Feature story of Transaction

Transaction is an integrity basic unit of Data Base working.
Transaction should satisfy **ACID** properties:

- **A**tomicity means that the operations inside of one transaction will fully complete or none of them will execute.
- **C**onsistency secures that Data Base will stay in concerted state regardless of the way of transaction completion: <COMMIT> or <ROLLBACK>.
- **I**solation provides the independence of actions between two executing competitive transactions, i.e. the work of two competitive transactions has no influence on each others execution.
- **D**urability implies that the fixed group of commands will be preserved in Data Base regardless of hardware or software errors.

The transaction starts implicitly when you begin to work with Data Base. But you can set it evidently with operator <START TRANSACTION>. By virtue of properties, transaction can be complete either of executing of all commands with operator <COMMIT WORK> or default on every operation with operator <ROLLBACK WORK>.

Isolation levels

Transactions support different isolation levels.
ANSI (**A**merican **N**ational **S**tandards **I**nstitute) supports 4 isolation levels of transactions.
The definitions of isolation levels were formulated in terms of "phenomenon":

- **R**ead **U**ncommitted (**RU**) – not fixed reading or phenomenon of «dirty» reading.

Imagine, User1 started transaction T1 for modifying data in table <EMPLOYEES>. User2 started another transaction T2 for reading data from this table. As a result, User2 can "see" all changes that were made with T1. In such a way, User2 can get data that are not really exist in Data Base if T1 would complete with operator <ROLLBACK>.

- **R**ead **C**ommitted (**RC**) – reading of fixed data or phenomenon of «unrepeatable» reading.

Suppose, User1 started transaction T1 for modifying data in table <EMPLOYEES>. User2 has started another transaction for reading data from this table. As a consequence, User2 will not "see" changes that were made by T1 until User1 would not fix changes he made. That is to say User2 can get different sets of data before and after fixing transaction T1 by User1.

- **R**epeatable **R**ead (**RR**) – repeatable reading or phenomenon of "phantom".

If User1 changes the data in table <EMPLOYEES> with transaction T1 then User2 will not "see" changes that were made with transaction T2 even if T1 would be completed with <COMMIT> operator. But with this isolation level can be phenomenon of "phantom".
The example of revealing of phantom:

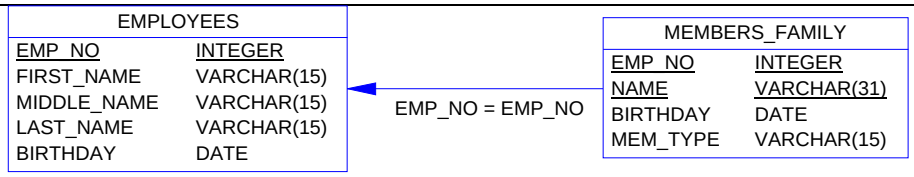
```
SELECT E.LAST_NAME || ` ` || E.FIRST_NAME, E.BIRTHDAY
FROM EMPLOYEES E
WHERE 5 < (     SELECT COUNT(*)
                FROM MEMBERS_FAMILY MF
                WHERE MF.EMP_NO=E.EMP_NO)
```

Transaction T1 reads data from table <EMPLOYEES> in regime of isolation level - <REPEATABLE READ>. Transaction T2 was started for modifying data in table <MEMBERS_FAMILY>. T2 changed data and fixed modification of data that are suitable to condition of query selection of transaction T1. In the result the User1 can get different sets of data, if he will execute the query above again.

- **S**erializable (**S**)- serialization.

The result of parallel executing transactions T1 and T2 would be the same as if transactions T1 and T2 would execute it consecutively.

*All examples in this article were considered on the following physical models:



- If by the terms of task it is necessary to "see" invariable data before, during and after execution of phases then cleverly to use transaction in **R**epeatable **R**ead (Snapshot) isolation regime.
  The most typical examples are analytical and computed tasks.

- If you need to "see" current data on present moment of time then you should use transaction in **R**ead **C**ommitted isolation regime.
  For example, the work of operator or dispatcher.

InterBase supports the following isolation levels of transactions:

- READ COMMITTED RECORD_VERSION
- READ COMMITTED NO RECORD_VERSION
- SNAPSHOT
- SNAPSHOT TABLE STABILITY