ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 3
Μέρος Α: Phantoms και Ghosts

Ξεκινήστε τρεις (3) συνεδρίες στο περιβάλλον της MySQL και προχωρήστε εκτελώντας τις εντολές που ακολουθούν με τη δεδομένη σειρά και στην κατάλληλη συνεδρία/παράθυρο, ανάλογα με τον χρωματισμό τους.

Client A
Client B
Client C

1. Initialization

```
SET AUTOCOMMIT  = 0;
set innodb_lock_wait_timeout=1000;
```

```
SET AUTOCOMMIT  = 0;
```

```
SET AUTOCOMMIT  = 0;
```

```
DROP TABLE T;
CREATE TABLE T (id INT NOT NULL PRIMARY KEY, s VARCHAR(40),  i SMALLINT);
insert into T(id,s,i) VALUES (1,'first',1), (2,'second',2), (3,'third',1), (4,'fourth',2), (5,'to be or not to be',1);
COMMIT;
```

2. Transaction A starts and sets its own snapshot (Isolation Level: RR)

```
COMMIT;
SET AUTOCOMMIT  = 0;
SET TRANSACTION
ISOLATION  LEVEL REPEATABLE READ;
SELECT * FROM T WHERE i = 1;
```

3. Transaction B starts and does its part (Isolation Level: RC)

```
COMMIT;
SET AUTOCOMMIT  = 0;
SET TRANSACTION
ISOLATION  LEVEL READ COMMITTED;
UPDATE T SET s = 'Updated by B' WHERE id = 1;
INSERT INTO T (id, s, i) VALUES (6, 'Insert Phantom', 1);
UPDATE T SET s = 'Update Phantom', i = 1 WHERE id = 2;
DELETE FROM T WHERE id = 5;
SELECT * FROM T;
```

4. Transaction C starts and conducts its first uncommitted read (Isolation Level: RU)

```
COMMIT;
SET TRANSACTION  ISOLATION  LEVEL READ UNCOMMITTED;
SELECT * FROM T;
```

5. Transaction A conducts a number of updates and it is placed on hold

```
SELECT * FROM T WHERE i = 1;
INSERT INTO T (id, s, l) VALUES (7, 'inserted by A', 1);
UPDATE T SET s = 'updated by A inside the snapshot' WHERE id = 3;
UPDATE T SET s = 'updated by A outside the snapshot' WHERE id = 4;
UPDATE T SET s = 'updated by A after it was updated by B' WHERE id = 1;
```

6. Transaction C conducts its second uncommitted read

```
SELECT * FROM T;
```

7. Transaction B commits (unblocking Transaction A)

```
COMMIT;
```

8. Table T as seen by client B

```
SELECT * FROM T;
```

9. Transaction C conducts its third uncommitted read

```
SELECT * FROM T;
```

10. Transaction A's snapshot

```
SELECT * FROM T WHERE i = 1;
```

11. Transaction A's view on table T

```
SELECT * FROM T;
```

12. Transaction A attempts to update phantoms (2) and ghost

```
UPDATE T SET s = 'updated after delete?' WHERE id = 5;
UPDATE T SET s = 'update inserted phantom?' WHERE id = 6;
UPDATE T SET s = 'update phantom update?' WHERE id = 2;
```

13. Transaction C conducts its fourth uncommitted read

```
SELECT * FROM T;
```

14. Transaction A's view on table T

```
SELECT * FROM T;
```

15. Transaction's A view on original snapshot

```
SELECT * FROM T WHERE i=1;
```

16. Transaction A deletes the inserted phantom (already updated by A)

```
DELETE FROM T WHERE id=6;
```

17. Transaction C conducts its fifth uncommitted read

```
SELECT * FROM T;
```

18. Transaction A commits

```
COMMIT;
```

19. Client A's view on table T

```
SELECT * FROM T;
```

20. Transaction C conducts its sixth uncommitted read and commits

```
SELECT * FROM T;
COMMIT;
```

21. Client C's view on table T;

```
SELECT * FROM T;
```

22. Client's B new xaction defaults to RR

```
SELECT * FROM T;
```

23. Client B commits and gets synchronized with the DB content

```
COMMIT;
SELECT * FROM T;
```

Παρατηρήστε/σχολιάστε τις περιπτώσεις των φαντασμάτων (phantoms) και οπτασιών (ghosts) στα παραπάνω.

Μέρος Β: Ιστορίες UPDATE

Client A
Client B

1. Initialization phase

```
mysql
SET AUTOCOMMIT = 0;
```

```
mysql
SET AUTOCOMMIT = 0;
```

```
DROP TABLE T;
CREATE TABLE T (id INT NOT NULL PRIMARY KEY, s VARCHAR(40), i SMALLINT);
insert into T(id,s,i) VALUES (1, 'first',1), (2, 'second',2), (3, 'third',1), (4, 'fourth',2);
COMMIT;
```

2. Transaction A and B start in RR-RR

```
COMMIT;
SET AUTOCOMMIT = 0;
SET TRANSACTION
ISOLATION LEVEL REPEATABLE READ;
SELECT * FROM T;
```

```
COMMIT;
SET AUTOCOMMIT  = 0;
SET TRANSACTION
ISOLATION  LEVEL REPEATABLE  READ;
SELECT * FROM  T;
```

3. Transaction A updates row with id=2

```
UPDATE T
SET s = 'updated by A'
WHERE id=2;
SELECT * FROM T;
```

4. Transaction B updates row with id=3

```
UPDATE T
SET s= 'updated by B'
WHERE id=3;
SELECT * FROM T;
```

5. Transactions A and B ROLLBACK and re-start

```
ROLLBACK;
SET AUTOCOMMIT  = 0;
SET TRANSACTION
ISOLATION  LEVEL REPEATABLE  READ;
SELECT * FROM  T;

ROLLBACK;
SET AUTOCOMMIT  = 0;
SET TRANSACTION
ISOLATION  LEVEL REPEATABLE  READ;
SELECT * FROM  T;
```

6. Transactions A updates row with s='second'

```
UPDATE T
SET i= 22
WHERE s='second';
SELECT * FROM T;
```

7. Transaction B updates row with s='third'

```
UPDATE T
SET i=33
WHERE s='third';
SELECT * FROM T;
```

8. Transactions A and B ROLLBACK and re-start

```
ROLLBACK;
SET AUTOCOMMIT  = 0;
SET TRANSACTION
ISOLATION  LEVEL REPEATABLE  READ;
SELECT * FROM  T;

ROLLBACK;
```

```
SET AUTOCOMMIT  = 0;
SET TRANSACTION
ISOLATION  LEVEL REPEATABLE  READ;
SELECT * FROM T;
```

9. Index created on T(s)

```
COMMIT;
SET AUTOCOMMIT  = 0;
SET TRANSACTION
ISOLATION  LEVEL REPEATABLE  READ;

CREATE  INDEX s_index ON T(s);

COMMIT;
SET AUTOCOMMIT  = 0;
SET TRANSACTION
ISOLATION  LEVEL REPEATABLE  READ;
```

10. Transaction A updates row with s='second'

```
UPDATE T
SET i= 22
WHERE s='second';
SELECT * FROM T;
```

11. Transaction B updates row with s='third'

```
UPDATE T
SET i=33
WHERE s='third';
SELECT * FROM T;
```

12. Clients A,B COMMIT and see the same T

```
COMMIT;

COMMIT;

SELECT * FROM T;
SELECT * FROM T;
```

Παρατηρήστε/σχολιάστε τη διαφορποίηση που προκαλεί στο συγχρονισμό των συναλλαγών η ύπαρξη του ευρετηρίου (index).

~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~