SUMMARY OF DATABASE STORAGE AND QUERYING

1. Why Is It Important?

Usually users of a database do not have to care the issues on this level. Actually, they should focus more on the logical model of a database rather than the physical model. However, the physical model of the database determines the performance of it on a large scale, and we must provide a good design of it according to the requirement of users. Therefore, the storage and querying of database system is really crucial. Also, it covers a lot of spread knowledge that ought to be integrated. Thus, knowing about the issues concerning the database storage and querying can help us design a better database system.

2. Introduction to Physical Storage

Generally speaking, the issues concerning database storage is making a trade-off between using the fast but expensive main memory and the slow but cheap hard-disk storage. The physical storage can be classified into different types.

(1) Cache

Cache is the quickest but most expensive storage. Usually it is managed by the system hardware, thus there is no need to concern it in a database system.

(2) Main Memory

Main memory is used to store the data being dealt with, and all the operations are done in main memory. It is usually too small and too expensive for a whole database system. In addition, in case of system crash or electricity failure, the contents in the main memory will be lost.

(3) Flash memory

Different from main memory, the data in flash memory remains after electricity failure. Though reading data from flash memory is as fast as from main memory, writing data to it is complex: it must erase the data before writing. The times that the flash memory being erased are limited.

(4) Magnetic-disk Storage

It is the main method to store data for a long time. Usually the whole database is stored in the magnetic-disk. Data must be read to main memory to be operated, and the result must be written back to magnetic-disk.

3. Record Organization

A file can be logically organized as a sequence of records of a database. Therefore, it is important to decide how records are organized in a file. In general, there are several methods in organizing records.

(1) Heap file organization

A record can be placed at any place in a file as long as there is enough space for it. It is easy to figure out that in this case, there is no order for the records.

(2) Sequential file organization

The records are stored according to a so called “search key”. Search key is an attribute or a set of attributes, and is used to search for the records. It does not have to be the primary key of a table, even unnecessary to be the super key.

(3) Hashing file organization

Use a hashing function to assign a value to each record and decide which block in a file the record belongs to.

Usually, the records of one relation (one table) are stored in one file. However, the multitable clustering file organization enables to store more relations in one file, thus accelerate the querying operations since it reduces the I/O operations.

4. Introduction to Index

The most common example of index is the catalogue of a book. It helps you to quickly find the desired chapters. The index in a database system also enables a quick querying when sacrificing some space to store the index. Two major types of index are to be introduced.

(1) Ordered index

The ordered index is based on the sequence of a certain value. Again, this value does not have to be the primary key of a relation. The most commonly used method of ordered index include **dense index**, in which each search key has an index; and **sparse index**, in which only some of the search keys have indexes. Sometimes

when the amount of data is really large, even using index cannot be as efficient as we want it to be. In that case, the multilevel index is needed. A good analogy to the multilevel index is the word at the beginning of each page in a dictionary.

One disadvantage of ordered index is that the performance of querying becomes worse with larger amount of data using sequential or binary search. Thus a new structure of index should be introduced. It is called **B+ tree**. It is a balanced tree and has a really low complexity when searching. Although it is troublesome to insert and delete entries in a B+ tree, if we do not have to update the database frequently, using it is a proper way to improve the performance of querying. For more discussion about B+ tree, you can refer to this blog by Shen Hao: Some details about B+ tree

(2) Hashing index

Using a uniform and random hashing function, we can distribute the records in different buckets so that each bucket has approximately same number of records. If one bucket is overflowed, we use an overflow bucket to deal with it. All the overflow buckets are connected in chains to the original overflowed bucket and form an overflow chaining. A hashing index stores the search keys and its relative pointers in the structure of hashing to implement the function.

The hashing mentioned above is called static hashing, and correspondingly, there is dynamic hashing which enables the dynamic alters of hashing function to suit the change in size of the database system.

(3) Comparison of ordered index and hashing index

Here is the guideline for deciding which kind of index to be used. If the querying of a range is seldom incurred, using the hashing index is fairly efficient. Otherwise, an ordered index is a preferred.

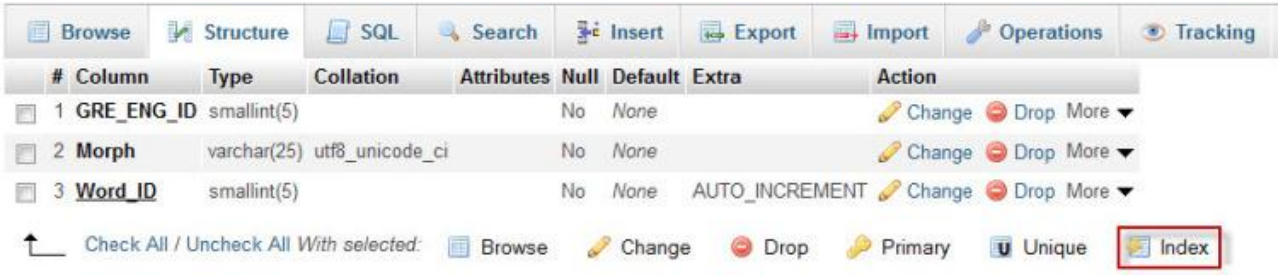
(4) Create index using SQL

When creating an index for a relation, using

CREATE INDEX <index-name> ON <relation-name> (<attribute-list>)

where "attribute-list" is the search key of the index.

It is even easier using MySQL phpmyadmin:



Source: http://toyhouse.cc/profiles/blogs/summary-of-database-storage-and-querying