Προχωρημένα Θέματα Τεχνολογίας & Εφαρμογών Βάσεων Δεδομένων
Άσκηση 1

Όθωνας Γκαβαρδίνας ΑΜ: 2620

ΘΕΜΑ 0
Το buffer-pull, είναι μια περιοχή στη μνήμη που χρησιμοποιείται για την γρήγορη προσπέλαση συχνά χρησιμοποιούμενων δεδομένων. Χρησιμοποιεί έναν αλγόριθμο τύπου LRU πάνω σε συνδεδεμένη λίστα από pages, που χωρίζεται σε δύο μέρη. Στο ένα μέρος του βρίσκονται τα λιγότερο χρησιμοποιούμενα δεδομένα και στο άλλο, δεδομένα που έχουν λάβει μέρος σε κάποιο query. Τα δεδομένα που φτάνουν στο old μέρος, σιγά σιγά αποσύρονται.

(α) Με την χρήση SELECT χωρίς WHERE γίνεται query για δεδομένα, πολλά από τα οποία δεν πρόκειται να χρησιμοποιηθούν, και έτσι αυτά μαζεύονται στη συνδεδεμένη λίστα του buffer pull, ενώ χρήσιμα δεδομένα που βρισκόταν εκεί για συχνή χρήση γίνονται evict. Λύση στο πρόβλημα αυτό αποτελεί, μια λεπτομέρεια της υλοποίησης LRU η οποία τοποθετεί τα δεδομένα που έρχονται για πρώτη φορά σε ένα old μέρος της λίστας, και αν επαναχρησιμοποιηθούν σε ένα new.

(β)
i) Με την εντολή **free** σε σύστημα ubuntu, εντόπισα ότι έχω **6103752 kB** μέγεθος RAM.
Μετατροπή σε bytes: **6250242048 bytes**.
Πολλαπλασίασα τον παραπάνω αριθμό x0.75 = **4687681536 bytes**. (75% της RAM)

ii) Συνδέθηκα στη MySQL ως root, και πληκτρολόγησα την εντολή:
show variables like 'innodb_buffer_pool_size';
Έλαβα ως αποτέλεσμα **134217728 bytes**.

iii) Πληκτρολόγησα την εντολή:
set global innodb_buffer_pool_size=4687681536;
OK

iv) Η εντολή **show variables like 'innodb_buffer_pool_size';**
αυτή τη φορά έφερε ως αποτέλεσμα **4697620480 bytes**, που είναι περίπου το 75% της RAM.

v) Κάθε φορά που κλείνω τη βάση με την εντολή **/etc/init.d/mysql stop**
χρειάζεται ξανά να πληκτρολογήσω την εντολή:
set global innodb_buffer_pool_size=4687681536;
,την επόμενη φορά που θα την ξανανοίξω.

(γ)
i) Με την παρακάτω εντολή έλαβα ως αποτέλεσμα για την βάση μου που ονομάζεται my_schema, ότι έχει μέγεθος **65536 bytes**.

```
SELECT table_schema AS "Database", SUM(data_length + index_length)
AS "Size"
FROM information_schema.TABLES
GROUP BY table_schema;
```

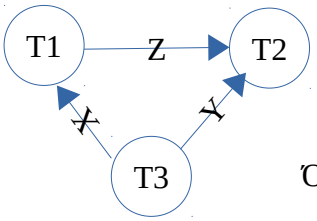
ii) Στη συνέχεια υπολόγισα το 10% της βάσης:
65536 * 0.1 = 6553.6 (~ **6554 bytes**) (0.07 MB)

iii) 65536 + 6554 = **72090 bytes**

set global innodb_buffer_pool_size=72090;

Τελικά η παρέμεινε ως είχε, στην τιμή **134217728 bytes**. (128 MB)
Στον υπολογιστή μου ισχύει η περίπτωση (γ).

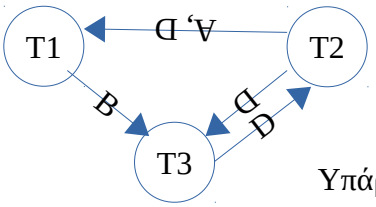
ΘΕΜΑ 1
(α)
S1: R1(X) R2(Z) R1(Z) R3(X) R3(Y) W1(X) W3(Y) C3 R2(Y) W2(Z) W2(Y) C1 C2



Όχι κύκλος, άρα σειριοποιήσιμο.

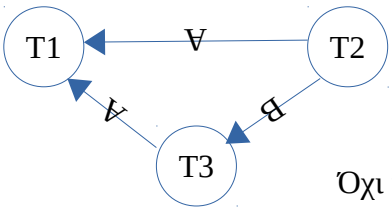
Σειριακό:
T3 / T1 / T2 (Ψάχνω κάθε φορά κάποιο κόμβο που να έχει indegree 0 και τον εξάγω.)
(Όπου Ti, όλες οι διαδικασίες της συναλλαγής i μαζί.)

S2: W2(A) R1(B) R2(D) W3(C) W1(B) W3(D) R3(B) R2(D) C2 C3 W1(D) R1(A) C1



Υπάρχει κύκλος T2 με T3, άρα όχι σειριοποιήσιμο.

S3: R3(A) W2(B) R1(V) W3(B) R3(A) C3 R2(A) R1(Y) R2(V) W1(A) C1 C2



Όχι κύκλος, άρα σειριοποιήσιμο.

Σειριακό:
T2 / T3 / T1

(β)
S1: R1(X) R2(Z) R1(Z) R3(X) R3(Y) W1(X) W3(Y) C3 R2(Y) W2(Z) W2(Y) C1 C2

T1: R1(X) R1(Z) W1(X)
T2: R2(Z) R2(Y) W2(Z) W2(Y)
T3: R3(X) R3(Y) W3(Y)

S1 (1)	X	Y	Z
T1	S		S
T2			S
T3	S	X	

-Εφ' όσον γνωρίζω όλη την S1 εξ' αρχής, μπορώ για την R1(X) να κλειδώσω με X-lock αφού αργότερα έχω W1(X).

output: X1(X) R1(X)

-Ομοίως για το R2(Z) με το W2(Z).

output: X1(X) R1(X) X2(Z) R2(Z)

-Στο επόμενο βήμα, δεν μπορώ να πω R1(Z) επειδή, υπάρχει αργότερα κάποιο W2(Z).
Επομένως θεωρώ πως το lock του προηγούμενου βήματος ήταν απλά S-lock.

output: X1(X) R1(X) S2(Z) R2(Z)

και έτσι γίνεται να συνυπάρξουν τα S1(Z) και S2(Z):

output: X1(X) R1(X) S2(Z) R2(Z) S1(Z) R1(Z)

-Ομοίως για το επόμενο βήμα, αλλάζω την πρώτη κλειδαριά σε S1(X).

output: S1(X) R1(X) S2(Z) R2(Z) S1(Z) R1(Z)

και τώρα θα έχω:

output: S1(X) R1(X) S2(Z) R2(Z) S1(Z) R1(Z) S3(X) R3(X)

-Κλειδώνω το επόμενο με X-lock, επειδή αργότερα υπάρχει W3(Y).

output: S1(X) R1(X) S2(Z) R2(Z) S1(Z) R1(Z) S3(X) R3(X) X3(Y) R3(Y)

-Η συναλλαγή T3 μπορεί σ' αυτό το σημείο να αρχίσει να ξεκλειδώνει.
Αφήνει λοιπόν τον πόρο X, τον οποίο δεσμεύει η T1, αυτή τη φορά για W με X-lock.

output: S1(X) R1(X) S2(Z) R2(Z) S1(Z) R1(Z) S3(X) R3(X) X3(Y) R3(Y) U3(X) X1(X) W1(X).

Ο πίνακας θα πάρει την εξής μορφή:

S1 (2)	X	Y	Z
T1	X		S
T2			S
T3		X	

-Στη συνέχεια η T3, έχοντας το X-lock, δε χρειάζεται να κλειδώσει ξανά, κάνει και το τελευταίο unlock και commit:

output: S1(X) R1(X) S2(Z) R2(Z) S1(Z) R1(Z) S3(X) R3(X) X3(Y) R3(Y) U3(X) X1(X) W1(X) W3(Y) U3(Y) C3

S1 (3)	X	Y	Z
T1	X		S
T2		X	S
T3			

-Το R2(Y), κλειδώνει με X-lock, επειδή υπάρχει το W2(Y):

output: S1(X) R1(X) S2(Z) R2(Z) S1(Z) R1(Z) S3(X) R3(X) X3(Y) R3(Y) U3(X) X1(X) W1(X) W3(Y) U3(Y) C3 X2(Y) R2(Y)

-Η συναλλαγή T1 μπορεί να κάνει unlock τον πόρο Z, και να τον δεσμεύσει με X-lock η T2:

output: S1(X) R1(X) S2(Z) R2(Z) S1(Z) R1(Z) S3(X) R3(X) X3(Y) R3(Y) U3(X) X1(X) W1(X) W3(Y) U3(Y) C3 X2(Y) R2(Y) U1(Z) X2(Z) W2(Z)

S1 (4)	X	Y	Z
T1	X		
T2		X	X
T3			

-Τέλος, γίνεται και W2(Y), καθώς η T2 έχει ήδη ένα X-lock, και γίνονται και τα τελευταία unlocks:

output: S1(X) R1(X) S2(Z) R2(Z) S1(Z) R1(Z) S3(X) R3(X) X3(Y) R3(Y) U3(X) X1(X) W1(X) W3(Y) U3(Y) C3 X2(Y) R2(Y) U1(Z) X2(Z) W2(Z) W2(Y) U1(X) C1 U2(Y) U2(Z) C2

Επομένως, μπορεί να προκύψει 2PL.

S2: Όχι, δεν μπορεί να προκύψει 2PL, καθώς το γράφημα είναι άκυκλο.

S3: R3(A) W2(B) R1(V) W3(B) R3(A) C3 R2(A) R1(Y) R2(V) W1(A) C1 C2

T1: R1(V) R1(Y) W1(A)
T2: W2(B) R2(A) R2(V)
T3: R3(A) W3(B) R3(A)

S3 (1)	A	B	V	Y
T1			S	
T2	S	X	S	
T3	S			

-Για το R3(A), απλά κλειδώνω με S-lock:

output: S3(A) R3(A)

-Για το επόμενο, με X-lock:

output: S3(A) R3(A) X2(B) W2(B)

-Για το επόμενο, με S-lock:

output: S3(A) R3(A) X2(B) W2(B) S1(V) R1(V)

-Στη συνέχεια θέλω ένα X-lock για το W3(B).
Το ενδεχόμενο να κάνω U2(B) άμεσα, απορρίπτεται επειδή υπάρχουν τα R2(A) και R2(V) αργότερα.
Το ενδεχόμενο να κάνω πρώτα S2(A) S2(V) και μετά U2(B) είναι αποδεκτό.

output: S3(A) R3(A) X2(B) W2(B) S1(V) R1(V) S2(A) S2(V) U2(B)

-και η T3 κλειδώνει τον πόρο B:

output: S3(A) R3(A) X2(B) W2(B) S1(V) R1(V) S2(A) S2(V) U2(B) X3(B) W3(B)

S3 (2)	A	B	V	Y
T1			S	
T2	S		S	
T3	S	X		

- Η T3 έχει ήδη ένα S-lock επομένως δε χρειάζεται να κλειδώσει ξανά, και κάνω και τα unlock για την T3 για το commit C3:

output: S3(A) R3(A) X2(B) W2(B) S1(V) R1(V) S2(A) S2(V) U2(B) X3(B) W3(B) R3(A) U3(A) U3(B) C3

S3 (3)	A	B	V	Y
T1			S	S
T2	S		S	
T3				

-Δεν χρειάζεται lock για R2(A), και χρειάζεται για R1(Y), και δε χρειάζεται για R2(V):

output: S3(A) R3(A) X2(B) W2(B) S1(V) R1(V) S2(A) S2(V) U2(B) X3(B) W3(B) R3(A) U3(A) U3(B) C3 R2(A) R2(A) S1(Y) R1(Y) R2(V)

-Τέλος η T1, αφήνει τον πόρο A, γίνεται το W1(A) με κλείδωμα και γίνονται και τα τελευταία unlocks πριν τα commit:

S3 (3)	A	B	V	Y
T1	X		S	S
T2			S	
T3				

output: S3(A) R3(A) X2(B) W2(B) S1(V) R1(V) S2(A) S2(V) U2(B) X3(B) W3(B) R3(A) U3(A) U3(B) C3 R2(A) R2(A) S1(Y) R1(Y) R2(V) U2(A) X1(A) W1(A) U1(A) U1(V) U1(Y) C1 U2(V) C2

Επομένως, μπορεί να προκύψει 2PL.

(γ)

S1: Σε ενδιάμεσο βήμα χρειάστηκε να γίνει unlock από τον T3 ο πόρος X και αμέσως μετά δεν έγινε η διαδικασία ξεκλειδώματος όλων των πόρων και commit. Το βήμα:

output: S1(X) R1(X) S2(Z) R2(Z) S1(Z) R1(Z) S3(X) R3(X) X3(Y) R3(Y) U3(X) X1(X) W1(X).

Επομένως, δεν μπορεί να προκύψει αυστηρό 2PL.

S2: Αφού όχι 2PL, όχι και αυστηρό 2PL.

S3: Ομοίως με S1, στο εξής βήμα:

output: S3(A) R3(A) X2(B) W2(B) S1(V) R1(V) S2(A) S2(V) U2(B) X3(B) W3(B)

Επομένως, δεν μπορεί να προκύψει αυστηρό 2PL.

(δ)
Και τα τρία:
-Σίγουρα όχι REPEATABLE READ ή SERIALIZABLE, γιατί όχι αυστηρά 2PL.
-Σίγουρα όχι READ UNCOMMITED, αφού έχουν R και W.

S1: R1(X) R2(Z) R1(Z) R3(X) R3(Y) W1(X) W3(Y) C3 R2(Y) W2(Z) W2(Y) C1 C2

-Θα μπορούσε να έχει προκύψει από το επίπεδο READ COMMITED, καθώς όλα τα W γίνονται commit, πριν να επαναχρησιμοποιηθεί ο πόρος τους.

S2: W2(A) R1(B) R2(D) W3(C) W1(B) W3(D) R3(B) R2(D) C2 C3 W1(D) R1(A) C1

ΘΕΜΑ 2:
(i)

!!! !!!

R3(A) R1(B) R3(B) R2(A) R2(C) W3(A) W2(C) R1(C) W1(B) W1(C) R2(A) C2 W2(A) W4(C)
C1 C2 C3

Η συναλλαγή T3 τροποποιεί το δεδομένο A, το οποίο διαβάζει η συναλλαγή T2 και στη συνέχεια η T2 κάνει commit, πριν να κάνει commit η T3.

(ii)

!!! !!!

R3(A) R1(B) R3(B) R2(A) R2(C) W3(A) W2(C) R1(C) W1(B) W1(C) R2(A) W2(A) W4(C)
C3 C2 C1 C4
!!!

W3(A) με R2(A) -> επομένως πρώτα τελειώνει το C3
W2(C) με R1(C) -> στη συνέχεια το C2
και τέλος τα υπόλοιπα 2.
Έτσι έχουμε ανανήψιμο χρονοπρόγραμμα.

Επίσης, επειδή W3(A) με R2(A) και αργότερα γίνεται commit η T3, και έτσι δεν ήταν επικυρωμένη.

(iii)

!!!

R3(A) R1(B) R3(B) R2(A) R2(C) W3(A) C3 W2(C) R1(C) W1(B) W1(C) R2(A) W2(A) W4(C)

Δεν μπορώ να κάνω commit για την T2 λόγω του W2(C). Συγκεκριμένα η αμέσως επόμενη R1(C), για να διαβάσει το δεδομένο C, και να είναι το χρονοπρόγραμμα αποφεύγον τη διάδοση ανακλήσεων, χρειάζεται να έχει γραφθεί το δεδομένο, από επικυρωμένη συναλλαγή. Η επικύρωση δεν γίνεται, αφού η T2 έχει στη συνέχεια κι άλλες εντολές (R2(A) W2(A)).

(iv) Αφού το χρονοπρόγραμμα δεν είναι αποφεύγον τη διάδοση ανακλήσεων, δεν είναι και αυστηρό. Η παραπάνω αιτιολόγηση εξηγεί το λόγο.