CS-245 Database System Principles – Winter 2004
Assignment 2

Due at the beginning of class on Thursday, January 22

- State all assumptions and show all work..
- Check http://www.stanford.edu/class/cs245/announcements.html for clarifications.
- You can email questions to cs245-staff@cs.stanford.edu

A relational database system holds three relations: *C* (companies), *P* (products) and *M* (models) with the following characteristics:

Relation *C* (company):

- Tuples are stored as fixed length, fixed format records, length 250 bytes.
- There are 15,000 *C* tuples.
- Tuples contain key attribute *C.N* (company number), length 20 bytes; other fields and record header make up rest.

Relation *P* (product):

- Tuples are stored as fixed length, fixed format records, length 200 bytes.
- There are 45,000 *P* tuples.
- Tuples contain attribute *P.N* (the company number who makes the product), length 20 bytes; other fields and record header make up rest.
- Tuples also contain attribute *P.I* (product identifier), length 25 bytes.

Relation *M* (model):

- Tuples are stored as fixed length, fixed format records, length 100 bytes.
- There are 135,000 *M* tuples.
- Tuples contain attribute *M.I* (the identifier of the product involved), length 25 bytes, and an attribute *M.P* (the price of the product), length 10 bytes; other fields and record header make up rest.

While the number of products associated with each company varies, for evaluation purposes we may assume that each company has 3 products, and each product has 3 model records associated with it. Thus, you can assume that there are 9 model records for each company record.

The records are to be stored in a collection of 4 kilobyte (4096 bytes) disk blocks that have been reserved to exclusively hold *C*, *P*, or *M* records, or combinations of those records, and indexes over the records. (That is, there are no other types of records in the blocks we are discussing in this problem.) Each block uses 50 bytes for its header; records are not spanned.

Two disk organization strategies are being considered:

1. Sequential
   - All the company (*C*) records are placed sequentially (ordered by company number) in one subset of blocks.
   - Product (*P*) records are separate in another set of blocks. Products are ordered by company number.
   - Finally, model (*M*) records are in a third set of blocks, ordered by product identifier.

2. Clustered
   - For each company (*C*) record, the 3 products for that company (*C.N = P.N*) reside in the same block.
   - Similarly, the 9 model records for each company reside in the same block as their company record .
   - The company records are sequenced by company number.

Problem 1. (20 points)

For each storage organization, compute the total number of disk blocks required to hold all three relations.

Problem 2. (10 points)

Imagine that we are told there are two main query types:
   - *Table scan*: scan all of the company (*C*) records in order by company id.
   - *Join*: For a given company number *C.N*, get the company record followed by all its model records. That is, get all model (*M*) records with *M.I = P.I* for each *P* record with *P.N = C.N*.

Which storage organization would you prefer for each query type? Why?

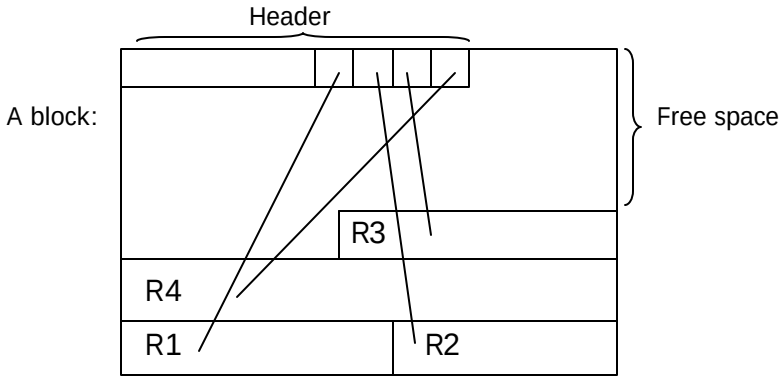
Problem 3. (35 points)

Let us now examine some indexing options. Imagine that you want to build a primary index on *C.N*, and a secondary index on *M.P*. Each index will associate with each key a 10 byte pointer to data (an 8 byte block id and a 2 byte offset). For each storage organization (sequential or clustered) and index type (primary and secondary), compute the number of 4096 byte blocks needed for the sparse index, and the number of 4096 byte blocks needed for a dense index. (Recall that 50 bytes of each block are used as a header.) If a particular type of index does not make sense, state so and explain why. Treat duplicates in the secondary index in a straightforward way; that is, for each *M* tuple there should be an index entry, even if there are multiple entries with the same key. Construct each index to use the minimum space possible without spanning index entries.

Problem 4. (15 points)

For the dense secondary index of problem 3 (on *M.P*) you decide that you want to create a bucket implementation to handle duplicate keys. Duplicate keys will be managed by inserting a single copy of the key into the secondary index, and associating with this key a 10 byte pointer to the beginning of a list of data pointers for that key. The lists of data pointers for the keys are packed into a sequence of 4096 byte "bucket" blocks (50 bytes of each block is still used as a header). Each block may contain a list for multiple keys; that is the pointer lists for the values "$10,000," "$11,000," "$12,000" and so on may coexist in the same bucket. Similarly, a list for one value may span multiple blocks; e.g. the list for "$13,000" may begin in one block and end in another. Assume that there are 10,000 unique *M.P* values. How many bucket blocks are needed, for each storage organization (sequential and clustered)? What is the new size of the dense secondary index under each storage organization? Construct the index and buckets to use the minimum space possible without spanning index entries or pointers.

Problem 5. (20 points)



Assume we are using the "indirection-in-block" approach (shown above) for storing records in each block. Suppose that we have 4096-byte blocks in which we store fixed-length records of 40 bytes. The block header consists of an 8-byte pointer to the "next" block and an offset table, using 2-byte pointers to records within the block. On an average day, three records per block are inserted, and two records are deleted. A deleted record must have its pointer replaced by a "tombstone," because there may be dangling pointers to it. For specificity, assume the deletions on any day always occur before the insertions. If the block is initially empty, after how many days will there be no room to insert any more records?