Chapter 8

Stage 4: Analytics Coding to Maintain Data Provenance

8.1 GUERRILLA ANALYTICS WORKFLOW

As in the previous chapter, we are still at the analytics workflow stage of writing program code to do data analytics. This is stage 4 in Figure 22. Data has been sourced, extracted, and loaded. You are now concerned with writing analytics code that preserves data provenance.

8.2 EXAMPLES

You can manipulate data in an almost infinite number of ways. Here are some typical examples encountered in an analytics project.

- **Casting:** Gather every dataset in a system extract and cast all its fields into a correct data type such as date, integer, or text.
- **De-duplicating:** Run through a large dataset of vendor details, identify duplicate vendors by name, and tag those duplicates in the vendor dataset.
- **Joining:** Take a year's product sales data and join customer information to it to produce an analytics dataset of what every customer purchased, and when they purchased it.
- **Filtering:** Step through a dataset of web log entries and remove all entries that occur before 8 am and are associated with the user id "SYSTEM."
- **Enriching:** Take a set of instant message conversations and enrich them by identifying entities such as people names, business names, and place names.
- **Deriving:** Add a new field to a dataset that is the sum of two fields that already exist in the dataset.
- **Cleaning:** Step through a set of names and remove any formatting characters such as apostrophes and quotes, remove multiple adjacent spaces, and convert the name to upper case.

You can probably think of further examples. In all cases, data is being changed. Depending on how this change is managed, data provenance can be maintained or lost.

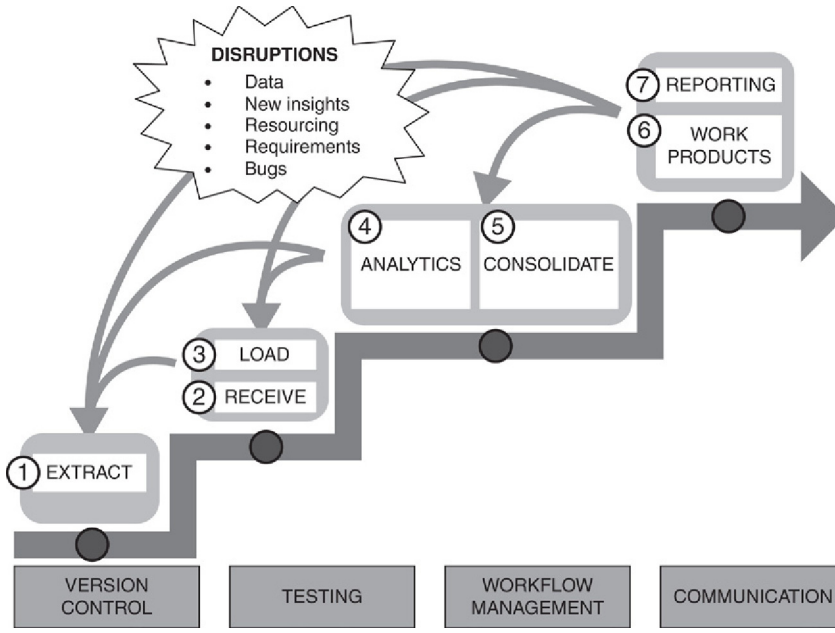

FIGURE 22 The Guerrilla Analytics workflow

8.3 PITFALLS AND RISKS

Since program code modifies data, there are many opportunities to make it difficult to maintain data provenance. In the worst case, you might break data provenance entirely. Here are examples of the pitfalls you will encounter.

- **Overwrites:** When cleaning a data field, the raw data is overwritten with the new clean data so the effect of the cleaning routine on that field can no longer be established.
- **Failure to distinguish derived data from original data:** Without some type of convention, it becomes difficult to know what data fields are derived by the analytics team, as opposed to being present in the original raw data.
- **Discarded records:** Data records can be directly deleted from data, perhaps because they are duplicates or because a business rule dictates their removal. However, there is then no way to assess the impact of the removal of these records, and no way to profile and test the removed records.
- **Difficulty in testing:** If modifications to data are not broken down into manageable data steps with intermediate datasets at the correct points in the data flow, then it becomes difficult to test that business rules were correctly applied to the data.
- **Loss of data record provenance:** If some type of record identifier is not carried through the data, it becomes difficult to trace a record back through the data flow to see where it came from and how it has been modified.

As always, a few simple Guerrilla Analytics practices and conventions go a long way toward mitigating these risks.

8.4 PRACTICE TIP 28: CLEAN DATA AT A MINIMUM OF LOCATIONS IN A DATA FLOW

8.4.1 Guerrilla Analytics Environment

In complex analytics data flows, there are numerous locations at which data cleaning can be performed. Regardless of whether you clean data early or late in the data flow, you should minimize the number of locations in which you do it.

War Story 9: You say tomato, I say TOMATO

Andrea was cleaning address data for mailing customers of Longwood Associates. The customer information came from several data sources, each with its own different format. A mistake was noticed in an output from the analytics team – a last name was inconsistently formatted and this was tracked back to Andrea's work. Some names were all upper case like "MAIN STREET" and some were all lower case like "main street." This was important both for presentation of the data, and for some of the data matching done earlier in the work product's data flow.

Andrea quickly introduced name cleaning in an early code file related to the offending dataset, making all its outputs upper case. After rerunning the data flow, there were still name inconsistencies present. Andrea chased down each of the offending data sources and added cleaning code to them separately, so that all their outputs were now upper case. Later in the project, a join in the middle of the data flow was not performing correctly because the join assumed lower case address details. Andrea was immediately on the case again (no pun intended). A quick cleaning step was introduced just before this join so that the data could be pushed through correctly as lower case.

After several of these ad-hoc types of changes, cleaning code had been scattered throughout the code base. It became very difficult for Andrea to know where particular data points were being modified. Effectively, data provenance was broken. The team paid the price in time wasted sifting through a code base scattered with ad-hoc cleaning and business rules.

If Andrea had been a Guerrilla Analyst, she would have identified one place in her data flow for cleaning. Early in the project, very little code might have been needed. As the project progressed and knowledge of the data improved, then she would have had that single data flow location for implementing additional cleaning rules. As new quirks in the data were discovered, these could be addressed in one place from which Andrea could report on the rules she had applied and assumptions she had made.

8.4.2 Guerrilla Analytics Approach

Whether data should be cleaned early in the data flow or later, is dependent on circumstances. Whichever decision is taken, ensure that data cleaning takes place in a minimum of locations in the data flow and is not scattered throughout the code.

8.4.3 Advantages

Performing cleaning in one location in the data flow makes cleaning routines easy to locate and maintain.

8.5 PRACTICE TIP 29: WHEN CLEANING A DATA FIELD, KEEP THE ORIGINAL RAW FIELD

8.5.1 Guerrilla Analytics Environment

Data always has quality issues or is not in a format that best suits an analysis. Data cleaning involves making changes to data to overcome these quality issues and to make the data amenable to analysis. Some examples include:

- Converting a data field to upper case for presentation purposes.
- Removing carriage returns and tab characters.
- Changing a coded system value into a meaningful business description. For example, "M" means "Male."

  Changing data in these ways affects data provenance.

8.5.2 Guerrilla Analytics Approach

When performing data cleaning, there are three things to keep in mind:

- **Copy:** Write the clean data field into its own new field rather than overwriting the original raw field.
- **Proximity:** Keep raw and clean fields close to one another so they can be easily compared.
- **Name:** Establish a naming convention so that cleaned data fields can be easily recognized and related back to their raw counterpart.

Figure 23 shows an example of a dataset with clearly identified clean and raw columns for the phone number.

ID	NAME	PHONE	PHONE_CLN
1023	Joseph	+44 1456 981	+441456981
1024	Jane	00 5673 97346	00 567397346
1025	Michael	1	1
1026	Stephen	9999999999	9999999999
1027	Ciara	01207 956 333	01207956333

FIGURE 23 A dataset with clean and raw data fields

8.5.3 Advantages

The advantage of this approach to data cleaning is that data provenance is preserved. Cleaned data can be easily identified and compared to its raw source.

Looking at the example dataset in Figure 23, you see that the incorrect phone numbers for Michael and Stephen are due to poor raw data, as opposed to any error in the cleaning rules the team has implemented.

8.6 PRACTICE TIP 30: FILTER DATA WITH FLAGS, NOT DELETIONS

8.6.1 Guerrilla Analytics Environment

Sometimes records need to be removed from a dataset. The records may not be relevant to an analysis, or they may be poor quality data that should not be part of the analysis. Filtering is the data manipulation step of removing data records. But if records are removed, you lose the profile of the original data and the ability to try out combinations of filters to assess their impact on the data.

8.6.2 Guerrilla Analytics Approach

Instead of deleting filtered records, flag these records for removal. This is like a simple switch. When the filter switch is on, the record is not included in an analysis. When the filter switch is off, the record can be included in an analysis.

8.6.3 Advantages

The advantages of the flag approach to data filtering are as follows:

- **Assess impact:** You can profile the data by the various filters to see the effect on the population of using a particular filter or combination of filters.
- **Keep your options open:** Since records are not deleted but are instead flagged for filtering, you can easily discard a filter without having to worry about recovering deleted data from earlier in the data flow.

There is a trade-off here between flexibility of filtering and efficiency of data storage. Keeping Guerrilla Analytics principles in mind, it is best to use flags early in a project when data are poorly understood. Later in the project, when data understanding stabilizes, and if storage space is still a concern, records can then be removed from data instead of using flags.

War Story 10: Beginning to Flag

Chris was working on a large tranche of data for Shook Inc., a music company specializing in rock and roll artists. The aim of the project was to apply a complicated set of business rules to Shook's book with the aim of designing new royalty structures. The first challenge was to whittle down the data by a

combination of rules that Chris was recommending from a clustering analysis, and that the customer needed to see for their own internal reasons. It was a big dataset of hundreds of artists' sales figures, record details, current contracts, and subgenres. This, in combination with Shook Inc.'s limited technology, meant that any rebuild of the data would take a couple of hours. Worse still, every time Chris would issue a segmentation, there would be disagreement and challenges from some of the client team. With even a small number of rules to consider, Chris was facing hundreds of potential rule combinations that may or may not have been acceptable to his client. This was wearing down Chris and frustrating his client.

Instead of taking a reactive approach, rebuilding data for every requested segmentation, Chris instead applied a set of flags to the dataset. Without wasting time on any filtering, Chris could now show the impact on the customer segmentations for any combination of flags. These candidate segmentations and their impact could be presented to Shook Inc. and Chris could quickly turn around new candidate rules that arose and were agreed. This meant Chris could remain agile in the face of changing requirements, respond to the project's growing data understanding, and explain the impact of various filter combinations without time-consuming data rebuilds. Rock "n" Roll Chris.

8.7 PRACTICE TIP 31: IDENTIFY FIELDS WITH METADATA

8.7.1 Guerrilla Analytics Environment

When cleaning data, you have already seen how it helps to keep the raw version of the data rather than overwrite it. However, it can still remain difficult in a dynamic environment to associate the raw data column with its clean counterpart.

Consider the example dataset in Figure 24. We cannot identify the raw and clean data fields because of a lack of consistency in naming fields. Is the PHONE field a raw field? Similarly the IN_SCOPE and ACTIVE? filters use different conventions. A query to switch on all filters would have to inspect and hard code every filter field from the data.

ID	CUSTOMER	ACTIVE?	PHONE	IN_SCOPE
205	Joseph	YES	+441456981	Y
206	Jane	NO	+567397346	Y
207	Michael	NO	1	N
208	Stephen	YES	9999999999	Y
209	Ciara	YES	01207956333	N

FIGURE 24 A dataset without any naming conventions

8.7.2 Guerrilla Analytics Approach

Imagine a typical office document such as a report. That report is a container of data. It contains sentences, paragraphs, appendices, tables, and figures. However, you can also use data to describe the document that contains this content. You can think of the whole document as having an author, a version, a publication date, and originating business department, for example. This "data about the data" is what is called metadata. What does this have to do with data manipulation?

If you name dataset fields according to a particular convention, then you can query and infer information from that field metadata.

8.7.3 Advantages

Figure 25 shows what the dataset now looks like after applying some simple conventions to field names and contents. Clean data field names are suffixed with "_CLN" and located beside their raw counterpart. Filter columns are prefixed with "IS_" and grouped together at the end of the dataset. This greatly improves clarity and data provenance.

In addition you can now do a lot with the data without specifying the exact structure of a dataset. For example, you could:

- Extract all clean fields from a dataset by looking for fields that have a name like "<something>_CLN."
- Switch on all filters by selecting data records where any fields called "IS_<something>" has a value of "YES."
- List all raw fields in a dataset by choosing fields that are not like "<something>_CLN" and not like "IS_<something>."

Simple conventions in clean field and filter field naming bring consistency to data fields, and also allow more powerful and flexible analytics that are not as dependent on a particular dataset structure.

ID	CUSTOMER	PHONE	PHONE_CLEAN	IS_IN_SCOPE	IS_ACTIVE
205	Joseph	+44 1456 981	+441456981	YES	YES
206	Jane	00 56 7397346	+567397346	YES	NO
207	Michael	1	1	NO	NO
208	Stephen	999 999 9999	9999999999	YES	YES
209	Ciara	01207956333	01207956333	NO	YES

FIGURE 25 A dataset with naming conventions

8.8 PRACTICE TIP 32: CREATE A UNIQUE IDENTIFIER FOR DATA RECORDS

8.8.1 Guerrilla Analytics Environment

In a data flow, raw data is manipulated by several data steps until an output is produced. In so far as possible, you would like to be able to trace individual data records through this data flow. This helps you debug and test program code, assess how each data record is being modified, and identify when data records are being dropped.

8.8.2 Guerrilla Analytics Approach

Creating a unique identifier (UID) for records helps maintain data provenance through a data flow. A good way to easily create a record UID is to apply a hash function to the full data record.

8.8.3 Advantages

Consider the simple data flow of joining two datasets as illustrated in Figure 26. The datasets have come from sources that did not have any primary key, so a UID has been added to both sources using a hash function. For any source record, it is simple to find where it ends up in the output of the data flow. The address dataset has a problem. One of its UIDs is repeated. When the datasets are joined on ADDR_ID, the resulting dataset has a repeated address for CUST ID 496. Using the address UID it is a simple matter to track down the offending record in the original address dataset. This is one simple example of how record UIDs are useful. The advantages of having a record UID are as follows:

- **Traceability:** If problems are encountered in a later dataset in a data flow, it is easy to trace problems back to the individual record at source without having to do complex dataset comparisons.

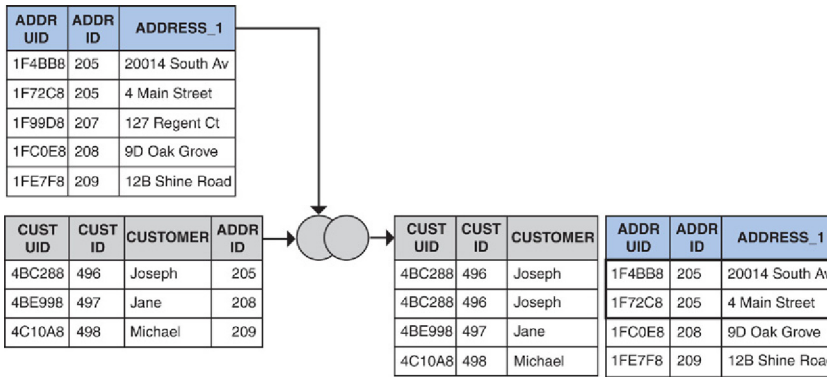

FIGURE 26 Data UID in a data flow

- **Finding dropped records:** If records are dropped from a join, it is simple to pull that population of dropped records out of the original source datasets by looking for missing UIDs.
- **Testability:** Much of testing becomes greatly simplified if there is a UID tracing a data record through a data flow. Testing will be discussed in detail in later chapters.
- **Duplicates:** If record UIDs are correctly calculated across an entire row of data, they are a quick way to compare and deduplicate rows without comparing every individual field.

8.9 PRACTICE TIP 33: RENAME DATA FIELDS WITH A FIELD MAPPING

8.9.1 Guerrilla Analytics Environment

In many scenarios, you are trying to bring together data from disparate systems. Because the data has different sources, it is unlikely to have the same format in each source. One system's CUSTOMER_ID field may be another system's CUST_ID field. Some fields will exist in only one of the systems and not the others. What is a flexible approach to appending data from disparate sources with different field names?

8.9.2 Guerrilla Analytics Approach

When different data sources must be appended together into a single data source, establish a data-field mapping to drive field renaming. A data-field mapping is simply a list of the field names in the disparate data sources, and the common destination field name in the unified output dataset.

Figure 27 shows an illustrative data mapping structure. The left "FROM" column is a listing of all raw field names that have to be renamed. The right "TO" column is a list of all field names that they get renamed to in the target dataset. If needed, a mapping dataset could also have columns to track sources of mappings, sign offs, and other important traceability information.

UID	FROM FIELD	TO FIELD
5GC298	CUSTOMER_ID	CUST_ID
3AE998	CUST_ID	CUST_ID
4C66A2	FNAME	FIRST_NAME
9H32B2	FIRST_NAME	FIRST_NAME

FIGURE 27 A field mapping dataset

8.9.3 Advantages

Taking a field mapping approach to field renaming has the following advantages.

- **Traceability:** The mapping dataset is a clear and traceable record of the business understanding of each field in the disparate sources. This can be important if there are misunderstandings later in the project.
- **Flexibility:** The mapping of fields exists in a separate dataset rather than being tightly embedded to program code. This makes changes to mappings more flexible. This is important as data understanding is evolving quickly.
- **Versioning:** If mappings change, versions of the mapping dataset can be maintained and compared with one another.

War Story 11: Field of Dreams/Nightmares

Sarah is a Guerrilla Analyst who has been drafted in to help Bean Enterprises with data integration and evaluation. Bean Enterprises sell healthy foods and snacks, and recently acquired another chain of similar stores called Lentil. As part of consolidating their product lines, Bean now want to look across their data store and Lentil's data store to get a single view of all products. The data stores are different. Bean uses a PRODUCT_IDENTIFIER field while Lentil have a PROD_ID field. Bean uses a CATEGORY field while Lentil uses something else. There isn't time or budget for a full data migration, so Sarah has to work quickly to get a reasonable view of the products across both data stores.

Her predecessor had made a start and began mapping each data field into a common field in program code. There are over 80 fields in 10 different tables, and no indication of who signed off any of the field mappings that have been completed so far. No wonder the analyst became overwhelmed and made a quick exit from this nightmare project.

Sarah's first step is to identify domain experts from Bean and Lentil who can help her understand the relevant product data fields from both data stores. She creates a big list of the in-scope fields from each data store and sets the domain experts to work agreeing where the fields are equivalent. This gives Sarah her data-field mapping. The data-field mapping is version controlled, and has clear sign off for every mapping. With even an early version of the mapping in place, Sarah can now apply it across all Bean and Lentil data and see what comes out. When the mapped field contents seem inconsistent, Sarah reports back to the domain experts and sets them investigating. When the mappings are good, Sarah can proceed with some early analytics for her customer.

After several weeks, all fields have a signed off mapping. Bean enterprises have benefited from incremental delivery of analytics instead of waiting months for a data migration exercise. In addition, they now have a valuable data asset should they wish to proceed with that data migration – a signed off mapping file for some of their critical data tables.

8.10 WRAP UP

This chapter has examined how to modify data in ways that preserve data provenance. In particular, you should now know the following.

- Code that changes data can cause loss of data provenance due to:
  - Overwrites of data.
  - Failure to distinguish derived data from original data.
  - Discarded records.
  - Difficulty in testing.
  - Loss of record provenance.
- Several simple practices can avoid these risks.
  - Clean data in a minimum of locations in a data-flow.
  - When cleaning data, keep a copy of the raw field.
  - Filter data with flags, not record deletions.
  - Identify clean fields and filter fields with metadata.
  - Create unique record identifiers.
  - Rename data fields with a data mapping rather than hard coding.