

CSE541 – Assignment 2

Assigned: Jan 18th, 2006
Due: Wed, Feb 1th, 2006

Purpose:

1. To improve your understanding of implementation issues for supporting indexes (through exploration of mySQL InnoDB).
2. To test your understanding of some multi-dimensional indexing approaches.

Deliverables:

Answer the questions in the last section.

The InnoDB storage engine provides extended functionality over myISAM tables. Some of its features (like support for crash recovery, and row-level locking for increased performance) we will explore later. For this assignment, you will focus on how InnoDB has implemented indexing, and what that means to the storage/performance trade-off.

The deliverables for this assignment include one analytical question, and answering some questions about how InnoDB implements some features. It's up to you how much time you spend exploring InnoDB. The more you play with the engine, the better you'll understand the trade-offs of its implementation.

Activities:

Throughout the experiments with InnoDB, use `SHOW INNODB STATUS` to examine buffer pool usage, disk reads and writes, adaptive hash statistics, etc.

- (a) Start your server. Create an InnoDB table and load it with the data from assignment 1. Create a primary (btree) index on the description field (whatever you called it). Execute a series of queries to access ranges of the data. Did InnoDB create an adaptive hash? What kind of buffer pool hit rate did you get? How many OS file reads and writes were done?
- (b) How big is the table and index when you have an index on the description field? Why? What happens if you add a secondary index on the price?
- (c) Stop and restart the server (you want to start with a clean buffer pool). Create a new table for the data, this time with a primary index on the ID, and a secondary index on the price. How big are your table and index? When you execute the same queries as before, do you get the same buffer pool hit rate? Why or why not? How about OS file reads and writes?
- (d) What happens to your OS file reads/writes if you delete a bunch of records, add a bunch of records, and/or edit the descriptions of records when you have the primary index on the description field?

Questions (answers to be handed in):

1. Multi-dimensional indexes

Consider a kd -tree that is perfectly balanced. If we execute a partial-match query in which one of the two attributes has a value specified, we will have to look at about \sqrt{n} out of the n leaves.

- (a) Why is this true?
- (b) If the tree split alternately in d dimensions, and we specified values for m of those dimensions, what fraction of the leaves would we expect to have to search
- (c) How does the performance of (b) compare with a partitioned hash table?

2. InnoDB index storage

- (a) For activity (a) above, where you created an InnoDB index on the description field, how much space does the index take for a single record? State any assumptions you make, and explain how you calculated the size. Include any space needed for headers, etc.
- (b) Suppose you added a secondary index on the price field. How much space does a single record in this secondary index take? Why?
- (c) Under what circumstances would the InnoDB storage formats for indexes be particularly efficient? What are the drawbacks to this format?