db hw8

3200105872 庄毅非

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.

Answer: I would choose SSD to store data. Because compared to magnetic disk, usually a SSD can support a larger IOPS and it can retrive a page in a shorter period of time. SSD can also supprt a higher data transform rate.

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.

Answer: I would choose magnetic disk to store data. Because only some blocks of the relation are accessed frequently, I can store them in a memory cache array to get a short data-retriving time. And my cost in buying disks in this case is much less compare to using SSD.

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?

Answer: I will use a (hash)map to store memory's information about its frame's usage in it(hashmap). In that way I can judge where this block is in buffer in O(1) time. If it is, I can also get memory_page_id of it in O(1) time.

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?

Answer: If the value is null, we will store 0 to the offset and length attribute of this value to avoid misreading.

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?

Answer: We can store a bitmap in the head position of a tuple block, each bit of this bitmap corresponds to a value of this tuple. If it's null, we just set corresponding bit to 0, and there is no need for us to store its offset and length. If it's not null and its length is variable, we set corresponding bit to 1, and store its offset, length and value in block. If it's not null and its length is fixed, we set corresponding bit to 1, and store its value in block.

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.

Answer:
advantages:

1. We can do faster sequential scan in a single relation by its clustered index, and some strategies (like prefetching) will work better in this case.
2. It's much easier for us to implement the DBMS since file structure is simpler.

disvantages:

1. When we do operation on multiple relations (like table join), our IO operations will be higher since different relations are stored in different disk blocks.
2. Our cost to maintain constraints on multiple tables (like foreign key cascade update or delete) will be higher.

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

advantages:

1. it's easier for us to do operation on multiple tables like join.
2. Our cost to maintain constraints on multiple tables (like foreign key cascade update or delete) will be lower.

disadvantages:

1. File structure for DBMS is more complex. It's harder for us to

implement it.

2. It's harder to do a sequential scan in single relation, and we may need to use pointer to connect tuple in a single page, which is more difficult to be implemented.