IT-344 Test Bank

Midterm

17.1 Which of the following is NOT primary storage?

a) register

b) cache

c) main memory

d) CD ROM

17.2 The time required to position the read/write head over a record to be read is called the \_\_\_\_ time

a) seek

b) block transfer

c) rotational time

d) Greenwich Mean

17.3 Which of the following are new storage systems

a) NAS

b) SANS

c) iSCSI

d) All of the above

e) two of the above

Chapter 18

18.1 A secondary index can be specified on \_\_\_\_\_\_\_\_\_.

a) any field of a file

b) any non-ordering field of a file

c) a primary key of the file

d) an ordering field of the file

18.2 A clustering index differs from a primary by \_\_\_\_\_\_.

a) the values the file is ordered on are not unique

b) being dense

c) being faster

d) more efficient

18.3 Which of the following data structures is used for multilevel indexes?

a) linked list

b) graph

c) B-tree

d) stack

Chapter 19

19.1 The first step in query processing is \_\_\_.

a) scan the query

b) use heuristic optimization

c) compute the cost of the operations

d) determine the execution plan

19.2 Which search method is the simplest to code for implementing the select command?

a) Using a hash key

b) linear search

c) binary search

d) using a primary index

19.3 What is the first step in a heuristic optimization of a relational algebra query tree?

a) Cascade of project

b) Commuting project with select

c) Commutating the project operation

d) Cascade of select

Chapter 20

20.1 Which of the following is NOT an important factor influencing physical database design?

a) analysis of the database queries

b) analysis of the frequency of queries and transactions

c) number of users

d) analysis of the uniqueness constraints on attributes

20.2 An example of denormalization of a relation is \_\_\_\_\_\_.

a) changing a relation from first to second normal form

b) changing a relation from third to second normal form

c) changing a relation from second to third normal form

d) all of the above

20.3 Which of the following is NOT a reason for revising the initial choice of indexes?

a) Certain indexes are on non-key fields

b) certain queries may take to run because they don’t use an index

c) certain indexes aren’t utilized

d) certain indexes may undergo too much updating

Chapter 21

21.1 A transaction is a \_\_\_\_ unit of work that must be either entirely completed or aborted.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | timed | c. | logical |
| b. | practical | d. | physical |

21.2 All transactions must display \_\_\_\_.

|  |  |
| --- | --- |
| a. | atomicity, consistency, and durability |
| b. | durability and isolation |
| c. | consistency, durability, and isolation |
| d. | atomicity, durability, consistency, and isolation |

21.3 As long as two transactions, T1 and T2, access \_\_\_\_data, there is no conflict, and the order of execution is irrelevant to the final outcome.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | shared | c. | unrelated |
| b. | common | d. | locked |

Chapter 22

22.1 In the optimistic approach, during the \_\_\_\_ phase, changes are permanently applied to the database.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | read | c. | write |
| b. | validation | d. | shared |

22.2 The \_\_\_\_ approach to scheduling concurrent transactions assigns a global unique stamp to each transaction.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | scheduled | c. | unique |
| b. | table-locking | d. | timestamping |

22.3 Lock \_\_\_\_ indicates the level of lock use.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | granularity | c. | growing |
| b. | shrinking | d. | serializability |

Final Exam

Chapter 23

23.1 The information stored in the \_\_\_\_ is used by the DBMS for a recovery requirement triggered by a ROLLBACK statement, a program’s abnormal termination, or a system failure such as a network discrepancy or a disk crash.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | data dictionary | c. | rollback manager |
| b. | metadata | d. | transaction log |

23.2 One of the entries in the system or transaction log is a list of all active transactions which is called a\_\_\_?

a. checkpoint

b. core dump

c. roadblock

d. timestamp

23.3 The \_\_\_\_\_ points to the most recent database pages on disk.

a) shadow directory

b) root directory

c) current directory

e) all of the above

Chapter 24

24.1 \_\_\_\_deals with ensuring that data is protected against unauthorized access, and if the data are accessed by an authorized user, that the data are used only for an authorized purpose.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | Integrity | c. | Availability |
| b. | Compliance | d. | Confidentiality |

24.2 “All users must have passwords” is an example of a \_\_\_\_.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | rule | c. | standard |
| b. | policy | d. | procedure |

24.3 A \_\_\_\_ is a named collection of database access privileges that authorize a user to connect to the database and use the database system resources.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | user | c. | profile |
| b. | role | d. | manager |

Chapter 25

25.1 A distributed database is composed of several parts known as database \_\_\_\_.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | sections | c. | partitions |
| b. | fragments | d. | Parts |

25.2 \_\_\_\_ distributed database systems integrate only one type of centralized DBMS over a network.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | Homogeneous | c. | Fully heterogeneous |
| b. | Heterogeneous | d. | Combination |

25.3 A(n)\_\_\_\_ database stores each database fragment at a single site.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | partially replicated | c. | fully replicated |
| b. | unreplicated | d. | partitioned |

Chapter 26

26.1 Which of the following is NOT part of the generalized model for active database.

a. event

b. condition

c. cascade less

d. action

26.2 Relations that support an associated time when the event is true are called

a. bi-temporal

b. valid time

c. transaction time

d. two of the above

26.3 Which of the following is NOT a feature of image analysis?

a. size

b. color

c. texture

d. shape

Chapter 28

28.1 Which if the following is a goal of data mining?

a. generating reports

b. retrieving facts

c. predicting behavior

d. finding erroneous data

28.2 An association rule is written in the form.

a. R(a,b)

b. a=>b

c. a|b

d. a=b

28.3 An algorithm for finding large itemsets is called \_\_\_\_\_\_ algorithm?

a. Cromardi’s

b. fast find

c. buchberger’s

d. apriori

Chapter 29

29.11 Data is captured from the OLTP system and placed on the \_\_\_\_ on a near-real time basis.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | decision support system | c. | data warehouse |
| b. | portal | d. | dashboard |

29.2 The basic star schema has four components: facts, \_\_\_\_, attributes, and attribute hierarchies.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | keys | c. | cubes |
| b. | relationships | d. | dimensions |

29.3 A(n) \_\_\_\_is a dynamic table that not only contains the SQL query command to generate the rows, but also stores the actual rows.

|  |  |  |  |
| --- | --- | --- | --- |
| a. | OLAP | c. | star schema |
| b. | materialized view | d. | data cube |

**IT 344 – Database Management Systems Test Bank**

**Chapter 17 – Disk Storage**

**17.1** If you have data that should not be lost on disk failure, and the data is write intensive, how would you store the data?

**17.2** If the slotted page architecture is used for storing tuples, can indices store pointers to actual tuples? If not, why not, and what do they store instead?

**17.3** Disk capacities have been growing very fast in the last few years, and cost per byte has been falling equally fast. However, access times (latency/seek) have decreased at a much slower rate. Which of RAID 1 or RAID 5 does this trend favor, and why?

**17.4** Describe one benefit of each of RAID 5 and RAID 1 (mirroring) relative to each other.

**17.5** In disks in the past, the number of sectors per track was the same across all tracks. These days, disks have more sectors per track on outer tracks, and fewer sectors per track on inner tracks (since they are shorter in length). What is the impact of such a change on each of the three main indicators of disk speed?

**17.6** Consider a RAID system with block striping (e.g. RAID 5) and a workload with many small I/Os and one long sequential read running concurrently.

**a.** How would the transfer rate for the sequential read on a RAID system with 8 disks compare with reading the data from a single dedicated disk, if one block is read at a time?

**b.** Suggest how to perform the read to improve the transfer rate from the RAID system.

**17.7** In a slotted page architecture, objects (tuples) are accessed indirectly via a header table at the beginning of the page. That is, a tuple id gives a page-id and an offset in the header table. Each entry in the table consists of a pointer (offset in the page) to an object. Describe the main advantage of this architecture over one where tuple-ids has a page-id and a direct offset to the tuple.

With the above scheme, the table at the beginning of the page can have “holes” corresponding to deleted tuples. Describe a small modification to the scheme, so that the table does not have to store any holes.

**17.8** (Buffer management)

**a.** Standard buffer(cache) managers assume each page is of the same size and costs the same to read. Consider a cache manager that, instead of LRU, uses the rate-of-reference to objects, i.e., how often an object has been accessed in the last n minutes. Suppose you wanted to cache objects of varying sizes, and varying read costs (e.g., web pages, whose read cost depends on the site you are fetching from).

**b.** Suggest how one may decide which page to evict.

**Chapter 18 – Indexing Structures for Files**

**18.1** Why are B-trees, which have a very large fanout (i.e., number of children), preferred over binary trees (having at most two children) in the context of disk- based databases?

**18.2** What is the maximum height of a B+ -tree with N keys and a maximum fanout of k.

**18.3** Suppose you want to build a B+ -tree index on a CD-ROM, where the trans- fer rates are comparable to disk transfer rates, but seek times are much higher. Should the block size be the same as on disk, higher or lower? Explain intuitively (in one or two sentences) why.

**18.4** Name two factors that affect the decision of the page size of a B+ -tree (e.g., should the page be 1KB or 16 KB).

**18.5** Give three alternative ways of evaluating a selection of the form σA=$1∧B=$2(r)

if secondary indices are available on A and on B.

**18.6** Suppose I store two dimensional points as the location attribute of a relation, and queries ask for tuples whose location is at exactly a given point. What sort of index is best for this case?

**18.7** Can a secondary (unclustered) index be worse than a file scan for a select?

Describe a rough scenario.

**18.8** Suggest an easy way (other than using buckets of key values) to implement B+ - trees on non-unique search keys. Briefly outline how to find all records with a given search key value using such an index.

**18.9** The textbook description of static hashing assumes that a large contiguous stretch of disk blocks can be allocated to a static hash table. Suppose you can only allocate C contiguous blocks. Briefly sketch how to implement the hash table, if it can be much larger than C blocks. Access to a block should still be efficient.

**18.10** What should you do to make sure that the worst case occupancy of a B+ -tree is at least 75%?

**18.11** Suppose I have a relation with nr tuples on which I want to build a secondary B+ -tree.

**a.** Estimate the cost of building the B+ -tree index by inserting one record at a time. Assume each page will hold an average of f entries.

**b.** Suggest a more efficient way of constructing the index; just outline the key ideas, no need for a detailed description.

**18.12** Suppose you want to efficiently support queries of the following form

select sum(A) from r where v1 < r.B and v2 > r.B

where v1 and v2 are values supplied at runtime; the same query is invoked repeatedly with different values for v1 and v2. Describe how one could store extra information in the internal nodes of a B+-tree index on r.B to efficiently compute the above sum.

**18.13** In extendible hashing, if the occupancy of a bucket is too low, what bucket(s) are you allowed to merge the underfull bucket with?

**18.14** Suppose we insert the following records into an extendible hash structure in the order shown: (B,1), (D,1), (D,2), (M,1), (P,1), (P,2), (P,3). Assume that B hashes to (binary) 0000, D hashes to 0101, M to 0111 and P to 0111. Show the resultant extendible hash structure, assuming that each bucket can hold at most 2 pointers. Assume that suffixes of the hash value (as shown in binary above) are used to identify buckets.

**18.15** Suppose that we are using extendible hashing on a file containing records with the following search key values: 2,3,5,7,11,17,19,23,29

Show the extendible hash structure for this file if the hash function is h(x)

= x mod 8 and buckets can hold three records. Further, assume that we consider the suffix of the bit representation (i.e., the last global depth bits) to index into the directory. Also, assume buckets can hold three records. Show all the intermediate steps while populating the hash index. Assume initially the hash index consists of two entries in the directory each pointing to a distinct empty bucket. Suppose that the following modifications are applied to the hash index:

Delete 11, Delete 31, Insert 1, Insert 15.

The changes are cumulative – each change is applied before applying the next. Show the hash index after each change.

**18.16**

**a.** Outline the steps in building a B+ -tree bottom-up (that is, directly from a given relation without performing a series of inserts).

**b.** Compare the cost of bottom up build as above with building a tree by a sequence of inserts, in terms of I/O as well as in terms of space used.

**c.** Very briefly suggest how an R-tree can be constructed bottom-up. Your answer need not be technically complete, or optimal in any sense, just briefly outline an idea/heuristic

**18.17**

**a.** Is it always a good idea to have an index. In other words, what are the pros and cons for creating an index.

**b.** Given an SQL query of the form select L from r1 , . . . rn where P where P represents a predicate, list indices that may potentially be useful for answering the query.

**18.18** Suppose access to some key values in a B+-tree is very common relative to others, that is the access pattern is highly skewed. Suggest a scheme for caching data at internal nodes of the B+ -tree to minimize access times. Assume that with each node you can allocate extra space for caching data.

**18.19** Describe an efficient algorithm to build a B+ -tree on a file which is sorted on the indexed attribute. Also, outline the cost of the algorithm in terms of I/O’s. Assume the file occupies N pages and the resulting B+ -tree will occupy M pages. Assume there are h + 2 buffer pages where h is the final height of the B+ -tree.

**Chapter 19 – Algorithms for Query Processing**

**19.1** Consider external sorting using replacement selection. What is the minimum, average, and maximum length of the runs generated? Explain when the mini- mum and maximum length runs are generated.

**19.2** Suppose you want to compute the join of relations A and B, where A fits in memory with plenty of space to spare, and B is much larger than memory. What join technique would you use?

**19.3** Very briefly outline how to extend merge-join to compute the full outer natural join.

**19.4** Describe how to compute full outer join by using an extension of the hash join algorithm.

**19.5** (Iterator model) Consider an iterator implementation of nested-loops join of relations a and b. Give pseudocode for the getnext() function.

**19.6** Pipelining is used to avoid writing intermediate results to disk. Suppose you are merge joining the results of two sorts. Describe how the output of the sort can be effectively pipelined to the merge join without being written back to disk.

**19.7** Explain how to implement A ✶ B using hashing, where A is used as the build relation.

**19.8** Explain how to implement the division operator, using sort-merge. (Recall that division of r(A, B) by s(B) finds values a for which (a, x) is in r for every

(x) ∈ s.)

**19.9** Suppose I want to provide an iterator interface to external sorting. Outline what should be done in the open(), getnext() and close() functions.

**19.10** When using the block nested loops join algorithm to join relations A and B, suppose memory has M blocks. How many blocks should be given to A and how many to B? What is the cost of the algorithm assuming A has ba blocks and B has bb blocks.

**19.11** Given a materialized view A ✶ B, how would you update it if some tuples are added to A; let iA denote the tuples added to A. .

**19.12** Suppose the memory size is M 4096 byte blocks. How big does a relation have to be to require recursive partitioning, if the average size of each partition must be 0.8 ∗ M blocks.

**19.13** What is the cost of finding the best left-deep join order and the cost of finding the best right-deep join order?

**19.14** Suppose you have a query that only wants the first ten results in sorted order of attribute A, of r ✶r.A=s.A s. Give conditions under which it suffices to use only the first ten results (in sorted order of A) of relations r and s to compute the result.

**19.15** What is the motivation behind the optimization heuristic ”Push selects through joins”.

**19.16** Let r and s be relations with the number of blocks in the two being br and bs respectively. Assume that there are no indices on the two relations and they are not sorted. Assuming infinite memory, what is the cheapest way (in terms of I/O operations) to compute r ✶ s and what is the amount of memory required for this algorithm.

**19.17** For each of the following pairs of queries, indicate under what conditions are they equivalent (i.e. they return exactly the same answers),

**a.** select distinct r.A from r and

select distinct r.A from r, s where r.B = s.B

**b.** select distinct r.A from r and

select distinct r.A from r, s

**19.18** Give examples to show non-equivalence of the following pairs of expressions.

Assume the relation schemas are A(X, Y ), B(X, Z ) and C(X, W ).

**a.** A ✶ (B ✶ C) and (A ✶ B) ✶ C? .

**b.** A ✶ (B ✶ C) and (A ✶ B) ✶ C?

**c.** A ✶ (B ✶ C) and (A ✶ B) ✶ C? (Hint: consider values common to A and C but not B.

**19.19** Why are the following equivalences incorrect; describe the reason either in words or through a small example. Assume that initially relations have no duplicates or nulls. (Some of the expressi**on**s below can generate tuples with nulls.)

**a.** ΠA (R − S) ↔ ΠA (R) − ΠA (S).

**b.** σB<4( A Gmax(B) (R)) ↔ A Gmax(B) (σB<4 (R)).

**c.** Does the above equivalence hold if max is replaced by min? Explain very briefly.

**d.** (R ✶ S) ✶ T ↔ R ✶ (S ✶ T ). ( ✶ denotes the natural left outer join).

(Hint: assume that the schemas of the three relations are R(a, b1), S(a, b2) and T (a, b3) respectively.)

**19.20** Consider

select r.A, r.B, r.C from r

where r.B = (select avg(s.B) from s where s.A=r.A)

Describe how to evaluate the above query efficiently using

**a.** Sorting of both r and s.

**b.** Caching, assuming there is a clustered index on s.A and V (B, r) is small compared to the size of memory.

**19.21** Suppose you have to compute A Gsum(C ) (r) as well as A,B Gsum(C ) (r). Describe how to compute these together using a single sorting of r.

**19.22** Consider a relation r(A, B, C), with an index on attribute A. Give an example of a query that can answered using the index only, without looking at the tuples in the relation.

**19.23** Suppose you want to get answers to r ✶ s sorted on an attribute of r, and want only the top n answers for some relatively small n. Give a good way of evaluating the query

**a.** when the join is on a foreign key of r referencing s.

**b.** when the join is not on a foreign key.

**19.24** Given two SQL queries of the for

select A1, . . . , An from R1, . . . , Rn where P1 groupby G1

and

select B1, . . . , Bm from S1, . . . , Sn where P2 groupby G2give conditions under which the result of the first query can be used to compute the second query.

(Note: Ri and Si is not the actual name of the relations, some Ri ’s may be the same as some Sj ’s, and similarly for the Ai ’s and Bi ’s; G1 and G2 are lists of groupby attributes and P1 and P2 are predicates.)

**19.25** Suppose you want to compute the results of all the following queries:

A,B,C Gcount(S) (r), A,B Gcount(S) (r), A Gcount(S) (r).

Outline an efficient sorting-based algorithm for computing them together.

**19.26** Suppose you want to compute a groupby on a relation, where you believe the number of groups is likely to be much smaller than memory, but aren’t 100% sure. Give an algorithm for computing the groupby, which works very well if the number of groups is small enough, and degrades in a smooth fashion if not.

**19.27** Suppose you have a huge relation and want to do a groupby and find groups whose count is greater than 1 percent of the size of the relation. The total number of groups may be very large, so you do not have enough memory to maintain a count for every group.

Suggest an efficient algorithm for doing so without sorting or partitioning the data – in fact the data may be read but never written. (Hint: Use some form of hashing where multiple groups may have the same hash value, to filter out many irrelevant groups).

**19.28** Consider a block nested loop join, where memory has M + 1 pages. Let the relations r and s have br and bs pages. Suppose we divide memory as follows:

one page for output, i pages for r and M − i pages for s.

**a.** What is the cost of block nested loop join of r and s, with r as the outer relation.

**b.** Based on your cost formula, what value of i gives the lowest cost.

**c.** If you could choose which of r and s is the outer relation, how would you choose it?

**19.29** Consider a join r ✶ s, where both r and s are pipelined in (that is, they are gen- erated by other operations, and are piped in a tuple at a time). Let the schemas be r(A, B) and s(A, C). Suppose both r and s, with indices on A, fit in memory.

Describe an algorithm that can generate tuples of r ✶ s without waiting for the entire input to be available. Structure your description by first giving the key idea and then more details.

**19.30** Consider the issue of interesting orders in optimization. Given a set of relations S to be joined, and a subset S1 of S, what are the interesting orders of S1?

**19.31**

**a.** Suppose the number of interesting orders for a query (and each of its inter- mediate relations) is “d”. Suppose also that there is only one join method available. Consider a join of 2n relations.

i. What is the number of different left-deep join trees.

ii. How many different plans does the System R optimizer actually examine.

iii. What is the maximum number of plans the optimizer has to store at any given point of time during the optimization phase?

(Hint: Recall that given a set of size n, the number of subsets of size i is n .)

i

**b.** Consider queries consisting only of joins (that is, no projections or selec- tions). Is it true that the System R algorithm chooses the “optimal” order amongst all join orders? Why? The “optimal” join order is the one with the minimum estimated cost.

**19.32**

**a.** Suppose the view v = r1 ✶ r2 has been materialized (computed and stored), and now a set of tuples i1 is inserted into r1 , and i2 into r2 . Assume you have the sets i1 and i2, as well as the contents of relations r1 and r2 after the insertion. Give an expression to compute the new tuples to be inserted into the view result computed earlier, in order to keep the materialized view up to date.

**b.** Suppose the view v =A Gsum B (r) has been materialized and then a single

tuple t is added to r. Describe how to keep the view up to date.

**c.** Suppose now that the view is v1 =A Gavg B (r). What extra do you have to store to allow the view result to be updated efficiently?

**19.33** Suppose you want to build a query optimizer can make use of materialized views. Disregard selections and projections, and assume queries as well as materialized views are just natural joins. Describe how to extend the System R style bottom-up join order optimization algorithm to handle materialized views.

**19.34** Suppose you are given a relation emp(empid, dept, salary) and wish to maintain a materialized view deptsalary(dept, totalsalary) which stores the total salary for all employees of each department. Suppose the system does not support materialized views but supports triggers.

**a.** Give triggers on insert and delete on emp to record changes into insert-delta and delete-delta relations. Don’t worry about updates.

**b.** Write a JDBC program to use the tuples in the delta relations to update the total salary. Be sure to handle new departments, and to remove tuples from the delta relation once they have been processed.

**NOTE 1**: For simplicity, don’t bother about departments that become empty (i.e. their last employee is deleted), but outline (in English, not SQL) how to detect and handle this case.

**NOTE 2**: Don’t bother to give code for setting up or deleting connections, statement objects etc.

**Chapter 20 – Physical Database Design and Tuning**

**20.1** What are the important factors that influence physical database design?

**20.2.** Discuss the decisions made during physical database design.

**20.3.** Discuss the guidelines for physical database design in RDBMSs.

**20.4.** Discuss the types of modifications that may be applied to the logical database design of a relational database.

**20.5.** Under what situations would denormalization of a database schema be used? Give examples of denormalization.

**20.6.** Discuss the tuning of indexes for relational databases.

**20.7.** Discuss the considerations for reevaluating and modifying SQL queries.

**20.8.** Illustrate the types of changes to SQL queries that may be worth considering for improving the performance during database tuning.

**Chapter 21 – Intro to Transaction Processing Concepts and Theory**

**21.1** Show a sequence of updates (withdrawal of money from an account, for ex- ample) by two concurrent transactions that results in an incorrect final state.

**21.2** Consider two transactions, one transfering Rs 50 from account A to account B and another transfering Rs 100 from account B to account A.

**a.** Show a schedule which is not serial, but is conflict serializable.

**b.** Show a schedule which is not conflict serializable and results in an inconsistent database state.

**21.3** Give an example of a schedule that is view serializable but not conflict seria izable. What is the essential feature of such schedules? . . .2+ 1

**21.4** Give the expansion of ACID.

**21.5** Give 2 reasons for allowing concurrent execution of transactions?

**21.6** Give an example of a serializable but not serial schedule (use only reads and writes, don’t include other operations). Explain why it is serializable.

**21.7** Give an example of a schedule that is serializable but not recoverable.

**Chapter 22 – Concurrency Control Techniques**

**22.1** Briefly explain what is recoverability and what is cascade freedom. Mention a protocol that ensures both properties, and explain why it ensures both the above properties.

**22.2** If many transactions update private items (eg. individual account balances) and a common item (eg. cash balance at a branch). What can you do to increase concurrency/throughput, without using any fancy locking modes, by ordering of operations in a transaction.

**22.3** Consider the following locking protocol: all items are ordered, and once an item is unlocked, only higher numbered item may be locked. Locks may be released at any time. Only X-locks are used.

Show by an example that this protocol does not guarantee serializability.

**22.4** Mention one locking protocol which ensures that deadlocks never occur, with- out ever aborting transactions. Give one major drawback of using the protocol.

**22.5** Is every conflict serializable schedule potentially generatable by two-phase locking? If not, give an example.

**22.6** (Locking)

**a.** What defines the serialization order of transactions if two-phase locking is used?

**b.** What are the minimal changes to two-phase locking to guarantee

i. recoverability (and explain why it works)

ii. cascade freedom (and explain why it works)

**22.7** (Deadlock prevention)

**a.** Outline the wound-wait and the wait-die schemes for deadlock prevention.

**b.** The wound-wait and wait-die are not used much in practise. Briefly ex- plain why this may be the case.

**22.8** (TSO Protocol)

**a.** Outline the TSO protocol for reads

**b.** Outline the TSO protocol for writes (including the Thomas write rule)

**c.** Outline how to modify the TSO protocol to guarantee recoverability and cascade freedom.

**22.9** The Oracle database implements a special concurrency control protocol for read-only transactions, whereby a read-only transaction sees a view of the database containing effects of all transactions that committed before it starts. Assume that updates are immediate, and only physical logging is used. Explain how to use undo logs to efficiently provide such a view to read-only transactions.

**22.10** In multigranularity locking, give the compatibility matrix for the different lock modes (the two normal modes plus the three intention lock modes). Also give an explanation of line each for those entries of the matrix that involve two intention lock modes.

**22.11** Show that given any schedule generated by two-phase locking, the transaction dependency graph is acyclic (thereby you would have shown that two-phase locking generates only conflict serializable schedules).

**22.12** Consider the validation based concurrency control protocol you have studied (which is also known as the optimistic concurrency control protocol). Outline the checks that a transaction must perform during validation.

**22.13** Consider multi-version timestamp ordering based concurrency control. Give conditions under which a version will definitely not be used again (and can thus be garbage collected).

**22.14** Recall the tree locking protocol: the first lock can be on any node of the tree, and subsequently nodes can be locked only if the parent is locked. Locking is not required to be two phase, but still guarantees conflict serializability.

**a.** Consider the following “dumb forest locking protocol”. A forest is a col- lection of trees. The dumb forest locking protocol uses the tree locking pro- tocol independently on each tree. As before locking is not required to be two-phase. Show, using an example that this protocol does not guarantee conflict serializability.

**b.** Describe a simple extension of the above locking protocols that can be used on a forest.

**22.15** Multi-version 2PL combines features of 2PL and multiversion concurrency con- trol: updaters use locking to see the latest version of data, while read only transactions use old versions of data based on timestamps. Describe the scheme briefly. Make sure you cover the points below, and any others that are important.

**a.** What are the actions of read-only transactions on begin/read/commit?

**b.** What are the actions of update transactions on begin/read/write/commit?

**22.16** Suppose a set of items forms a directed acyclic graph (DAG). Show that the following protocol assures conflict serializability.

**a.** The first lock can be on any node

**b.** Subsequently, a node n can be locked only if the transaction holds a lock on at least one predecessor of n, and the transaction has locked each predecessor of n at some time in the past.

**Chapter 23 – Database Recovery Techniques**

**23.1** What information do undo log records contain, and what are they used for?

**23.2** (Recovery)

**a.** Copying the page table in shadow paging can be quite expensive for very large databases. Suggest an implementation of page tables that can reduce the copying required (assuming transactions are small).

**b.** Give two reasons (other than the above) why shadow paging is not used in databases today.

**c.** In the advanced recovery algorithm, explain why an action is never (successfully) undone more than once, regardless of failures during rollback.

**23.3** (Aries)

**a.** If at the beginning of the analysis pass, a page is not in the checkpoint dirty page table, will we need to apply any redo records to it? Why?

**b.** What is RecLSN, and how is it used to minimize unnecessary redos?

**c.** What problem could arise if there is insufficient log space during recovery?

How does Aries prevent this from happening?

**d.** How does Aries handle actions that cannot be undone, such as deleting an

operating system file?

**e.** Suppose you want a transaction consistent archival dump of your database.

Suggest a way to do this with minimal interruption of the system. You may create a point in time where no transactions are active, by aborting all active transactions, but must be able to restart transactions immediately after all active transactions have been aborted.

**23.4** The Oracle database system uses undo log records to provide a snapshot view of the database to read-only transactions. The snapshot view reflects updates of all transactions that had committed when the read-only transaction started; updates of all other transactions are not visible to the read-only transactions.

Give a scheme for buffer handling whereby read-only transactions are given a snapshot view of pages in the buffer. Include details of how to use the log to generate the snapshot view, assuming that the advanced recovery algorithm is used. Assume for simplicity that a logical operation and its undo affect only a single page.

**23.5** Consider “dirty” pages in the buffer (i.e., each page that has some updates that are not yet reflected on disk) Suppose you maintain a table that gives the log sequence number of every dirty page in the buffer and periodically write this table out to the log.

**a.** Describe how to use this table to minimize work during the redo recovery phase of the advanced recovery scheme.

**b.** Convert the above scheme to also write out the list of active transactions with the table. Describe recovery with this addition.

**23.6 a.** Normally an action such as an insertion/deletion/update of a relation also updates all indices on the relation. Suppose, index maintenance is deferred, that is, it is performed only at the end of the transaction. What problems can this cause for a transaction that involves multiple steps.

**b.** Consider an update statement that scans a relation and deletes tuples with a salary field of less than 50, and reinserts the tuple with the salary value increased by 10. What problems could arise if the update to the relation is done while the scan is in progress. Describe a technique to avoid the problem.

**23.7** If you wanted high availability (i.e., the system should be up for as much time as possible), would you use 2-very safe or 2-safe replication? Why?

**23.8** Consider a hot-spare system, where there is a primary computer and a backup computer. Log records generated at the primary are sent over the network to the backup computer, where they can be used to keep a copy of the database in sync with the primary. If the primary computer fails, the backup starts process- ing transactions. There are three logging techniques in such an environment: one safe where a transaction commits as soon as the commit record hits the disk at the primary, two-very-safe where the transaction commits only if the com- mit log record hits the disk at the primary and the backup, and two-safe which is the same as two-very-safe if both sites are up, but if the backup is down a transaction commits as soon as the commit record hits the disk at the primary.

Describe some benefits and drawbacks of each of the three schemes.

**23.9** Suppose you wanted to maintain a remote backup copy of a database.

**a.** Briefly outline how you could use the system log generated by the ad- vanced recovery algorithm to keep it synchronized with the primary copy.

**b.** Describe what happens on transaction abort.

**c.** Describe what should be done when the remote backup fails, and when it recovers.

**d.** Describe how the remote backup can take over processing if the primary site fails (assume that all the log records generated at the primary have reached the remote backup).

**Chapter 24 – Database Security**

**24.1.** Discuss what is meant by each of the following terms: *database authorization, access control, data encryption, privileged (system) account, database audit, audit trail.*

**24.2.** Which account is designated as the owner of a relation? What privileges does the owner of a relation have?

**24.3.** How is the view mechanism used as an authorization mechanism?

**24.4.** Discuss the types of privileges at the account level and those at the relation level.

**24.5.** What is meant by granting a privilege? What is meant by revoking a privilege?

**24.6.** Discuss the system of propagation of privileges and the restraints imposed by horizontal and vertical propagation limits.

**24.7.** List the types of privileges available in SQL.

**24.8.** What is the difference between *discretionary* and *mandatory* access control?

**24.9.** What are the typical security classifications? Discuss the simple security property and the \*-property, and explain the justification behind these rules for enforcing multilevel security.

**24.10.** Describe the multilevel relational data model. Define the following terms:

*apparent key, polyinstantiation, filtering.*

**24.11.** What are the relative merits of using DAC or MAC?

**24.12.** What is role-based access control? In what ways is it superior to DAC and MAC?

**24.13.** What are the two types of mutual exclusion in role-based access control?

**24.14.** What is meant by row-level access control?

**24.15.** What is label security? How does an administrator enforce it?

**24.16.** What are the different types of SQL injection attacks?

**24.17.** What risks are associated with SQL injection attacks?

**24.18.** What preventive measures are possible against SQL injection attacks?

**24.19.** What is a statistical database? Discuss the problem of statistical database security.

**24.20.** How is privacy related to statistical database security? What measures can be taken to ensure some degree of privacy in statistical databases?

**24.21.** What is flow control as a security measure? What types of flow control exist?

**24.22.** What are covert channels? Give an example of a covert channel.

**24.23.** What is the goal of encryption? What process is involved in encrypting data and then recovering it at the other end?

**24.24.** Give an example of an encryption algorithm and explain how it works.

**24.25.** Repeat the previous question for the popular RSA algorithm.

**24.26.** What is a symmetric key algorithm for key-based security?

**24.27.** What is the public key infrastructure scheme? How does it provide security?

**24.28.** What are digital signatures? How do they work?

**24.29.** What type of information does a digital certificate include?

**Chapter 25 – Distributed Database**

**25.1** What are transaction scale up and batch scale up? For each, give an example of a database application class where it is a useful performance metric.

**25.2** If you were building a highly parallel database system, which of the following architectures would you use, and why: shared nothing, shared disk or shared memory.

**25.3** Two-phase commit

**a.** What is the motivation for two-phase commit and what does it ensure?

**b.** Outline the main phases of two-phase commit during normal operation.

**c.** If a coordinator fails after a site has voted to commit the transaction, what can the site do?

**d.** If a site fails before informing the coordinator that it is ready, what should the coordinator do?

**e.** When a failed site recovers, what should it do?

**25.4** In 2PC, suppose the coordinator fails after sending a <prepare T > message.

Describe the protocol used by participating sites (before the coordinator recovers) to handle the transaction.

**25.5** Suppose all data is stored partitioned in round robin fashion. The partition op- erator can be specified in the form part[pa](r) where r is a relation and pa is a list of the partitioning attributes; don’t worry about the exact partitioning vector. Give a good parallel plan using the partition operator and relational algebra operators (operating on local data) for the query

SELECT r.C, count(\*) FROM r, s WHERE r.A = S.B AND r.C = s.D GROUP BY r.C

**25.6** 2PC

**a.** In two-phase commit, outline what steps a site has to take when it recovers and finds a <**ready** T, L> record in its log; assume that the coordinator of T is known, and can be contacted, but the status of the transaction has not been finalized yet.

**b.** Now, suppose that the coordinator cannot be contacted. What can a site do to find the status of a transaction T , assuming it knows the set of sites participating in transaction T .

**25.7** Majority protocol

**a.** Explain how reads and writes are executed with replicas handled using the majority protocol.

**b.** The majority protocol requires that multiple replicas be updated atomically. How can this be done?

**c.** In the event of a network partition, does the majority protocol ensure consistency or availability? Briefly explain your answer.

**Chapter 26 – Enhanced Data Models for Advanced Applications**

**26.1.** What are the differences between row-level and statement-level active rules?

**26.2.** What are the differences among immediate, deferred, and detached *consideration* of active rule conditions?

**26.3.** What are the differences among immediate, deferred, and detached *execution* of active rule actions?

**26.4.** Briefly discuss the consistency and termination problems when designing a set of active rules.

**26.5.** Discuss some applications of active databases.

**26.6.** Discuss how time is represented in temporal databases and compare the different time dimensions.

**26.7.** What are the differences between valid time, transaction time, and bitemporal relations?

**26.8.** Describe how the insert, delete, and update commands should be implemented on a valid time relation.

**26.9.** Describe how the insert, delete, and update commands should be implemented on a bitemporal relation.

**26.10.** Describe how the insert, delete, and update commands should be implemented on a transaction time relation.

**26.11.** What are the main differences between tuple versioning and attribute versioning?

**26.12.** How do spatial databases differ from regular databases?

**26.13.** What are the different types of spatial data?

**26.14.** Name the main types of spatial operators and different classes of spatial queries.

**26.15.** What are the properties of R-trees that act as an index for spatial data?

**26.16.** Describe how a spatial join index between spatial objects can be constructed.

**26.17.** What are the different types of spatial data mining?

**26.18.** State the general form of a spatial association rule. Give an example of a spatial association rule.

**26.19.** What are the different types of multimedia sources?

**26.20.** How are multimedia sources indexed for content-based retrieval?

**26.21.** What important features of images are used to compare them?

**26.22.** What are the different approaches to recognizing objects in images?

**26.23.** How is semantic tagging of images used?

**26.24.** What are the difficulties in analyzing audio sources?

**26.25.** What are deductive databases?

**26.26.** Write sample rules in Prolog to define that courses with course number above CS5000 are graduate courses and that DBgrads are those graduate students who enroll in CS6400 and CS8803.

**26.27.** Define clausal form of formulas and Horn clauses.

**26.28.** What is theorem proving and what is proof-theoretic interpretation of rules?

**26.29.** What is model-theoretic interpretation and how does it differ from proof theoretic interpretation?

**26.30.** What are fact-defined predicates and rule-defined predicates?

**26.31.** What is a safe rule?

**26.32.** Give examples of rules that can define relational operations SELECT,

PROJECT, JOIN, and SET operations.

**26.33.** Discuss the inference mechanism based on relational operations that can be applied to evaluate nonrecursive Datalog queries.

**Chapter 28 – Data Mining Concepts**

**28.1.** What are the different phases of the knowledge discovery from databases?

Describe a complete application scenario in which new knowledge may be mined from an existing database of transactions.

**28.2.** What are the goals or tasks that data mining attempts to facilitate?

**28.3.** What are the five types of knowledge produced from data mining?

**28.4.** What are association rules as a type of knowledge? Give a definition of support and confidence and use them to define an association rule.

**28.5.** What is the downward closure property? How does it aid in developing an efficient algorithm for finding association rules, that is, with regard to finding large itemsets?

**28.6.** What was the motivating factor for the development of the FP-tree algorithm for association rule mining?

**28.7.** Describe an association rule among hierarchies with an example.

**28.8.** What is a negative association rule in the context of the hierarchy in Figure 28.3?

**28.9.** What are the difficulties of mining association rules from large databases?

**28.10.** What are classification rules and how are decision trees related to them?

**28.11.** What is entropy and how is it used in building decision trees?

**28.12.** How does clustering differ from classification?

**28.13.** Describe neural networks and genetic algorithms as techniques for data mining. What are the main difficulties in using these techniques?

**Chapter 29 – Overview of Data Warehousing and OLAP**

**29.1.** What is a data warehouse? How does it differ from a database?

**29.2.** Define the terms: OLAP (online analytical processing), ROLAP (relational

OLAP), MOLAP (multidimensional OLAP), and DSS (decision-support systems).

**29.3.** Describe the characteristics of a data warehouse. Divide them into functionality

of a warehouse and advantages users derive from it.

**29.4.** What is the multidimensional data model? How is it used in data warehousing?

**29.5.** Define the following terms: star schema, snowflake schema, fact constellation, data marts.

**29.6.** What types of indexes are built for a warehouse? Illustrate the uses for each with an example.

**29.7.** Describe the steps of building a warehouse.

**29.8.** What considerations play a major role in the design of a warehouse?

**29.9.** Describe the functions a user can perform on a data warehouse and illustrate the results of these functions on a sample multidimensional data warehouse.

**29.10.** How is the concept of a relational view related to a data warehouse and data marts? In what way are they different?

**29.11.** List the difficulties in implementing a data warehouse.

**29.12.** List the open issues and research problems in data warehousing.

Questions for Chapters 17, 18, 19, 21, 22, 23 and 25 are from **Database System Concepts** by Silberschatz, et. al., 6th ed. 2013. The remaining chapter’s questions are from the review questions at the end of the chapters from the textbook.