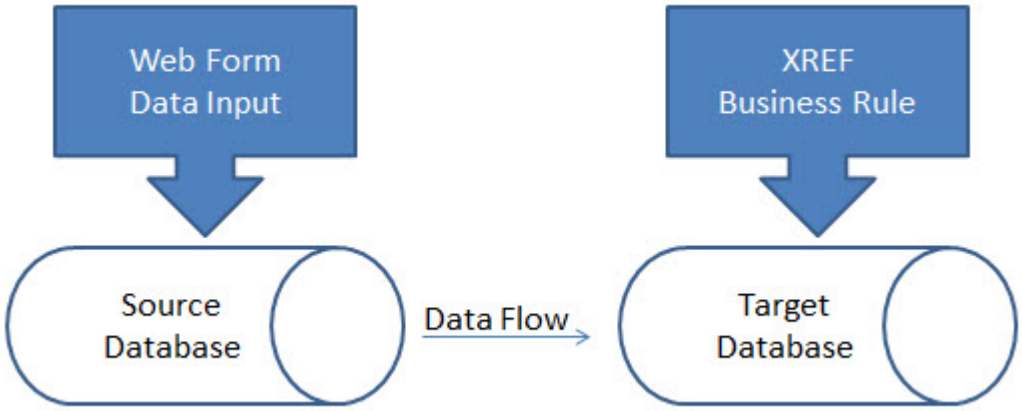
Meet XWRITE, XREF's New Big Brother

The introduction of Hyperion 11.1.2 has some fantastic improvements. Many of these have been long awaited. The next few articles on In2Hyperion will describe some of the enhancements to Hyperion Planning, Hyperion Essbase, and Hyperion SmartView.

XREF Background

If you have been developing Planning applications, you are probably very familiar with the XREF function. This function is used in business rules, calculation scripts, and member formulas. It provides a method to move data from one plan type (Essbase database) to another plan type. It is executed from the target database and pulls the data from the source. XWRITE was actually introduced in later versions of 11.1.1.x, but is very stable in 11.1.2.x. XWRITE is executed from the source and pushes data to the target. This function is a huge improvement over XREF.

XREF will copy data to a target database and must be executed from the target database. The function pulls data rather than pushing it. This causes two challenges. Normally, the data is entered in the source database and is copied to the destination database. When a Planning web form is saved, it can only execute a calculation on the database the web form is connected to (at least in older version – stay tuned). This means an XREF function cannot be used when the form is saved. The user has to go to another form, or execute a business rule manually, for the data to move.

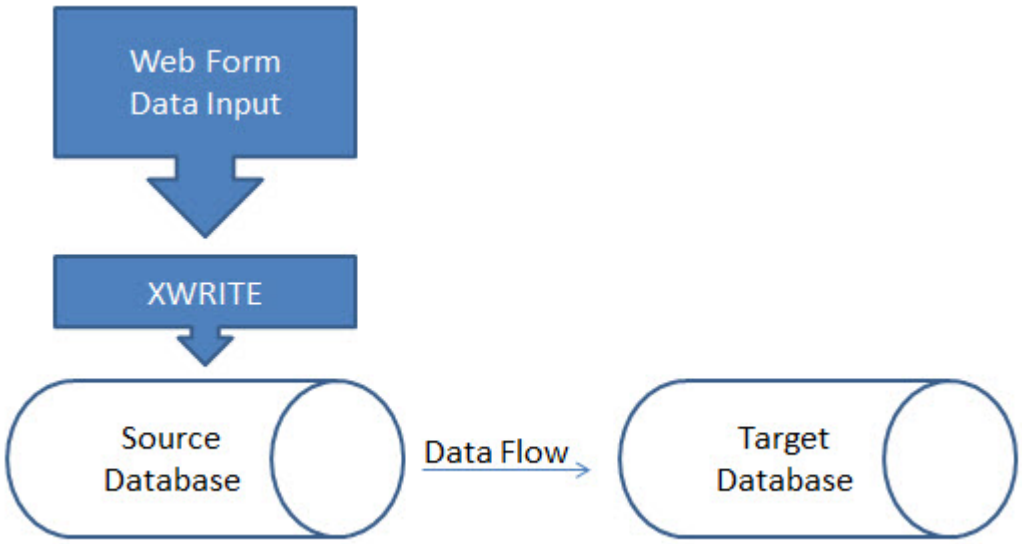


The larger issue with XREF is accounting for block creation. Remember, XREF pulls data from a source. The destination may not have blocks that exist where the data will reside. XREF does NOT account for the creation of the blocks if blocks don't exist. XREF must be used in conjunction with the CREATEBLOCKONEQUATION setting. This is acceptable when fixing on very finite levels of data, but execution on larger amounts of data results in an extremely slow data movement process. Essbase is responsible for the slow data movement process because it traverses all possible sparse member combinations to validate existence of data on the source. Normally, data exists at a very small percentage of the possible blocks. In addition to the slow data movement process, it's worth noting that the XREF function can also create blocks in your database which are unnecessary; ultimately increasing the size and decreasing the speed of your application.

Welcome to XWRITE

XWRITE is the opposite of XREF. Rather than using XREF to pull the data from the target, XWRITE enables you to push data from the source. Pushing data resolves the issues which XREF creates.

When XWRITE is executed from a web form, thus pushing data from the source to the target, there's no longer a need to account for this process with two web forms or the manual execution of a business rule.



Since XWRITE is executed from the source, there's no longer a need for looking at every possible sparse member combination on the target. Using a FIX statement enables Essbase to decipher which blocks need to be copied, removing the guesswork and subsequently the requirement of CREATEBLOCKONEQUATION. Utilizating the XWRITE function results in faster processing and efficient block creation.

Prior to XWRITE, my preferred method of data movement involved exports from the source and imports to the target; thus eliminating the need for the XREF function. The introduction of XWRITE has reduced the need for a data export/import process.