**COMP 5120/6120 Database Systems I**

**Homework #3**

1. Answer the following questions about files and indexes:
   1. What alternatives are available for the data entries in an index?

There are three main alternatives for what to store as a data entry in an index:

1. A data entry h is an actual data record (with search key value k).

2. A data entry is a (k, rid) pair, where rid is the record id of a data record with search key value k.

3. A data entry is a (k. rid-list) pair, where rid-list is a list of record ids of data records with search key value k.

Of course, if the index is used to store actual data records, Alternative (1), each entry b is a data record with search key value k. We can think of such an index &'3 a special file organization. Such an indexed file organization can be used instead of, for exarnple, a sorted file or an unordered file of records. Alternatives (2) and (3), which contain data entries that point to data records, are independent of the file organization that is used for the indexed file (i.e., Storage and Indexing 277. the file that contains the data records). Alternative (3) offers better space utilization than Alternative (2), but data entries are variable in length, depending on the number of data records with a given search key value.

* 1. What is the difference between a clustered index and an unclustered index? If an index contains data records as ‘data entries’, can it be unclustered?

Clustered vs. unclustered: If order of data records  
is the same as, or `close to’, order of data entries,  
then called clustered index.  
§ Alternative 1 implies clustered.  
§ A file can be clustered on at most one search key.  
§ Cost of retrieving data records through index varies  
greatly based on whether index is clustered or not!

When a file is organized so that the ordering of data records is the same as or close to the ordering of data entries in some index, we say that the index is clustered; otherwise, it clustered is an unclustered index.

An index that uses Alternative (2) or (3) can be a clustered index only if the data records are sorted on the search key field. Otherwise, the order of the data records is random, defined purely by their physical order, and there is no reasonable way to arrange the data entries in the index in the same order

So, in practice, a clustered index is an index that uses Alternative (1), and indexes that use Alternatives (2) or (3) are unclustered. We sometimes refer to an index using Alternative (1) as a clustered file, because the data entries are actual data records, and the index is therefore a file of data records. (As observed earlier, searches and scans on an index return only its data entries, even if it contains additional information to organize the data entries.)

If the index is clustered, i.e., we are using the search key of a clustered file, the rids in qualifying data entries point to a contiguous collection of records, and we need to retrieve only a few data pages. If the index is unclustered, each qualifying data entry could contain a rid that points to a distinct data page, leading to as many data page l/Os 8.'3 the number of data entries that match the range selection,

The main differences are:

There is a main cluster but there can be many unclustered indexes.

The main data is represented by an index in the clustered index. The index of unclustered index only represents data copy.

Pointer to data block is stored in the clustered index. But in the unclustered index, both pointer and value are stored.

It can be clustered and unclustered.

* 1. How many clustered indexes can you create on a file? Would you always create at least one clustered index for a file?

Only one can be created to avoid replication and redundancy, avoiding them is also more beneficial.

At most one, because we want to avoid replicating data records. Sometimes, we may not create any clustered indexes because no query requires a clustered index for adequate performance, and clustered indexes are more expensive to maintain than unclustered indexes.

1. Explain the terms seek time, rotational delay, and transfer time.

seek time ( moving arms to position disk head on track)  
§ rotational delay ( waiting for block to rotate under head )  
§ transfer time ( actually moving data to/from disk surface )

1. What is sequential flooding of the buffer pool?

sequential flooding: Nasty situation caused by  
LRU + repeated sequential scans.  
§ # buffer frames < # pages in file means each page  
request causes an I/O. MRU (Most-recently-used)  
much better in this situation (but not in all  
situations, of course).  
Namely, LRU and CLOCK are susceptible to sequential flooding, where the buffer pool’s contents are corrupted due to a sequential scan. Since sequential scans read many pages quickly, the buffer pool fills up and pages from other queries are evicted as they would have earlier timestamps. In this scenario, the most recent timestamp does not accurately reflect which page we actually want to evict

1. Describe two possible record formats. What are the trade-offs between them?

In a fixed-length record, each field h&<; a fixed length (that is, the value in this field is of the same length in all records), and the number of fields is also fixed. The fields of such a record can be stored consecutively, and, given the address of the record, the address of a particular field can be calculated using information about the lengths of preceding fields, which is available in the system catalog.

In the relational model, every record in a relation contains the same number of fields. If the number of fields is fixed, a record is of variable length only because some of its fields are of variable length. One possible orga,nizatioll is to store fields consecutively, separated by delimiters (which are special characters that do not appear in the data itself). This organization requires a scan of the record to locate a desired field. An alternative is to reserve some space at the beginning of a record for use 1:LS an array of integer offsets-the ith integer in this array is the starting address of the ith field value relative to the start of the record. Note that we also store an offset to the end of the record; this offset is needed to recognize where the last held ends.

Fixed length record format is easy to implement. Since the record size is fixed, records can be stored contiguously. Record address can be obtained very quickly. Variable length record format is much more flexible.

1. Why do frames in the buffer pool have a pin count instead of a pin flag

In addition to the buffer pool itself, the buffer manager maintains some bookkeeping information and two variables for each frame in the pool: pin count and dirty. The number of times that the page currently in a given frame has been requested but not released-the number of current users of the page--is recorded in the pin\_count variable for that frame. The Boolean variable dirty indicates whether the page has been modified since it was brought into the buffer pool from disk

Initially, the pin\_count for every frame is set to 0, and the dirty bits are turned off. When a page is requested the buffer manager does the following: 1. Checks the buffer pool to see if some frame contains the requested page and, if so, increments the pin\_count of that frame. If the page is not in the pool, the buffer manager brings it in as follows: (a) Chooses a frame for replacement, using the replacement policy, and increments its pin\_count. (b) If the dirty bit for the replacement frame is on, writes the page it contains to disk (that is, the disk copy of the page is overwritten with the contents of the frame). (c) Reads the requested page into the replacement frame. 2. Returns the (main memory) address of the frame containing the requested page to the requestor. Incrementing pirLco'llnt is often called pinning the requested page in its frame. When the code that calls the buffer manager and requests the page subsequently calls the buffer manager and releases the page, the pin\_count of the frame containing the requested page is decremented. This is called unpinning the page. If the requestor has modified the page, it also informs the buffer manager of this at the time that it unpins the page, and the dirty bit for the frame is set

Frames in a buffer pool have a pin count instead of a pin flag for several reasons:

**Granular Control:** Pin counts offer finer-grained control over the usage of buffer pool frames, allowing for more precise tracking of concurrent operations.

**Concurrency Handling:** Pin counts help prevent race conditions by indicating how many operations are actively using a frame at a given time.

**Avoiding Premature Replacement:** They help ensure that frames are not prematurely replaced while they are still in use, which can improve system performance.

**Detecting Unused Frames:** Pin counts can identify frames that are no longer actively used, making them available for replacement or reuse.

**Resource Tracking:** They provide a way to monitor and audit the usage of buffer pool resources, aiding in performance optimization and troubleshooting.

Overall, pin counts provide a robust mechanism for managing buffer pool resources efficiently and ensuring the reliability and performance of database systems and other applications that rely on buffer pools.