Using Stored Procedures and Triggers

Note:
Most if not all these considerations apply to normal SPaTs usage without Adabas Vista too!

- General Adabas usage
- Transactional usage

General Adabas usage

Commands issued by *participating* triggers affect the same Adabas client session as the originating client. It is the responsibility of the designer/programmer (of the originating client and participating trigger) to make sure Adabas usage does not collide in any way.

For example, it is possible for the originating client program and the participating trigger program to use the same partitioned, translated or ordinary Adabas file – but this means Adabas Vista will issue competing OP (or other) for the same session which will impact one or both programs causing unpredictable and error results.

The programmer must make sure all Adabas activity for these two peers of one session do not interfere with each other. Adabas Vista goes some way to help with this by making sure collisions do not occur within the database where the trigger fires ("the home database"), but Vista is unable to help with any other database access.

Transactional usage

As already explained commands issued by *participating* triggers (and the original client) act as the same Adabas session and may affect databases other than the home database. If these involve transactional commands then this means either or both of the two peers may have transactions in databases beyond the scope of the home database.

In industry terms this is a distributed transaction. This presents a problem in that only one of the two peers can control the completion of the transaction (commit or backout) which means the default action is that all the additional databases modified by the peer that does **not** control transaction completion are ignored causing unpredictable and error results – including data integrity issues.

Adabas Vista helps in some way by making sure the distributed parts of the transaction in the peer where the completion is issued are taken care of, in a serial way (multiple `ET` or multiple `BT` commands in series) but that does not take care of the problem on the other peer side.

The way to address this problem completely is to adopt usage of Adabas Transaction Manager because it is able to make sure all parts of the distributed transaction caused by both peers are completed safely. Please refer to the Triggers section of the Programmers’ Guide in the Adabas Transaction Manager documentation for more information.

It is the responsibility of the designer/programmer to make sure the appropriate processing takes place. Some possibilities would be:

1. Deploy Adabas Transaction Manager so that distributed transactions can be freely used.

2. Make sure only the home database is modified even if Vista translation and partitioning is used by either or both of the peers.

3. Make sure only one of the peers modifies databases beyond the home database – and at least make sure Adabas Vista is there to propagate serial `ET`/`BT` commands automatically. This does not provide full data integrity (only Adabas Transaction Manager can accomplish this) but at least most of the time it will be okay because Adabas is so reliable.