Final Exam Study Guide CSC 321/621 – Spring 2012

Graduate students will have one or more additional questions (in the same amount of time)

I would review all of the "practice problems" that popped up in lecture – usually written with red italics in the slides or explicitly labeled on slide as a "Practice Problem".

The test will likely be quite "problem-oriented" ("prove this fragmentation is valid", "optimize this query using these four transforms") and "argument-oriented" ("Bob is considering to use a hash-based physical representation of this relation. Here are the likely queries to be employed on that relation. Argue for or against Bob's choice.").

Key areas to focus on (material on the test will only come from this list):

- What a view is; how a view is defined; pros and cons of using views; pros and cons of the two implementation approaches for views; notion of view maintenance
- Stored procedures and functions; you may be asked to interpret and read a stored procedure (but not write one); pros and cons of using stored procedures
- The notion of triggers; when triggers can be 'triggered'; utility of using triggers
- The notion of a transaction; deep meaning of the terms commit and rollback; the ACID properties of databases w/ respect to transactions
- The notion of concurrency control and when it is needed with respect to transactions; ability to recognize and discuss why concurrency control is needed (such as the ATM example)
- The two types of locks used in concurrency control (read/shared; write/exclusive); when locks can be granted; terminology related to schedules, including serial and serializable; what 2PL is and how it works
- Two types of localized failure (affecting memory, affecting disk); the relationship between transactions and recovery; the notion of a "database buffer" and how buffers are managed; the notions of checkpoints and logfiles – how they are used, what they represent, and their purpose; immediate update vs deferred update and effects on recovery of using the different update styles
- Familiarity with the "shared nothing" architectural model for databases; advantages and disadvantages of distributed databases
- The notion of relation fragmentation; the four types of fragmentations used on database schemas (horizontal, vertical, mixed, derived); the rules for how fragmentation should be constructed (the 3 principles); the notion of replication of fragments; the pros and cons of fragmentation and replication in terms of access and maintenance; the four distributed transaction types
- How transactions can be distributed as sub-transactions; extension of 2-phase locking to support concurrency (including replicates) – there are three extensions we studied!

- Possible sources of failure in a distributed database management system; commitment management and recovery in a distributed database using 2-phase commits (2PC) make sure you understand the stages of the 2PC protocol and how failures can affect the various stages and how recovery should work after failure in various stages
- The meaning of a page in a physical representation of a database, and how pages and tables are related; hashing based search with overflow chaining (you can ignore the other approaches to hash overflow management); compare and contrast of hashing & indexing
- The meaning of an index, types of indices (primary, clustering, secondary) and how they are defined; advantages of using indices, costs of maintaining indices, the B+-implementation of indices; how to search and maintain a B+-tree index
- Goals of query optimization; given a cost model and a query, ability to compute the relative costs of alternative groundings of a query; ability to determine whether alternative groundings of a query are equivalent; tree representations of queries; goals of heuristic optimization approaches on tree representations of queries; given a list of query transformation rules, be able to apply them to a query tree to improve the query w/ respect to efficiency

Things you will NOT have to do/know:

- Do not worry about the SQL syntax for creating indices, creating views, etc. – the particular syntax issues are something you could look up in the real world.
- Writing through views, migrating rows in a view
- Database access control
- You will not be asked to write a stored procedure
- You do not need to memorize the meaning of the terms “lost update problem”, “uncommitted dependency problem”, “inconsistent analysis problem”, but you should be aware of the general reasons why they can be problematic
- The low-level details behind lock implementations (how they are stored… extra bits, extra tables)…
- The proof of why 2PL works
- You do not need to memorize the data storage hierarchy, but you should understand why it is the goal to do as much in memory as possible without going to disk
- The details of log-file management so that log files don’t get out of control (writing to two log files, etc)
- Isolation levels
- The various “parallel architectures” except for “shared nothing”
- Homogeneous vs. heterogeneous databases
- Distributed naming transparency
- The master/slave and P2P implementations of replication
- Heap-based table representations, dynamic/extendable hashing, any hashing-overflow technique not explicitly listed in the previous list of topics to know
- Computing the number of items that can be held in a B+ tree of a certain size
- Bitmap and join indices

- Do not memorize the transformation rules used in query optimization – they will be given to you; you should, however, understand how and why they are used and be able to use them
- The rigorous, low level cost analysis (binary searching is O(…), hash indexing is O(…)) discussed on the last day will not be on the test