

Siddharth Bhat (20161105)

1 Q1

2 Q2

3 Q3

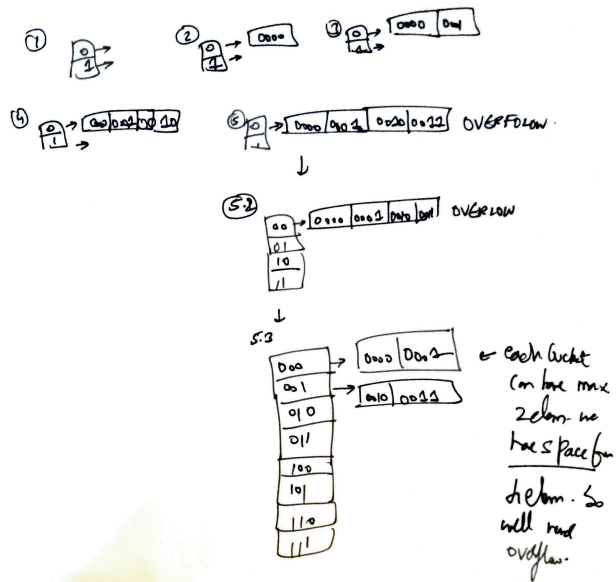
4 Q4

5 Q5

6 Q6

When we insert 0011, there are four records for bucket 0, which overflows. Adding a second bit to the bucket addresses doesn't help, because the first four records all begin with 00. Thus, we go to  $i = 3$  and use the first three bits for each bucket address. Now we have space for 4 records in each bucket; however, we can get at maximum 2 records per bucket — we have 4 bit hashes, our buckets are keyed by 3 bits, leaving 1 bit of ambiguity. 1 bit can be 2 records. When 1111 is inserted, we have two records in each bucket.

We assume we have access to MSG.



7 Q7

**7.0.1 Q7.1**

$SL_x(E)$  for a shared lock by transaction  $x$  on elem  $E$ .  $U_x(E)$  for unlock by transaction  $x$  on elem  $E$ .

$T_1$	$T_2$	$T_3$
$SL_1(A)$		
$R_1(A)$		
	$SL_2(B)$	
	$R_2(B)$	
		$SL_3(C)$
		$R_3(C)$
$SL_1(B)$		
$R_1(B)$		
	$SL_2(C)$	
	$R_2(C)$	
		$SL_3(A)$
		$R_3(A)$
$XL_1(A)$		
$W_1(A)$		
	$XL_2(B)$	
	$W_2(B)$	
		$XL_3(C)$
		$W_3(C)$

All locks are accepted, since there are no conflicting locks.

### 7.0.2 Q7.2

Table will be the same as for the above, since we did not have any read action which was followed by a write action of the same element by the same transaction.

### 7.0.3 Q7.3

## 8 Q8

$r_1(O_1) \mapsto T_1$  puts  $IS(B_1); S(O_1); \text{release}$   
 $r_2(O_2) \mapsto T_2$  puts  $IS(B_1); S(O_2); \text{release}$   
 $r_3(O_1) \mapsto T_3$  puts  $IS(B_2); S(O_3); \text{release}$   
 $w_1(O_3) \mapsto T_1$  puts  $IX(B_2); X(O_3); \text{release}$   
 $w_2(O_4) \mapsto T_2$  puts  $IX(B_2); X(O_4); \text{release}$   
 $w_3(O_5) \mapsto T_3$  puts  $IX(B_2); X(O_5); \text{release}$   
 $w_1(O_2) \mapsto T_1$  puts  $IX(B_2); X(O_3); \text{release}$

## 9 Q9

### 9.0.1 9.1

In Undo Logging Logging, we need to write all we need to write all modified data to disk before committing a transaction. This may need a large number of disk I/Os. This is unlike the case of Redo logging, which allows changes to be present in-memory; only need to flush changes before committing.

### 9.0.2 9.2

Selinger optimization improves upon DP approach by keeping for each not only the plan of least cost, but also plans that have higher cost but produce a result that is sorted in an order that may be useful for parent queries.

### 9.0.3 9.3

View serializable: If a given schedule is found to be view equivalent to some serial schedule. Alternatively, there are no cycles in the dependency graph.  
Conflict serializable: If there are no cycles in the conflict graph.

### 9.0.4 9.4

We can use strict 2-phase locking for recoverability. This requires that in addition to the lock being 2-Phase, all Exclusive(X) Locks held by the transaction be released until after the Transaction Commits.

### 9.0.5 9.5

Database operations are in fact relational algebra operations. These operations are pure mathematical expressions, and are generally reads or writes into disjoint pieces of data. This makes them naturally parallelizable.

#### 9.0.6 9.6

File system does not generally have multiple readers and writers to a single file. It also does not need to manage structured data. Hence, many of the ACID like concerns simply do not occur in the case of a file system.

#### 9.0.7 9.7

the commit bit for X is true if and only if the most recent transaction to write X has already committed. The purpose of this bit is to avoid a situation where one transaction T reads data written by another transaction U, and U then aborts. This problem, where T makes a "dirty read" of uncommitted data, certainly can cause the database state to become inconsistent, and any scheduler needs a mechanism to prevent dirty reads.

#### 9.0.8 9.8

- Two-phase locking - 2PL.
- General lock based solutions.
- Timestamp ordering.
- Validation based concurrency control.

Increment based locking is good in this case because it allows to add or subtract a constant from an element, which is what most kinds of bank transactions are. Increment locks on the same element do not conflict with each other.

#### 9.0.9 9.9

recovery manager will have to DODO

#### 9.0.10 9.10

all trees of n vertices is  $n^{n-2}$ . Number of left-deep trees is  $n!$ .  $n^{n-2}$  is much larger than  $n!$ .