DB101 – Quizzes

Goetz Graefe – Madison, Wis.

Quiz on sorting

1. The course themes are transactions, storage formats, and query processing – which ones can use sorting, and how?
2. Hollerith's 1890 machine for counting and sorting – did it use merge sort or distribution sort?
3. What are the 2 or 3 phases of external merge sort?
4. When do you recommend quicksort, when merge sort?
5. How deep is a binary heap with capacity N?
6. The traditional tree-of-losers priority queue is particularly suited for which context or application?

Quiz on sorting

1. The course themes are transactions, storage formats, and query processing – which ones can use sorting, and how?
   * *transactions: sort log records for single-phase restore*
   * *storage: sort future index entries, sort for compression*
   * *queries: sort rows for 'join', 'distinct', 'group by', 'intersect'*

2. Holleriths 1890 machine for counting and sorting – did it use merge sort or distribution sort?
   distribution sort

Quiz on sorting

3. What are the 2 or 3 phases of external merge sort?

   input & run generation, final merging & output, intermediate merge steps if required

4. When do you recommend quicksort, when merge sort?

   quicksort: internal sort, keys of near-uniform distribution

   merge sort: external sort, keys with skew, long keys

Quiz on sorting

6. How deep is a binary heap with capacity N?
   $log_2(N)$

7. The traditional tree-of-losers priority queue is particularly suited for which context or application?
   merging sorted runs, including run generation by sorting "sorted runs" of a single record

Quiz on lock durations

1. What is the “gold standard” for correctness in concurrency?
2. In concurrency control, what is an action, what a transaction?
3. In database concurrency control, what are some differences between latching and locking?
4. In traditional locking schemes, when are locks acquired and when are they released?
5. In controlled lock violation, what constraint or “control” is imposed on a violating transaction?
6. In deferred lock enforcement, which conflicts are detected immediately and which ones are deferred?

Quiz on lock durations

1. What is the “gold standard” for correctness in concurrency?
   equivalence to a serial execution, preferably the same sequence as commit log records in the recovery log

2. In concurrency control, what is an action, what a transaction?
   action: a single method invocation, or similar
   transaction: user programmed script of actions, executed as a unit with ACID guarantees

Quiz on lock durations

3. In database concurrency control, what are some differences between latching and locking?
   latches coordinate ***threads*** *to protect* ***in-memory data structures*** *during* ***critical sections;*** *locks coordinate* ***transactions*** *to protect logical* ***database contents*** *during entire user-defined* ***transactions***

Quiz on lock durations

4. In traditional locking schemes,
   when are locks acquired and when are they released?
   before (first) access, *after commit including hardening (i.e., writing commit log record to recovery log on stable storage)*

Quiz on lock durations

5. In controlled lock violation, what constraint or “control” is imposed on a violating transaction?
   violated S locks: completion dependency
   violated X locks: commit dependency

6. In deferred lock enforcement, which conflicts are detected immediately and which ones are deferred?
   detected immediately: ww conflicts
   deferred to commit logic: rw & wr conflicts

Quiz on lock sizes

7. What are the differences between “read committed”, “repeatable read”, and “serializable” transaction isolation?
8. When locking preserves the absence of a key value, what is actually locked in the different locking schemes?
9. Give examples of false conflicts in the contexts of
   a. ~~controlled lock violation~~,
   b. ~~deferred lock enforcement~~,
   c. IBM’s key-value locking (ARIES KVL), and
   d. Microsoft’s key-range locking (KRL).

Quiz on lock sizes

7. What are the differences between “read committed”, “repeatable read”, and “serializable” transaction isolation?

   rc: instances come and go; no uncommitted “dirty” read

   rr: instances may appear, but won’t disappear once seen

   serializable: stable set of instances, “repeatable count”

Quiz on lock sizes

8. When locking preserves the absence of a key value, what is actually locked in the different locking schemes?
   ARIES/KVL: a distinct key value + a gap
   ARIES/IM: a logical row, all its index entries + gaps
   KRL: an index entry (only one if duplicates exist) + gap
   orthogonal KRL: a gap between index entries
   orthogonal KVL: a gap between distinct key values, or just a partition within such a gap

Quiz on lock sizes

9. Give examples of false conflicts in the contexts of ~~controlled lock violation~~, ~~deferred lock enforcement~~, IBM’s key-value locking (ARIES KVL), and Microsoft’s key-range locking (KRL).
   both *ARIES/KVL and KRL: one transaction requires* ***phantom protection*** *in a gap between existing key values, another transaction* ***fetches existing key values***

Quiz on logging and recovery (1 of 2)

1. [2] Why do databases use “write-ahead” logging?
2. [1] When are transaction updates guaranteed persistent?
3. [6-8] Name 3-4 classes of failures; outline their recovery.
4. [1] Define system availability using MTTF and MTTR, i.e., mean time to failure and mean time to repair.
5. [3] In system restart, when are new checkpoints possible, when are new user transactions possible?
6. [2] Outline log archiving for single-phase restore and for instant restore.

Quiz on logging and recovery (2 of 2)

7. [bonus +2] In class, we saw an example recovery log with a log record “written page … with PageLSN …”. How is this log record useful after a system failure?

Quiz on logging and recovery

1. [2] Why do databases use “write-ahead” logging?

   to save log records before overwriting database contents and to ensure rollback (“undo”) if necessary

2. [1] When are transaction updates guaranteed persistent?

   when the transaction’s commit log record is in the recovery log on stable storage

Quiz on logging and recovery

3. [6-8] Name 3-4 classes of failures; outline their recovery.

   * *transaction failure* → *rollback (linked list of log records)*

   * *system failure* → *restart (log analysis, redo, undo)*

   * *media failure* → *restore (backup, log replay)*

   * *page failure* → *repair (2nd linked list of log records)*

4. [1] Define system availability using MTTF and MTTR, i.e., mean time to failure and mean time to repair.

   MTTF / (MTTF + apparent MTTR)

Quiz on logging and recovery

5. [3+1] In system restart, when are new checkpoints possible, when are new user transactions possible?

   checkpoints & new transactions <u>at the same time</u>:

   ** traditional restart → after "undo", i.e., all recovery*
   ** optimized ARIES → after "redo", i.e., a predictable time*
   ** instant restart → after log analysis*
   i.e., <u>after recovery of all server state</u> (tx, lock, buf mgrs)

Quiz on logging and recovery

6. [2] Outline log archiving for single-phase restore and for instant restore.

   * *single-phase restore ⇐ sorted log records*
   * *instant restore ⇐ indexed log records*

7. [+2] How is a log record “written page… with PageLSN…” useful after a system failure?

   Log analysis removes the page from its “in-doubt” list, i.e., pages possibly dirty in the buffer pool during the crash.

Quiz on storage formats

1. [3] Name (at least) 3 methods to speed up large scans (of the 5 methods discussed in class).
2. [2] What is the principal function of all database indexes?
3. [3] Name (at least) 3 areas in which databases provide more functionality than typical key-value stores.
4. [2] Name advantages of traditional hash indexes over b-trees on hash values, and vice versa.
5. [2] External merge sort wants high merge fan-in; what other considerations apply to merging in LSM-forests?
6. [2] In write-opt'd b-trees and in foster b-trees, how do fence keys allow continuous comprehensive consistency checks?

Quiz on storage formats

1. [3] Name (at least) 3 methods to speed up large scans (of the 5 methods discussed in class).
   3 of: columnar storage, large transfers (I/O pages), parallelism, compression, zone filters
2. [2] What is the principal function of all database indexes?
   Mapping keys to values (e.g., rows, row identifiers, etc.)

Quiz on storage formats

3. [3] Name (at least) 3 areas in which databases provide more functionality than typical key-value stores.
   Schema, integrity constraints, complex transactions, physical data indep (index selection & maintenance), privacy, security
4. [2] Name advantages of traditional hash indexes over b-trees on hash values, and vice versa.
   Direct single-page access – fast creation & maintenance, phantom protection by locking gaps, less code overall

Quiz on storage formats

5. [2] External merge sort wants high merge fan-in; what other considerations apply to merging in LSM-forests?
   Many runs (partitions) waiting to be merged increase effort and latency in query execution
6. [2] In write-opt'd b-trees and in foster b-trees, how do fence keys allow continuous comprehensive consistency checks?
   Root-to-leaf traversals check key ranges and solve the "cousin problem"

Quiz on query processing (1 of 3)

1. Can a b-tree index on columns (A,B,C) support …
   a. [2] … a query predicate “B=47”? If yes, how?
   b. [1] … a scan “order by B, C”? If yes, how?
   c. [1] … a scan “order by B, C, A”? If yes, how?
2. What is the relationship between …
   a. [2] … algorithms for “join” and “intersect”?
   b. [2] … algorithms for “group by” and “distinct”?
   c. [2] … distribution sort and hash-partitioning (assume single-threaded algorithms)?

Quiz on query processing (2 of 3)

3. What are interesting orderings, e.g., of merge join and…
   a. [1] … storage structures, e.g., b-tree indexes?
   b. [1] …other operators in a plan, e.g., another merge join?
4. [2] When is hash join the most efficient join method?
5. [2] When is lookup join the efficient alternative?
6. [2] What are “covering indexes” and “index-only retrieval”?

Quiz on query processing (3 of 3)

7. [2] Why is cardinality estimation difficult and error-prone?
8. [2] How did query optimization in System R use dynamic programming?
9. [1] What is index intersection?
10. [2] When and why should it be used?

Quiz on query processing

1. Can a b-tree index on columns (A,B,C) support …
   a. [2] … a query predicate "B=47"? If yes, how?
      Yes, by enumerating all values of A.
   b. [1] … a scan "order by B, C"? If yes, how?
      Yes, by merging |A| runs sorted on B,C.
   c. [1] … a scan "order by B, C, A"? If yes, how?
      Yes, by (stable) merging.

Quiz on query processing

2. What is the relationship between …
   a. [2] … algorithms for “join” and “intersect”?
   Same algorithms, different predicates & output columns
   b. [2] … algorithms for “group by” and “distinct”?
   Same algorithms, different output keys & columns
   c. [2] … distribution sort and hash-partitioning?
   Hash-partitioning is distribution sort on hash values with near-uniform distribution of short (integer) keys

Quiz on query processing

3. What are interesting orderings, e.g., of merge join and…
   a. [1] … storage structures, e.g., b-tree indexes?
      A b-tree on the join keys permits merge join without sort.
   b. [1] …other operators in a plan, e.g., another merge join?
      Two merge joins on the same columns save sorting the intermediate result.
4. [2] When is hash join the most efficient join method?
   If the smaller input fits in memory and the larger input is unsorted.

Quiz on query processing

5. [2] When is lookup join the efficient alternative?
   When the outer input is small and the inner input is indexed.
6. [2] What are “covering indexes” and “index-only retrieval”?
   An index “covers” a query and thus permits “index-only retrieval” if the index holds all columns the query needs.

Quiz on query processing

7. [2] Why is cardinality estimation difficult and error-prone?
   No summary statistics (histograms, sketches, etc.) can describe all real-world data distributions.
8. [2] How did query optimization in System R use dynamic programming?
   It first built single-table plans, then two-table plans from those, then three-table plans from those, etc.

Quiz on query processing

9. [1] What is index intersection?
   Each index produces a list of row identifiers, row ids in the intersection of two or more such lists satisfy the entire conjuctive query.
10. [2] When and why should it be used?
    Multiple indexes together may reduce the result set so much that fetching the remaining few rows is very cheap.