数据库作业week9

12.1

> SSDs can be used as a storage layer between memory and magnetic disks, with some parts of the database (e.g., some relations) stored on SSDs and the rest on magnetic disks. Alternatively, SSDs can be used as a buffer or cache for magnetic disks; frequently used blocks would reside on the SSD layer, while infrequently used blocks would reside on magnetic disk.

> a. Which of the two alternatives would you choose if you need to support real-time queries that must be answered within a guaranteed short period of time? Explain why.

SSDs can be used as a storage layer between memory and magnetic disks because of its fast speed of querying. If magnetic disks are used to between memory and SSD,some requests may be token slowly.

> b. Which of the two alternatives would you choose if you had a very large *customer* relation, where only some disk blocks of the relation are accessed frequently, with other blocks rarely accessed.

SSDs can be used as a buffer or cache for magnetic disks. Since of that frequently used blocks would reside on the SSD layer, while infrequently used blocks would reside on magnetic disk. And the relation of *customer* is larger that it's too expensive to store it on SSDs.

13.5

> It is important to be able to quickly find out if a block is present in the buffer, and if so where in the buffer it resides. Given that database buffer sizes are very large, what (in-memory) data structure would you use for this task?

I prefer hash table for this task. We can use the hash function to quickly find out if a block is present in the buffer, and if so where in the buffer it resides.

13.9

> In the variable-length record representation, a null bitmap is used to indicate if an attribute has the null value.

> a. For variable-length fields, if the value is null, what would be stored in the offset and length fields?

We set the length field as -1 while the offset field can be any value,since of -1 of length field represents the variable value is null.

> b. In some applications, tuples have a very large number of attributes, most of which are null. Can you modify the record representation such that the only overhead for a null attribute is the single bit in the null bitmap?

We can move the null bitmap to the beginning of the record ,so we can use the single bit in the null bitmap as the null value of the attribute. It's useful when tuples have a very large number of attributes, most of which are null.

13.11

> List two advantages and two disadvantages of each of the following strategies for storing a relational database:

> a. Store each relation in one file.

Advantages

- Since each relation is stored in its own file, we can easily place frequently called relations on the SSD, while rarely used relations can be stored on the disk drive.
- If the sequence of data blocks for a given file is stored close together on the disk platters, reading the relationship data from the hard disk to memory is faster because the data blocks are close together, thus reducing the movement of the disk arm.

Disadvantages

- Optimizations such as **multiclusterable file organization** are not possible because each relation is stored in its own file.
- Opening files is expensive: each access to a relation must first obtain the path to the corresponding file for the relation via the **Data Dictionary**/**System Catalog**. Once the path is found, opening the file (e.g. using the **open()** system call) incurs overhead.

> b. Store multiple relations (perhaps even the entire database) in one file.

Advantages

- Optimizations such as **clusterable file organization** can be implemented if desired
- Assuming that the entire database is stored in a single file, we only need to call the **open()** system call once to get all the information.

Disadvantages:

- If the database stores all relationships in a single file, the **data dictionary** might note down the block containing each relationship record in a data structure (such as a chained table). However, this would deprive us of the benefit of sequential reads from hard disk to main memory.
- Since all the relationships of a database are stored in the same file, it is difficult to optimize, for example, by placing some relationships on a solid-state drive and others on disk.