

All About Record Matching for GoldMine, Using GoldBox - Short Version

PART1: GENERAL INFORMATION ABOUT MATCHING

WHAT MATCHING DOES: Suppose we want to Import a **Source** file of people's names and addresses to GoldMine. We believe that we may have some of these people in GoldMine already, and we don't want to add duplicate Contact records to the database. **So, we actually want to do an Import (add records)**, with an **Update option** (look for apparent duplicates and, when duplicates are found, do **NOT** add new records for them). With an Update, you have the additional options of ignoring a matching Source record entirely; or of using the data in the Source record to over-write the corresponding data in the matching GoldMine record.

But how do we know if two records are duplicates? We **define matching criteria** as part of the process of setting up the Import/Update. Then, when we run the procedure, before each Source record is Imported, our Import/Update routine will check it against our pre-defined matching criteria. Because this is a simple example, let's just say that we want to recognize a match if a GoldMine record can be found with the **same Contact name as the "Name" field in the Source record we are about to Import**. All we have to do is "map" the Name field of the Source file to the Contact field in GoldMine; and then select the **Contact Index** (one of GoldMine's standard Indexes) as our matching criteria. (You'll soon see that this matching scheme is **TOO** simple for the real world).

When the Import/Update runs, it will evaluate every record in the Source file as to whether or not it's a duplicate for an existing GoldMine record, using the Index. If it isn't, it will Import the Source record as a new GoldMine record. But if it is determined to be a dupe, it will either be ignored, or will over-write existing GoldMine data, depending on the choice we have made in setting up the Import/Update.

SO WHY ARE DUPES SUCH AN IMPORTANT THREAT, ANYWAY? They make our data ambiguous; it's like the proverbial man with two (or more) watches...he is never really sure exactly what time it is. But with data, the uncertainty can cost real money and vital good will. One GoldMine User can attach an important Detail record to a Contact record; and a second GoldMine User may **fail to find that Detail because it's attached to a duplicate record**. Likewise, the "found" value of a field may depend on which of one or more duplicates exist. Mail may be sent to an incorrect address; AP actions may be duplicated; all sorts of consequences can result from duplicate records.

So, when we do Imports, we need to do our best to eliminate the creation of duplicate records. And so we need to periodically do **Merge-Purges** on our data, to combine duplicate records that already exist **into one, correct GoldMine record**.

RECORD MATCHING TECHNIQUES: GoldMine offers only one type of matching for its version of **Import/Update** setups¹: **GoldMine Index matching** (using one of GoldMine's standard Indexes). In the above example, we used a GoldMine Index match by selecting the **Contact Index**. If the value in the **Name** field of a record in the **Import Source file** is an **EXACT MATCH** for a value in GoldMine's **Contact Index**, then a match is "made". An Update (rather than an Import) will be done with that Source record.

The problem with GoldMine Index matching is quickly obvious: just because two people have the same name, it does not follow that they are the **same person** (and **that's** what matters). If it were possible to add **additional data fields** to GoldMine's matching criteria (as can be done with GoldBox's Custom Matching), it would at least be possible to improve the **safety** of the match criteria. But with GoldMine alone, it isn't possible to do that, and that makes GoldMine's Import with Update option unsuitable for serious data management (unless a "**perfect**" match field is available; more about that next).

By contrast, GoldBox offers three different kinds of matching for a **Main Contact Import/Update**:

- **Index matching** (same as GoldMine, and almost never used, unless the "**perfect**" match fields **Accountno** or **Recid** matching are available in the Source file)

¹ GoldMine's version of an **Import/Update** is an **Import with the "Profile Options" button on the Field Mappings screen pressed**, to reveal the "Select the match field to be used" drop-down menu. Selecting a field name from that menu actually selects the corresponding GoldMine Index to be used for matching.

• **Custom matching** – this enables selection of **multiple fields** (up to a max field length of 100 characters) for use in an “exact text Custom Index match” regime. GoldBox builds its own custom matching Index for this purpose. Also, for this option only, GoldBox offers two “Smoothing” functions that help turn **nearly exact** matches into matches, in a safe manner...

SmoothComp does a virtual replacement or drop out of certain common details of a company name (like Inc., Co., etc), to “make” matches that would otherwise be missed, based on those details alone.

SmoothName does a virtual replacement of the Contact’s **first** name (like “Robert” with “Bob”), and then attempts to make the match based on those virtual values. Neither function actually changes the actual data in any way; it’s all done virtually.

Competitive offerings, including GoldMine’s Merge/Purge, use alternative matching approaches to make matches that compensate for minor differences in the actual matching criteria. While they have their differences, for purposes of this discussion it’s fair to consider “**fuzzy logic**” matching; **Soundex** (a primitive form of fuzzy logic); and the **Points System** to be the same in this regard: **they all involve accepting matches that are not exact**. All are based on a “degree of sameness” between one record and another, based on an algorithm that tests each of the records, and compares their “scores”. During the setup of the procedure, you have to decide how high that score must be in order for a match to be “made”.

GoldBox’s Smoothing functions work differently. **When using them, GoldBox still requires exact matches**. The smoothing functions make virtual changes that are under your control, **based on rules or tables that you manage**. These virtual changes can often help make matches that would be obscured by minor, predictable “noise” in the data; but once the virtual “corrections” have been made, an **exact match** of the virtual data is still required. I firmly believe that GoldBox’s Custom Matching using the Smoothing Functions (with well-managed Smooth tables) is the **SAFEST, MOST EFFECTIVE** form of data matching that is available for GoldMine data, period.

• **Direct SQL Query Matching**² – this design permits the use of a **series of matching attempts** on each Source record, and the recognition of a “made” match if **ANY** of the attempts is satisfied. Naturally, it’s important that **EACH** of these multiple criteria be a **SAFE** option (more on the important subject of safety soon).

THE IMPORTANCE OF SAFETY: In addition to **Import/Update of Main Contacts** and **Merge-Purge**, GoldBox offers **Import/Update of Tab records**, like Additional Contacts, Details, History, Pending, etc. This is something **GoldMine does not offer at all**. Because of the variety of Tab records (each of which requires its own matching criteria), I will not cover that in detail in this paper. The principles, however, are the same, and GoldBox offers suggestions with these Setups. Matching is also used in **Deduping** individual tables.

Matching technology generally relies entirely on an **automated** matching process using criteria that were chosen prior to running the process. It follows that, for Import/Updates, it’s **important** to choose matching criteria that are as **SAFE** as possible. It is better to use matching that may permit the creation of a few duplicate records, if that matching also ensures that **NO false matches** are made and processed.

Let’s go ahead and discuss matching safety thoroughly, because it affects **ALL** uses of record matching.

While different programs may use a variety of techniques to define and locate matches, there is a sort of **Golden Rule of Matching** that should **always** be paramount:

It is better to miss a true match than it is to find and process a false match

There are **no exceptions** to this rule, because (short of complete backup restoration immediately after a bad procedure), **repairing the damage done by completing a procedure based on a false match is either extremely difficult, or impossible**.

² Direct SQL Query Match is for versions of GoldBox where the GoldMine data resides in SQL. For GoldMine data in dBase, GoldBox-5 has Direct Search Filter Matching, which is comparable to Direct SQL Query Matching, but for dBase.

Besides, you have no assurance that such an error will be discovered quickly. You may find that you have acted on incorrect data again and again, before discovering the problem; and that, by then, any hope of reconstructing the truth is long gone.

The consequence of **missing a true match** is that a duplicate record is added to (or left in) the database; not ideal, certainly, but not a disaster, either. The consequences of finding and **processing a false match**, however, are much more serious. They include:

1. **Overwriting good data** in an existing Contact record **with false data**. Depending on how the Import/Update, Merge-Purge or Dedupe is set up, this can mean that an existing database record suddenly acquires false Contact data; false History; false Activities; and so on. And that means that an important Contact record can suddenly become useless; or even worse, seriously misleading.
2. **Failure to create** (in the case of an Import/Update); or **Deletion of** (in the case of a Merge-Purge or Dedupe); **a valid GoldMine record**. This can result in a lost opportunity for that Contact person. Another possibility: with an Import/Update, the data that was misdirected to a falsely matched Contact may have been intended for an existing Contact; but it will never get there, because of the false match.

These kinds of errors can be disastrous to a database, and **MUST** be avoided! Riskier matching criteria can mean more duplicates found, and it's tempting to interpret that as a better result. But if the price you pay for removing more dupes is processing some false matches, **that price is just too high**.

PART 2: EXAMPLES OF SAFE AND EFFICIENT MATCHING CRITERIA

• **For Import/Updates of Main Contacts:** Main Contact records represent people, so the more information we can put in our matching criteria about **who the Contact is; where the Contact is located; and how we can reach the Contact**, the better. Here are 2 examples, using **Custom Matching** (with data residing in SQL):

SmoothName(Tgt->Contact) + PhoneStrip("Tgt->Phone1") + Left(Tgt->Zip,5)

SmoothComp(Tgt->Company, "C") + SmoothName(Tgt->Contact) + PhoneStrip("Tgt->Phone1")

Each of the above contains the **SmoothName()** function, which increases our chances of making a match by accepting nicknames, like "Bob" for "Robert". Each also contains **Phone1** (stripped of any formatting characters). Together, this is probably enough to ensure safety (i.e. no false matches). But we add an additional field to each, to further ensure safety.

But doesn't the use of 2 matching expressions imply that we are running multiple instances of the same Import/Update? Indeed it does!

One of the cool things about GoldBox is that you can **Write Back** any field (I always use the Accountno) of a record that has been processed to the Source file. So, for example, if the first record in my Source file finds a match on my first run, using the first matching expression above, I can have GoldBox Write Back the Accountno of that matching record (to a field I designate) in the Source. On my second, and all subsequent passes, I **filter out any record in the Source file that contains an Accountno**; this eliminates that first record from further consideration.

This means that I can make as many copies of my original Setup as I want, using a different matching expression for each; dump them all into a Q-file; and run them, one after the other. On the **last** of these Import/Update Setups (only), I check the **ADD No-matches** box, and let GoldBox create new records from all the Source records that failed to find a match in any of my attempts.

The result is an integrated system of Import/Update that is maximized for **SAFETY**, yet also maximized for **EFFICIENCY** (that is, for making the greatest possible number of safe matches, because of the multiple combinations of fields that we try). It's not out of the question to use a half dozen or more Setups, especially when the data is of unknown quality (like from a web form or a trade show). Such a system is easy to construct; and thanks to the Q-file, easy to run. Of course, more time is required to run such systems; you may not use that many Setups for Import/Updates that must be done every day.

One important detail: when running any procedure that involves matching, it's **crucial** to use the Filter capability that GoldBox offers in the Setup to **eliminate any Source records where ANY of the matching fields is empty**. Empty fields always match one another, and so considering a record with an empty matching field is asking for big trouble!

In addition to Custom Matching, you can use GoldBox's 3rd type of matching. In SQL versions, this is called **Direct SQL Query Matching**. (In GoldBox 5, the dBase version, it's called **Direct Search Filter Matching**; a similar idea but executed different; not covered in this paper). Here's an example of a Direct SQL Query Match:

```
('<<Gxi->Company>>' = CONTACT1.COMPANY' AND '<<Gxi->Contact>>' = CONTACT1.CONTACT AND
'<<Gxi->Phone1>>' = CONTACT1.PHONE1)
OR
('<<Gxi->Contact>>' = CONTACT1.CONTACT AND '<<Gxi->Phone1>>' = CONTACT1.PHONE1) AND
'<<Left(Gxi->Zip, 5)>>' = LEFT(CONTACT1.CONTACT))
```

This is similar in idea to the Custom Match expressions from the previous page. But note that we cannot use **SmoothComp()** or **SmoothName()**; and we cannot use the GoldBox function **PhoneStrip()**. As a result, this type of matching—while every bit as safe as Custom Matching—is less efficient; it will find fewer matches.

So, why might you use Direct SQL Query Matching instead of Custom Matching? I only use it when my Source file is relatively small (say, no more than 2,000 records); and the GoldMine table is much larger (say 100,000 records or more). Because Direct SQL Query does not have to build a Custom Index, it will operate much faster, **under those conditions**, than Custom Matching will. Still, if you have the option to use Custom Matching, it would always be the preference.

• **For Merge-Purge:** The exact same **Custom Match** options are available with Merge-Purge as are available with Import/Update. In fact, Custom Matching is the **ONLY** matching option available with Merge-Purge. There is no separate Source file, so there is no Write Back to the Source file.

SAFETY is even more critical for a Merge-Purge than it is for an Import/Update, because Merge-Purge actually deletes records from the database. In fact, a Merge-Purge that is done without all the necessary safeguards in place is probably the **single most damaging thing** that a User can do to a database. Control who uses it carefully!

This is not a tutorial on Merge-Purge, but I will mention a couple of important facts:

1. **There are 3 Modes of GoldBox's Merge-Purge:**
 - a. **Manual** - I do not use and will not discuss this Mode.
 - b. **Count (with optional Preview)** – This Mode is very important, and I'll discuss it more in a moment.
 - c. **Batch** – This is the “for keeps” Mode; it makes actual changes to GoldMine as a batch. It is something that can do a lot of good (or ill) to your data.
2. **No Merge-Purge should ever be run in Batch Mode unless it is completely SAFE.** You can make a Merge-Purge **safe by design**; or **safe by review**.
 - a. **Safe by design** means the **matching expression** used is so restrictive that it **cannot find false matches**. A Merge-Purge Setup that is safe by design can be run in Batch Mode at any time, provided that all the other details involved (like Backfill, and the other options of Merge-Purge) are filled properly.
 - b. **Safe by review** means that a responsible person has used the GoldBox Merge-Purge Primer Table to view the Preview of the Merge-Purge that was produced by running a Merge-Purge Setup in Count Mode with the Preview option checked. All questionable Dupe Group members should have been deleted from the Primer Table. Any changes in the default Survivor record should have been checked. And the **ONLY** way such a Setup should be run in Batch mode is if the **InMrgPrimer()** function has been added to the Setup's **Filter**; and if the **InMrgPrimer(1)** function has been added to the **1st Dupe Expression**. For further details, consult GoldBox's Help file.

Much as with Import/Updates, it is possible to assemble multiple Merge-Purge Setups into a Q-file, and run them in series. However, it is vitally important that **ALL** the Setups that are run this way **MUST be “Safe by design”**. All other Merge-Purge Setups **MUST be made “Safe by review”**.

• **For Dedupe One GoldMine Table:** This is used to find pairs of duplicate records within a single GoldMine table, and to delete one of them without any sort of merge or backfill. Generally, this is used for Tab records, like History, Details, Pending, etc. However, there is one special use for it: **to Dedupe the Contact2 table before running a Merge-Purge**.

This is **IMPORTANT**, because if you run a Merge-Purge on a database that contains Contact2 dupes (that is, two or more Contact2 records with the **same Accountno** value), the Merge-Purge process will actually **DESTROY** both the good record and its dupe, both Contact1 and Contact2!!! (The same is true with GoldMine’s Merge/Purge.) So, **ALWAYS** run a Dedupe of Contact2 before running a Merge-Purge!

However, when using Dedupe for all other purposes, record matching is more complicated than just selecting the **Accountno** field. In fact, it’s complicated enough that GoldBox will load a “suggested” match expression when you click on that button. Here’s an example, for a Detail:

TgtDup->Accountno+TgtDup->Rectype+TgtDup->Contact+TgtDup->Contsupref+Left(TgtDup->Address1,14)

Accountno is part of the expression, of course (you don’t want to Dedupe **BETWEEN** Contact records). But there are also a number of other fields that should be checked, to ensure you don’t miss a difference between records. As always, we really want to avoid false matches! Depending on how the Detail Type we are Deduping is set up, we may need to work with the above expression, to check other fields, too.

For Dedupe, we have essentially that same Custom match expression capability. We can use as many fields as needed, but the 100 character limit applies. GoldBox will create its own Custom Index based on the Expression used. Also, a filter is available, to help reduce the number of records that must be evaluated. There’s even a query option, although I confess I don’t use it.

Our overall strategy of **many safe attacks** at the problem is the sanest way to go. That’s true for Merge-Purge, but also for Import/Update...actually, for just about any process where it’s impossible or impractical to fix things with a single, simple effort. Fine-tuning data integrity is more like the careful chipping away that a sculptor does, than like the brutal drop of a forge.

Finally, it’s worth mentioning that we didn’t tackle most of the work required in creating an Import/Update, or a Merge-Purge Setup. Most of that effort doesn’t directly involve matching. It’s the **first** time you go through the full setup that’s the toughest. The repeated attempts with different matching criteria are comparatively very easy.

And so I encourage you to think about this: if you are a major multi-tasker, you should probably consider eliminating yourself as the person who creates these Setups. Whoever creates them will have a lot of balls in the air while doing the work, especially if there’s any complexity at all involved. Whoever does this kind of work should be able to do it almost interruption-free. The devil really is in the details on these things; you can get it 99% right, and still have a real misfortune on your hands.

If you can justify it, your best bet is to select one of your employees, and get that person some training. The cost of training can often be minimized by doing actual work in the training process. Of course, if you just can’t spare someone for that, the best thing to do is get an expert to do the work. Deciding “I’ll train myself and do it in my spare time” is probably not going to look like a good option in the long run.

The best news of all is this: however you make the investment to get things done right, you will end up with data that will save you money because it’s right. And, once you have the GoldBox Setups that got you there, you’ll have all you need to keep yourself there!