# The Inmon legacy problem

1, Firewatch Data Team publishes a cleansed dataset deduplicated on Longitude, Latitude, Discovery Date and Contained Date. The pipeline is published in the source-aligned Firewatch repo.

2, Marketing Data Team consumes the deduplicated table to create a map visualization.

3, HR Data Team finds the dataset but can’t use it because they want to analyze the manually entered duplicate data for a KPI. They make a change request to the Firewatch team.

4, At this point the Firewatch Data Team has three choices:

* Tell HR that the current version of the table is the “single version of truth” and they really don’t need that report anyways.
* Develop an alternative version of the table to serve HR needs.
* Change the existing table and break the Marketing report as a side effect.

5, They choose the second option and give the task to the Firewatch data engineer.

6, HR is satisfied at first but soon they realize, that they need additional columns and different filtering and contact the Firewatch data engineer to implement changes.

7, At this point the Firewatch Data Team is developing and maintaining two tables in their own repo that are owned by other teams. On top of that it is done by the Data Engineer of the Firewatch data team, who is not familiar with the HR and Marketing domains and serves other departments instead of focusing on Firewatch pipelines.

8, In the end both tables get broken because the Firewatch Data Engineer spots a bug and “fixes” both tables breaking the HR and Marketing reports as a side effect.

The problem here is that multiple, use-case specific versions of truth are published as public services. These are not owned by the teams that consume the data in a specific way, which is problematic because this way the data publishers are unaware which version of the truth is consumed by which consumer, and when they modify the public service, it breaks/bugs the consumer’s solutions without notion. Of course, many times it can happen that a single version of truth exists (not really because you probably want to pre-filter/select for low-latency and cost), but this case is better to be viewed as coincidence to have a generic pattern that can handle all the “multiple versions of truth” use-cases (just like decoupling over DRY in general).

Following this trail of thought it becomes evident that cleansing pipelines should be decoupled/decentralized from the EDW/Data Product to a Data Mart/consumer solution scope. In practice this means that cleansing pipelines belong in Data Marts and their respective repos.

# The Kimball legacy problem

* If an Entity is always mutable than should we treat an Open Sales Order as such even if it should be treated as an Event after it’s closed?
* And what about trigger-based Master data archives? Is a customer archive an Event?

I don’t think so….

* calling the mutable data “snapshot”
* and immutable data “archive”

is more generic.