 欢迎大家添加答案，如果能把你的答案，想法或者疑问写在每道题的下面，那就最好不过了。
此文档会在7月2日晚11：59时关闭，有需要的同学请自行备份。

2019 Model

Q1. a) What computer hardware aspects influence the performance of a database system?

Database architecture
Memory chip (Moore's law)
Processor performance (Joy's law)
Memory access time （Hit ratio）
Speed of seek and rotation (Disk access time)
RAID
Bus

b) An electronic fund transfer application has a function that can transfer an amount of X from account A to account B. The updates in the function are written as a single transaction. However, the function was modified by a newly employed programmer with the view of improving the performance of the function by dividing the updates into two sequential transactions. What can go wrong with such a decision?

This will cause an atomicity problem. The transaction may not fail if any of the two sub transactions fails, it doesn't keep the transaction's atomicity.

Q2. a) What are sufficient conditions to achieve the ACID properties? Do not give details of ACID properties to answer the question.

A sufficient condition for isolation is that concurrent executions are equivalent to sequential ones, so that all transactions appear as if they were executed one after the other rather than in an interleaved manner; this condition is made precise through serializability.

2pl + well-form
b) What are the problems in the following history generated by the execution of transactions T1 and T2? Give its dependency relation.
Note: the following is not complete history and you will be given complete history if such a question is asked.
<(T1,Slock A), (T1,Xlock B), (T2, Slock A), (T2, Read A),
(T2, Xlock B), (T2, Write B), ….,(T2, Unlock B), (T1, Read A), (T1, Write B), (T1,unlock A), (T1, Unlock B)>

Provide a correct history for the above to rectify the problems.

The history is illegal. It grants conflict Xlocks on B. After it grants Xlock on B to T1 (H[1]), it should not grant Xlock on B to T2 (H[4]).

T1 XlockB 之后T2 不能加Xlock B了，要等T1 unlock B后才能加锁，所以这属于什么问题？因为两个XL incompatible。 The history is ***illegal (Set 5 P18)***
(T2, Write B)这个操作无效了吧。有没有人解释一下DEP(H)。。。
DEP(H)={<T2,B,T1>} I think that we only care about Write-W, W-Read, R-W dependencies.
DEP(H) = {<T1,A,T2>,<T1,B,T2>,<T2,A,T1>,<T2,B,T1>}
T1 and T2 depend on each other, there may be a deadlock happening during the transactions since the T1 holds the Xlock of resource B, T2 cannot get the resource B until T1 commits.
<(T1,Slock A),(T2,Slock A),(T2,Read A),(T2,Xlock B),(T2, Write B)(T2, Unlock B),(T1, Read A), (T1, Xlock B),(T1, Write B),(T1, Unlock A), (T1, Unlock B)>
Q3. a) Describe The need for a nested transaction model. What commit rules are followed in the Nested Transaction model and why?

A subtransaction can either commit or abort, however, commit cannot take place unless the parent itself commits.
Subtransactions have Atomicity, Consistency, and Isolation properties but not have Durability property unless all its ancestors commit.
Commit of a sub transaction makes its results available only to its parents, not available to its siblings.
These three roles keep the ACID properties of the main transaction.
b) Describe how one can modify the content of a disk page (block) atomically. How is this done in the case of writing log pages?
Either entire block is written correctly on disk or the contents of the block is unchanged. It requires duplex write:
• a block of data is written two disk blocks sequentially
• determine whether the contents of a disk block has an error or not by checking its CRC.
• each block is associated with a version number
• the block with the latest version number contains the most recent data
• if one of the writes fail, system can issue another write to the disk block that failed
• it always guarantees at least one block has consistent data
In case of writing log pages, multiple logs fit in a single log page so it can be written efficiently. And we never overwrite of remove logs, so there is no problems of losing existing (flushed) logs.

Logged write - it is similar to duplex write except one of the writes go to a log. This method is very efficient if the changes to the block are small.

Q4. Describe locking protocols used for various degrees of isolation. What are the benefits of various degrees of isolation?
Degree 3: A three degree isolated transaction has no lost updates, and has repeatable reads. Lock protocol is two phase and well formed. It has medium overheads and is sensitive to all kinds of dependencies: write -> write, write -> read, read -> write. It is able to recover or rollback.
Degree 2: A two degree isolated transaction has no lost updates and no dirty reads. Lock protocols are two phased with respect to exclusive locks and well formed with respect to

Read and Write (Non repeatable reads). It has medium concurrency and medium overhead.
It can roll back and recover.
Degree 1: A one degree isolation has no last updates. Low protocol is two phase with respect to exclusive locks and well formed with respect to writes.
It has high concurrency and small overhead. It is able to rollback and recover.
Degree 0: A zero degree transaction does not overwrite another transaction's dirty data if the other transaction is at least One degree. Lock protocol is well-formed with respect to writes.
It has the maximum concurrency and least overhead, but it is not able to rollback or recover.
Benefits: different degrees of isolations have different overhead in terms of performance, provide different concurrency capabilities and determine if a transaction can rollback. So we can choose different degrees of isolation as needed.

Q5. a) Why (CLRs) in are written and when?
Before restoring the old value of a page, write a CLR. While we try to do crash recovery, we need to repeat the history to reconstruct by reapplying all updates and CLRs.
When: when transaction abort, before restoring the old value of a page (write CLRs); and in the UNDO phase during cash recovery, *before(?)* undo updates (write CLRs).

b) why log does not have to be flushed on Abort Transaction but on Commit Transaction?
Because after a commit transaction, the data that is committed needs to write to disk only if flushedLSN is bigger or equal to the lastLSN in the transaction table.
Aborting a transaction rolls back the current transaction and causes all the updates made by the transaction to be discarded. So it’s fine we lost logs of the current transaction.
While on Commit, we need to flush logs to ensure durability.

Q6. The following is a log after a crash. Describe the steps that are performed to recover the system after a crash if ARIES is used for recovery. Your answer should clearly show the construction of Transaction Table and Dirty Page table during the recovery. The format of a log record is (LSN, TYPE, TID, Previous LSN)
Log:
(00, begin checkpoint,-, -)
(05, end checkpoint,-, -)
(10, writes to page 5, T1, -)
.
. This missing part will be provided in the exam.
.
(100, End, T4, 90)
(110, write page 2, T3, 50)
(120, write page 8, T2, 60)

Q7. a) Compute probability of a deadlock? What are the assumptions made with such simplistic model
number transactions = n + 1 n when n is large
each transaction access r locks exclusively { that is it takes an exclusive lock sequentially
total number of records in the database = R
on average each transaction is holder r/2 locks approximately (minimum zero and

maximum r)
transaction just commenced holds none and transaction about to nish holds r of them
average number of locks taken by the other n transactions = n * r/2 = nr/2
b) A company uses k-way disk mirroring for its storage system and has decided to reconfigure the system to m-way mirroring to increase its storage capacity. You will be given necessary parameters if such a question is asked. What is the ratio of change in MTTF of the new storage system over the old system? You do not need a calculator to solve this problem.
Calculated by failvote.
$M = (\frac{1}{m} + \frac{1}{m-1} + \frac{1}{m-2} + ... + \frac{1}{n}),\ n = m/2 + 1$
$K = (\frac{1}{k} + \frac{1}{k-1} + \frac{1}{k-2} + ... + \frac{1}{j}),\ j = k/2 + 1$
The final ratio change is equal (M/K).

+1 ==>

2018

Q1. A computer system running database systems has 32 GB of main memory used for its buffer cache and an SSD of 128GB of storage which is also used as a buffer cache for the hard drive consisting of 4TB storage. The main buffer cache and solid-state drive provide cache hit rates of 60% and 90% respectively. The memory access time is 100 times faster than the SSD access, and each SSD access takes 0.01 milliseconds. The hard drive average access taking into both seek and rotational latency is 50 times slower than SSD. what would be the average access time to access data from the hard drive with and without SSD as a buffer.

SSD as a buffer cache.
SSD access time = 0.01 milliseconds
Access(memory) = 0.0001 ms
Access(hard drive) = 0.5 ms
1) Without SSD:
   60%*Access(memory)+(1-60%)*Access(hard drive)
2) With SSD
   60%*Access(memory)+(1-60%)*((1-90%)*Access(hard drive)+90%*Access(SSD))
   Not sure about the solution

Q2. An electronic fund transfer application had a function Transfer() that can transfer an amount of X from account A and Y from account B to account C. The account updates in the function are written as a single transaction.

However, a newly employed programmer wanted to reduce the amount of locking duration and modified the function Transfer() into three separate sequential transactions. The first transaction updates account A, the second updates account B and third update updates account C. What are the implications of such a modification?

In the new function, if any of the transaction A or transaction B aborts (for example, the account A or B is under 0), the main function will not be ended, so it cannot keep the whole transaction's atomicity.

On the other hand, it will affect the consistency of the transactions. For example, there are two transactions T1 and T2, we assume when the T1 is on its third transaction, T2 is on its first transaction which has the account A that has been modified by T1. If T1 aborts in the third transaction, and the whole transaction should be rolled back. Then T2 needs rollback as well.

Q3. Describe how granular locking mode such as IX (intention to take exclusive lock) can decrease the number of locks needed and also decreases the duration of lock contention in the database system. You can use a simple example to answer the questions.

Whenever we put an intent lock on an object on a certain level, all the descendants of the object will be automatically granted the locks, which means we do not need to put locks on those objects, in this way, we can decrease the number of locks needed.

As for how can it decreases the duration of lock contention: (I'm not sure)

Ref:

https://www.sqlpassion.at/archive/2016/05/16/why-do-we-need-intent-locks-in-sql-server/

"Imagine you want to acquire an Exclusive Lock at the table level. In that case, SQL Server has to know if there is an incompatible lock (like a Shared or Update Lock) somewhere else on a record. Without Intent Locks SQL Server would have to check every record to see if an incompatible lock has been granted.

But with an Intent Shared Lock on the table level, SQL Server knows immediately that a Shared Lock has been granted somewhere else, and therefore an Exclusive Lock can't be granted at the table level. That's the whole reason why Intent Locks exist in SQL Server: to allow efficient checking if an incompatible lock exists somewhere within the Lock Hierarchy. Quite easy, isn't it?"

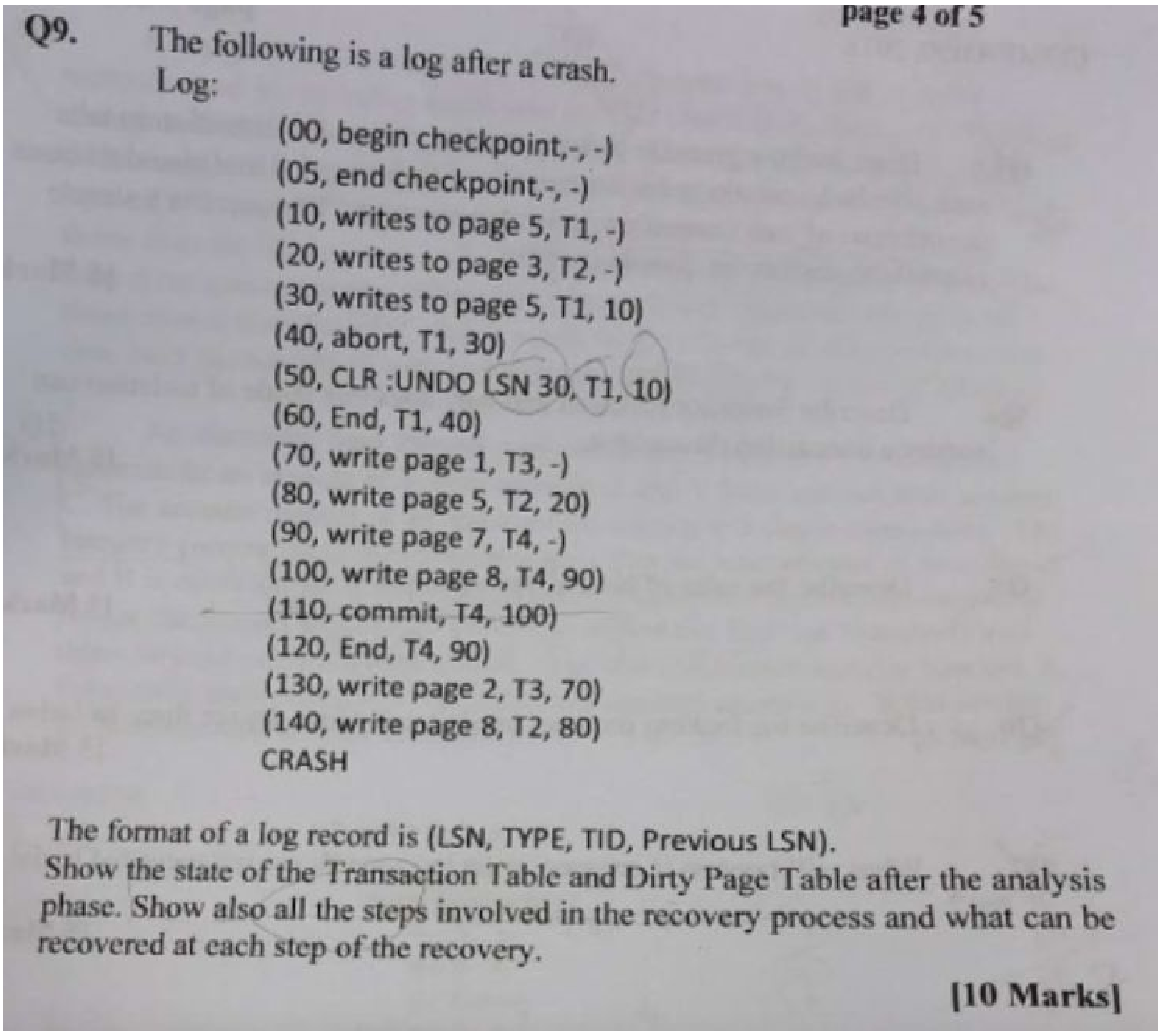
Q4. Describe Snapshot Isolation and why does this mode of isolation can increase transaction throughput.
Check the data which are to be updated before write.

Q5. Describe the rules of Nested Transactions.
Commit rule
A subtransaction can either commit or abort, however, commit cannot take place unless the parent itself commits.
Subtransactions have Atomicity, Consistency, and Isolation properties but not have Durability property unless all its ancestors commit.
Commit of a sub transaction makes its results available only to its parents, not available to its siblings.
Roll back rules
If a subtransaction rolls back all its children are forced to roll back
Visibility rules
Changes made by a sub transaction are visible to the parent only when the sub transaction commits. Whereas all objects of parents are visible to its children. Implication of this is that the parent should not modify objects while children are accessing them. This is not a problem as parent is not run in parallel with its children.

Q6. Describe the locking protocol needed to achieve degree three isolation.
2 phase locking on both Slocks and Xlocks
Well formed

Q7. What happens if compensation log records are not recorded in the log?
If the database system crashed, the log system cannot undo the effects of the failed transactions.

Q8. What are the rules of WAL (Write Ahead Logging)? Describe the purpose of each rule.
Must force the log record for an update before the corresponding data page gets to disk (stolen)
    guarantees Atomicity
Must write all log records to disk (force) for a transaction before commit
    guarantees Durability
Q9.



Q9. The following is a log after a crash.
Log:

```
(00, begin checkpoint,-, -)
(05, end checkpoint,-, -)
(10, writes to page 5, T1, -)
(20, writes to page 3, T2, -)
(30, writes to page 5, T1, 10)
(40, abort, T1, 30)
(50, CLR :UNDO LSN 30, T1, 10)
(60, End, T1, 40)
(70, write page 1, T3, -)
(80, write page 5, T2, 20)
(90, write page 7, T4, -)
(100, write page 8, T4, 90)
(110, commit, T4, 100)
(120, End, T4, 90)
(130, write page 2, T3, 70)
(140, write page 8, T2, 80)
CRASH
```

The format of a log record is (LSN, TYPE, TID, Previous LSN).
Show the state of the Transaction Table and Dirty Page Table after the analysis phase. Show also all the steps involved in the recovery process and what can be recovered at each step of the recovery.

[10 Marks]

Transaction Table

Xid	Status	LastLSN
T2	Running	140
T3	Running	130

Dirty read page

Page number	OldestLSN
5	10
3	20
1	70
7	90
2	130

8	100

Redo all the activities
10->30->50
20->80->140
70->130
90->100->110->120

我觉得答案应该是：10->20->30->50->70->80->90->100->130->140 +2

再按照从小到大排个序
Undo effects of failed transaction
140->80->20
130->70

不需要按照T来写吧？我觉得undone order是 +1
140 130 80 70 20

Q10. The database system has the following properties: the number of records in the system is 4 billion (2^32); the average number of concurrently executing transactions is 4096(2^12); the average number of records accessed per transaction is 64(2^6). What is the probability of a deadlock in such a database system? Suggest two ways how one can decrease this probability.
Total number records R = 2^32
Number transactions n = 2^12
Each transaction access r = 2^6

$n \times nr^4/4R^2 = \frac{2^{24}\times 2^{24}}{4\times 2^{64}} = \frac{1}{2^{18}}$

Decrease the number of concurrently executing transactions or increase（？decrease） the average number of records accessed per transaction.

Q11. What are the main assumptions made in improving the reliability of a system? Describe how an n-plex system can improve very high reliability.
We assume that failures of components are independent and that P(AB) approximately equals to 0, so P(A+B) = P(A)+P(B).
P(X) is the probability of an event X happening in a certain period. Event A and Event B are independent. Therefore, P(A and B) = 0. P(A or B) = P(A)+P(B)
The faulty equipment is repaired with an average time of MTTR(mean time to repair) as soon as a fault is detected.