 Solution

Periodically update. The database management system will update the dirty in the buffer pool in a period. After the period, all the dirty bits in all pages will be unset. The professor mentioned that this is used in many commercial companies.