Well-formed distributed transaction enforcement

A transaction is considered to be *distributed* when it modifies more than one database (RM) in the same transaction. This means the Adabas command interface has to be used carefully to ensure compliance with correct distributed transaction behavior. In other words, a well-formed distributed transaction is one where the modifications to all the databases are all applied (committed) together - and it is the programmer's responsibility to make sure this happens.

Natural ensures that distributed transactions are well-formed because it automatically generates a series of `ET` commands for all modified databases when it encounters an `END TRANSACTION` statement. There is no programmer control over this, nor should there be. Natural keeps a list of all modified databases internally so when an `END TRANSACTION` is encountered it simply iterates through the list issuing `ET` commands. Therefore, Natural truly enforces the use of well-formed distributed transactions

None of what Natural does guarantees integrity of distributed transactions, but it does ensure that where they are used they are well-formed. To guarantee integrity you need to adopt Adabas Transaction Manager, see Automatic distributed transaction integrity.

Of course, 3GL programming is not as easy as Natural. The following 3GL logic is illegal in terms of distributed transaction programming because it applies a modification to one Adabas database but undoes the modifications to the other - this is poor (ill-formed) distributed transaction programming:

1.	Modify data in Adabas database A
2.	Modify data in Adabas database B
3.	Issue ET command to Adabas database A
4.	Issue BT command to Adabas database B