Data Migration Best Practices

An attempt to help you not delete your production database

1 - Data Migrations Checklist

The following is a semi-profanity-ridden attempt at explaining one way to do data migrations while following best practices. It is rather long and laced with colorful language. If you have read it already, or if you want to avoid the profanity, you can consult the following checklist in the *beautiful* table below.

Note that all elements are considered mandatory.

As a quick note, and a reminder even if you've read the whole version:

> DO NOT MODIFY SOURCE DATA FILES, EVER.

If you're doing data migrations, either use a script to modify the source files and save the edited version, or use excel workbooks that open the source file and then save the edited result elsewhere. Yes, even if the source is an excel file.

Why? Because sources change. People forget stuff, files aren't well formatted, shit gets broken, and people are human - meaning that one-time data import is actually going to be done multiple times. Edit the source file, and get to do everything all over again. Use scripts or workbooks to do the transformations ? Point that to the new source file and BAM Bob's your uncle.

Scripts you might want to use:

- OpenRefine
- SFXD's PSCSV
- Salesforce's official Data Migration Tool for cross-org data loading
- Amaxa for related objects, done by David Reed

Or, if you prefer excel, open a blank workbook, Import the source file via the "data" ribbon tab, select "from text/csv" (or whatever matches based on your source type), then save it as both:

- the construction excel,
- a NEW csv file after doing your changes in formula columns.

That way when you change the source file you can just open the construction book again and resave.

Action	Completed?
DO YOU HAVE A BACKUP	
Is it UTF-8 encoded	

Did you check it is readable and well formatted	
Does it have carriage returns stored as carriage returns, not as spaces	
Is it up to date	
Do you have a mapping for every object and field	
Did you determine an ExternalID for each object	
Did you determine source of truth (whether to overwrite or not) for reach field	
Did the client sign off on the mapping	
Do you have the source data	
Is it in a format your tool can read	
Are dates and date-times well formatted (yyyy-mm-dd || yyyy-mm-ddT00:00:00z) and are times exported in UTC	
Are field lengths respected (emails not longer than 80 chars, Names not longer than 40, etc)	
Do numbers have the right separators	
Do all tables have the required data for loading (Account Name, Contact Last Name, etc etc etc)	
Do all fields that have special characters or carriage returns have leading and trailing qualifiers (")	
Do all records have an external Id	
Did you do a dummy load with only one field mapped to make your sure tool can read the entire file	
Are you doing transformations	
Did you document them all	
Did you automate them so you can run them again with a click	
Did you read the LDV guide if you are loading more than 1M records	
Did you activate validation rules bypass	
Did you check all automations to deactivate any that should be, including email alerts	
Did you warn the client about when you would do the data load	
Did you warn the client about how long the data load would take	
--------- run the migration ----------	
Did you reactivate all automations	
Did you remove validation rule bypass	
Did you tell the client you were done and they could check	
Did you check the quality of the data	

2 - Data Migration Step-by-step - Before Loading

Introduction

You're going to have to map data from various sources into Salesforce. **IT'S THAT BIG MIGRATION TIME.**

Well let's make sure you don't have to do it again in two days because data is missing, or delete production data.

> Salesforce does not back up your data.

If you delete your data, and the amount deleted is bigger than what is in the recycle bin, if will be deleted forever. You could try restoring it via Workbench, praying that the automated Salesforce jobs haven't wiped your data yet.
If you update data, the moment the update hits the database (the DML) is done, the old data is lost. Forever.

If you don't have a backup, you could try seeing if you turned on field history.

If worst comes to worst you can pay 10 000€ (not joking, see here) to Salesforce to restore your data. Did I mention that Salesforce would give you a CSV extract of the data you had in Salesforce ? Yeah they don't restore the org for you. You'd still need to restore it table per table with a data loading tool.

But let's try to avoid these situations, by following these steps. These steps apply to any massive data load, but especially in case of deletions.

GENERAL DATA OPERATIONS STUFF

Tools

> Do not use Data Loader if you can avoid it. If you tried doing a full data migration with Dataloader, you will not be helped. By this I mean I will laugh at you and go back to drinking coffee. Dataloader is a BAD tool.

Amaxa is awesome and handles objects that are related to one another. It's free and awesome.

Jitterbit is like Dataloader but better. It's free. It's getting old though, and some of the newer stuff won't work like Time fields.

Talend requires some tinkering but knowing it will allow you to migrate from almost anything, to almost anything.
Hell you can even use SFDX to do data migrations.

But yeah don't use dataloader. Even Dataloader.io is better, and that's a paid solution. Yes I would recommend you literally pay rather than use Dataloader.

If you MUST use dataloader, EXPORT THE MAPPINGS YOU ARE DOING. You can find how to do so in the data loader user guide: https://developer.salesforce.com/docs/atlas.en-us.dataLoader.meta/dataLoader/data_loader.htm

> Even if you think you will do a data load only once, the reality is you will do it multiple times. Plus, for documentation, having the mapping file is best practice anyway. Always export the mapping, or make sure it is reusable without rebuilding it, whatever the tool you use.

Volume

If you are loading a big amount of data or the org is mature, read this document entirely before doing anything. LDV starts at a few million records in general, or several gigabytes of data. Even if you don't need this right now, reading it should be best practice in general.

Yes, read the whole thing. The success of the project depends on it, and the document is quite short.

Deletions

> If you delete data in prod without a backup, this is bad.
> If the data backup was not checked, this is bad.
> If you did not check automations before deleting, this is also bad.

Seriously, before deleting ANYTHING, EVER:

- get backup
- check automations
- check backup is valid.

Data Mapping

For Admins or Consultants: you should avoid mapping the data yourself. Any data mapping you do should be with someone from the end-user's who can understand you are saying. If no one like this is available, spend time with a business operative so you can do the mapping and make them sign off on it.

The client signing off on the mapping is drastically important, as this will impact the success of the data load, AND what happens if you do not successfully load it - or if the client realizes they forgot something.

Basic operations for a data mapping are as follow:

- study Source and target data model
- establish mapping from table to table, field to field, or both if necessary.
- for each table and field, establish Source of Truth, meaning which data should take priority if conflicts exist
- establish an ExternalId from all systems to ensure data mapping is correct
- define which users can see what data. Update permissions if needed.

Data retrieval

Data needs to be extracted from source system. This can be via API, an ETL, a simple CSV extract, etc. Note that in general it is better if storing data as CSV can be avoided - ideally you should do a point-to-point load which simply transforms the data - but as most clients can only extract csv, the following best practices apply:

- Verify Data Format
  - Date format yyyy-mm-dd
  - DateTime format yyyy-mm-ddT00:00:00z
  - Emails not longer than 80 char
  - Text containing carriage returns is qualified by "
  - Other field-specific verifications re. length and separators for text, numbers, etc.
- Verify Table integrity
  - Check that all tables have basic data for records:
    - LastName, Account for Contact
    - Name for Account
    - Any other system mandatory fields
  - Check that all records have the agreed-upon external Ids
- Verify Parsing
- Do a dummy load to ensure that the full data extracted can be mapped and parsed by the selected automation tool

Data Matching

You should already have created External Ids on every table, if you are upserting data.
If not, do so now.
DO NOT match the data in excel.

Yes, INDEX(MATCH()) is a beautiful tool. No, no one wants you to spend hours doing that when you could be doing other stuff, like drinking a cold beer.

> If you're using VLOOKUP() in Excel, stop. Read up on how to use INDEX(MATCH()). You will save time, the results will be better, and you will thank yourself later. Only thing to remember is to always add "0" as a third parameter to "MATCH" so it forces exact results.

Store IDs of the external system in the target tables, in the ExternalId field. Then use that when recreating lookup relationships to find the records.

This saves time, avoids you doing a wrong matching, and best of all, if the source data changes, you can just run the data load operation again on the new file, without spending hours matching IDs.

3 - Data Migration Step-by-step - Loading

FIRST STEPS

1. Login to Prod. Is there a weekly backup running, encoded as UTF-8, in Setup > Data Export
   - Nope
     Select encoding UTF-8 and click "Export Now". This will take hours.
     Turn that weekly stuff on.
     Make sure the client KNOWS it's on.
     Make sure they have a strategy for downloading the ZIP file that is generated by the extract weekly.
   - Yup
     - Is it UTF-8 and has run in the last 48 hours ?
       - Yup
         Confer with the client to see if additional backup files are needed.
         Otherwise, you're good.
       - Nope
         If the export isn't UTF-8, it's worthless.
         If it's more than 48h old, confer with the client to see if additional backup files are needed. In all cases, you should consider doing a new, manual export.

         > SERIOUSLY MAKE SURE YOU CHANGE THE ENCODING. Salesforce has some dumb rule of not defaulting to UTF-8. YOU NEED UTF-8. Accents and ḍîá□ řĩ♭î□ s exist. Turns out people like accents and non-roman alphabets, who knew?

     - If Data Export is not an option because it has run too recently, or because the encoding was wrong, you can also do your export by using whatever too you want to Query all the relevant tables. Remember to set UTF-8 as the encoding on both export and import.
2. Check the org code and automation
   - Seriously, look over all triggers that can fire when you upload the data.
     You don't want to be that consultant that sent a notification email to 50000 people.
     Just check the triggers, WFs, PBs, and see what they do.
     If you can't read triggers, ask a dev to help you.
     Yes, Check the Workflows and Process Builders too. They can send Emails as well.
   - Check Process Builders again. Are there a lot that are firing on an object you are loading ? Make note of that for later, you may have to deactivate them.
3. Check data volume.

- Is there enough space in the org to accommodate the extra data ? (this should be pre-project checks, but check it again)
- Are volumes to load enough to cause problems API-call wise ?
  If so, you may need to consider using the BULK jobs instead of normal operations
- In case data volumes are REALLY big, you will need to abide by LDV (large data volume) best practices, including not doing upserts, defering sharing calculations, and grouping records by Parent record and owner before uploading. Full list of these is available in the pdf linked above and here.

PREPARING THE JOBS

Before creating a job, ask yourself which job type is best.

Upsert is great but is very resource intensive, and is more prone to RECORD_LOCK than other operation types. It also takes longer to complete.
Maybe think about using the BULK Api.
In all cases, study what operation you do and make sure it is the right one.
Once that is done...

You are able to create insert, upsert, query and deletion jobs, and change select parts of it. That's because you are using a real data loading tool.

This is important because this means you can:

- Create a new Sandbox
- In whatever tool you're using, create the operations you will do, and name them so you know in which order you need to trigger them.
- Prepare each job, point them to a sandbox.
- Do a dummy load in sandbox. Make sure to set the start line to something near the end so you don't clog the sandbox up with all the data.
- Make sure everything looks fine.

If something fails, you correct the TRANSFORMATION, not the file, except in cases where it would be prohibitively long to do so. Meaning if you have to redo the load, you can run the same scripts you did before to have a nice CSV to upload.

GETTING READY TO DO THAT DATA OPERATION

> This may sound stupid but warn your client, the PM, the end users that you're doing a data load. There's nothing worse than losing data or seeing stuff change without knowing why. Make sure key stakeholders are aware of the operation, the start time, and the estimated

> end time. Plus, you need them to check the data afterwards to ensure it's fine.

You've got backups of every single table in the Production org.
Even if you KNOW you do, you open the backups and check they are not corrupt or unreadable.
Untested backups are no backups.
You know what all automations are going to do if you leave them on.
You talked with the client about possible impacts, and the client is ready to check the data after you finish your operations.
You set up, with the client, a timeframe in which to do the data operation.
If the data operation impacts tables that users work on normally, you freeze all those users during that timeframe.

Remember to deactivate any PB, WF, APEX that can impact the migration. You didn't study them just to forget them.

If this is an LDV job, take into account any considerations listed above.

DATA OPERATION

1. Go to your tool and edit the Sandbox jobs.
2. Edit the job Login to point to production
3. Save all the jobs.
4. You run, in order, the jobs you prepared.

When the number of failures is low enough, study the failure files, take any corrective action necessary, then use those files as a new source for a new data load operation.

Continue this loop until the number of rejects is tolerable.

This will ensure that if some reason you need to redo the entire operation, you can take the same steps in a much easier fashion.

Once you are done, take the failure files, study them, and prepare a recap email detailing failures and why they failed. It's their data, they have a right to know.

POST-MIGRATION

- Make sure everything looks fine, that you carried everything over.
- Warn their PM that the migration is done and request testing from their side.
- If you deactivated Workflows or PBs or something so the migration passes, ACTIVATE THEM BACK AGAIN.
- Unfreeze users if needed.

>

> Go drink champagne.

IF SHIT DOESN'T LOOK RIGHT

> You have a backup. Don't panic.

- Identify WTF is causing data to be wrong.
- Fix that.
- Get your backup, restore data to where it was before the fuckup. Ideally, only restore affected fields. If needed, restore everything.
- Redo the data load if needed.