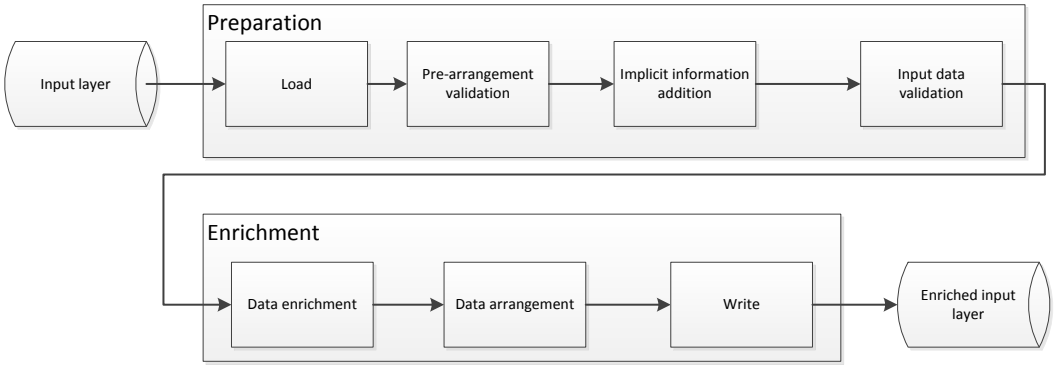
Annex III: Transformation process

Sub-phases of the transformation process

The common phases of the transformation process phases can be further broken down in sub-phases:



- Preparation phase:
  - **Load**: This sub-phase includes the technical transformations aimed to get the data from the input cubes.
  - **Pre-arrangement validation**: Includes the validations that have to be done before creating unions of cubes (those belonging to the entities “Loans” and “Protection received”). Concretely, the validations that check identifiers that have to be unique across more than one cube.
  - **Implicit information addition:** Includes the derivations that add variables that have an implicit value in the input. It also includes the technical transformations for merging all the loans cubes, and all the protection received cubes.
  - **Input data validation**: Includes all the remaining validations referred to the input layer information.
- Enrichment phase:
  - **Data enrichment**: Includes all the derivation rules that serve to create new information that is not implicit.
  - **Data arrangement**: Includes some technical transformations to arrange data into the enriched input layer model.
  - **Write**: Include technical transformations to write the resulting information into the enriched input layer cubes.

Dependencies of transformation schemes

Due to the fact that most of the transformation schemes depend on other schemes in the sense that they use datasets that are generated in other schemes we provide a graphical illustration of such dependencies for each transformation scheme following the header “Scheme dependencies” in the “Natural language” section. These dependencies can be computed from the transformation content in the database (i.e. the tables TRANSFORMATION_SCHEME, TRANSFORMATION, TRANSFORMATION_NODE).

Please note that we implemented some restrictions to the transformation schemes that are taken into account when computing these dependencies; first we do not include validation and put schemes in the dependency tree

and second, such a dependency tree does not contain any duplication of related schemes (although multiple schemes in such a tree may depend on the same transformation scheme).