1. Исследование уровня транзакций Snapshot

T1 – Tn – транзакции

T1: уровень Snapshot

```
select * from types where type_id=0
```

T2: уровень Read Commited

```
update types set title_type='Contract' where type_id=0
```

Запустим на выполнение T1, и T2 и подтверждая T2, получим:

TYPE_ID	TITLE_TYPE
0	New title

Это показывает то что T1 использует старую информацию, ее слепок до выполнения всех транзакций. Вернем значение обратно в New title.

2. Исследование Read Commited

T1: read commited

T2: rw stability

```
update types set title_type='Contract' where type_id=0
```

Запускаем T2 не подтверждая, получаем результаты первой транзакции:

TYPE_ID	TITLE_TYPE
0	New title

Попробуем записать первой транзакцией что-нибудь в это таблицу.

Messages

Unsuccessful execution caused by system error that does not preclude successful execution of subsequent statements.
lock conflict on no wait transaction.

Это говорит о том что доступ к этой таблице возможен только для чтения, а на запись она заблокирована транзакцией T2 которая стартовала как RW stability.

Теперь подтвердим T2. Получим результат первой транзакции:

TYPE_ID	TITLE_TYPE
0	Contract

Как видно данные обновились, тогда как в случае с Snapshot нам показывалась старая информация, можно сказать что read commited показывает все подтвержденные транзакции, а снимок - соответственно показывает данные которые были подтверждены до запуска транзакции.

3. RW Table Stability

Данные уровни существуют для предотвращения одновременного доступа к ресурсу, например одной и той же записи в таблице. Если запустить транзакцию в режиме **RW stability** то это приведет к тому, что другие транзакции не смогут получить доступ к данной таблице для записи, а только для чтения. Продемонстрируем:

T1: RW stability

```
update types set title_type='Contract' where type_id=0
```

T2: Read commited

```
update types set title_type='Others 5' where type_id=4
```

После выполнения T2 и не завершения T1 получим:

Messages

Unsuccessful execution caused by system error that does not preclude successful execution of subsequent statements.
lock conflict on no wait transaction.

T2:

```
select first 2 * from types
```

TYPE_ID	TITLE_TYPE
0	Contract_5
1	Contracct_additional

Как видно, доступ на чтение есть.

T2:

```
update types set title_type='Others 5' where type_id=4
```

Завершим T1 и запустим T2, получим:

TYPE_ID	TITLE_TYPE
0	Contract_5
1	Contracct_additional
2	TZ
3	Fixes
4	Others 5

Как видно данные обновились, тоесть T2 получило доступ к таблице после окончания транзакции T1.

T1: RW stability

```
select * from types where type_id=0
```

T2: RO stability

```
select first 2 * from types
```

TYPE_ID	TITLE_TYPE
0	Contract_5
1	Contracct_additional

Таким образом операции выборки из БД не влияют на доступ к таблицам в режиме RW stability.

4. RO Table Stability

T1: **RO stability**

```
update types set title_type='Contract_8' where type_id=0
```

Результат:

Messages

The INSERT, UPDATE, DELETE, DDL or authorization statement cannot be executed because the transaction is inquiry only.
attempted update during read-only transaction.

Что говорим о том, что это должна быть операция чтения.

T2: RW stability

```
update types set title_type='Contract_5' where type_id=0
```

Messages

Unsuccessful execution caused by system error that does not preclude successful execution of subsequent statements.
lock conflict on no wait transaction.

T2: Read commited

Messages

Unsuccessful execution caused by system error that does not preclude successful execution of subsequent statements.
lock conflict on no wait transaction.

Собственно до закрытия транзакции T1 нет возможности ни писать, ни читать из базы.

T1: **RO Stability**

```
select first 2 * from types
```

T2: RO stability, RW stability

```
select * from types where type_id=0
```

Данные варианты возможны.

T2: RW Stability, Read commited, Snapshot

```
update types set title_type='Contract_8' where type_id=0
```

Это невозможно.

Messages

Unsuccessful execution caused by system error that does not preclude successful execution of subsequent statements.
lock conflict on no wait transaction.

Выводы:

В работе исследовались различные уровни изоляций для firebird 2.5. В качестве вывода можно сформулировать некоторые правила по использованию различных уровней изоляции.

1. Когда требуются актуальные данные, необходимо использовать Read commited. Но данный тип транзакции не обеспечивает повторимости чтения, так как подтвержденные данные становятся видны этой транзакции.
2. Когда необходимо предотвратить изменение одних и тех же данных, можно выставить на эту транзакцию уровень изоляции RW stability, это позволит запретить запись в таблицу других процессов, но оставит чтение.
3. Слепок может обеспечит повторимость чтения, но обладает не самой актуальной информацией, так как создает список транзакций завершенных до ее старта.