Debunk The Myth: Never Fix On Dense Members

The generic rule in Essbase is that calculations FIX on sparse members because sparse members are what define the number of blocks. When you want to limit the members of the block on which the calculation is executed, an IF statement is appropriate.

Quick Overview of Dense and Sparse

If you are unfamiliar with the concept of dense and sparse, here is a quick overview. A data block in Essbase is constructed from the dense dimensions of the database. The number of members in each dense dimension impacts the size of each data block. The combination of a member in each sparse dimension is what defines a block. The number of members in the sparse dimensions directly correlates to the number of blocks that may exist.

For a more detailed overview, reference Sparse, Dense, and Blocks For Dummies.

Comparison To Relational Database

A FIX is a lot like a SELECT statement in a relational database using a WHERE clause. The WHERE clause limits the number of records, an Essbase FIX limits the number of blocks for which an action is taken. An IF statement in Essbase is similar to a CASE statement in a relational database in that it executes on all the records and acts only when a criteria is met.

Limiting the records

Relational Example

```
UPDATE table_name
SET Salary=Annual Salary * Merit Increase
WHERE Year=2013;
```

Essbase Example

```
FIX(“2013”)
  Salary = “Annual Salary” * “Merit Increase”;
ENDFIX
```

Executing on all records when they meet criteria

Relational Example

```
SELECT
   CASE
       WHEN Year = 2013 THEN Salary = Annual Salary * Merit
Increase
      ELSE Salary = Salary
   END
FROM table_name
```

Essbase Example

```
IF(@ISMBR(“2013”))
  Salary = “Annual Salary” * “Merit Increase”;
ENDFIX
```

When running an UPDATE query, limiting the number of records is more efficient than running the query on all the records and checking for specific criteria to execute the logic.

Why Fix On Dense?

The reason we are taught to FIX on sparse dimensions and use IF on dense dimensions is that a FIX will improve performance by limiting the number of blocks on which the calculation executes. There is no reason to FIX on dense dimensions because it isn’t limiting the number of blocks on which the calc is executed.

Forget all that!

Calculations still run for every intersection, not JUST the intersection of sparse members. Assume a calculation fixes on one intersection of sparse members. Also assume that there are 20 measures and 12 periods that are stored, and both dimensions are dense. The following calculation

```
Salary = Annual_Salary * Merit_Increase;
```

would run on every dense combination, so it would execute 240 times (12 x 20). You can easily prove this by incrementing the value of one dense member by 1.

```
Salary = Salary   1;
```

If Salary starts as #Missing, or 0, and the above line is executed, Salary will be 20 for each month.

Solution

This can easily be resolved. Since you only want the calculation to execute one time on the block, add one member from the measures dimension to your fix statement. This member doesn't have to be the member you are calculating. I typically will fix on a generic measure to eliminate confusion. Change the calculation to the following.

```
FIX(No_Measure)
  Salary = Salary   1;
ENDFIX
```

Make sure Salary is set to #Missing or 0, and execute the new calculation. When the new calculation script is executed, you should see a value of 1 for every month.

In a situation where Salary = Annual_Salary * Merit_Increase, the result will be correct regardless of whether the calculation fixes on one measure, but the performance will be far worse when executed on every Measure because it will run

the same calculation multiple times.