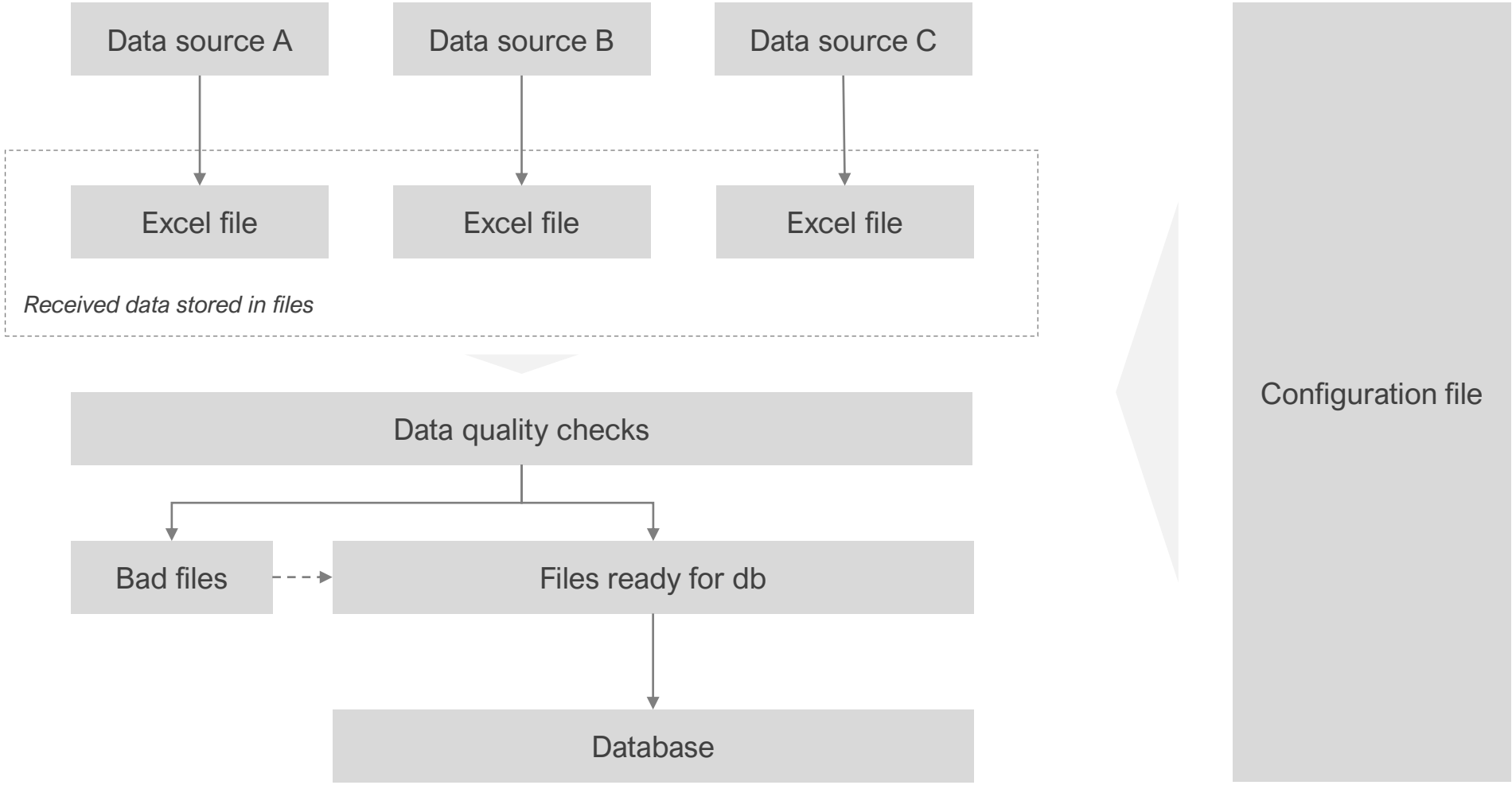
Data source processing

from request to DB

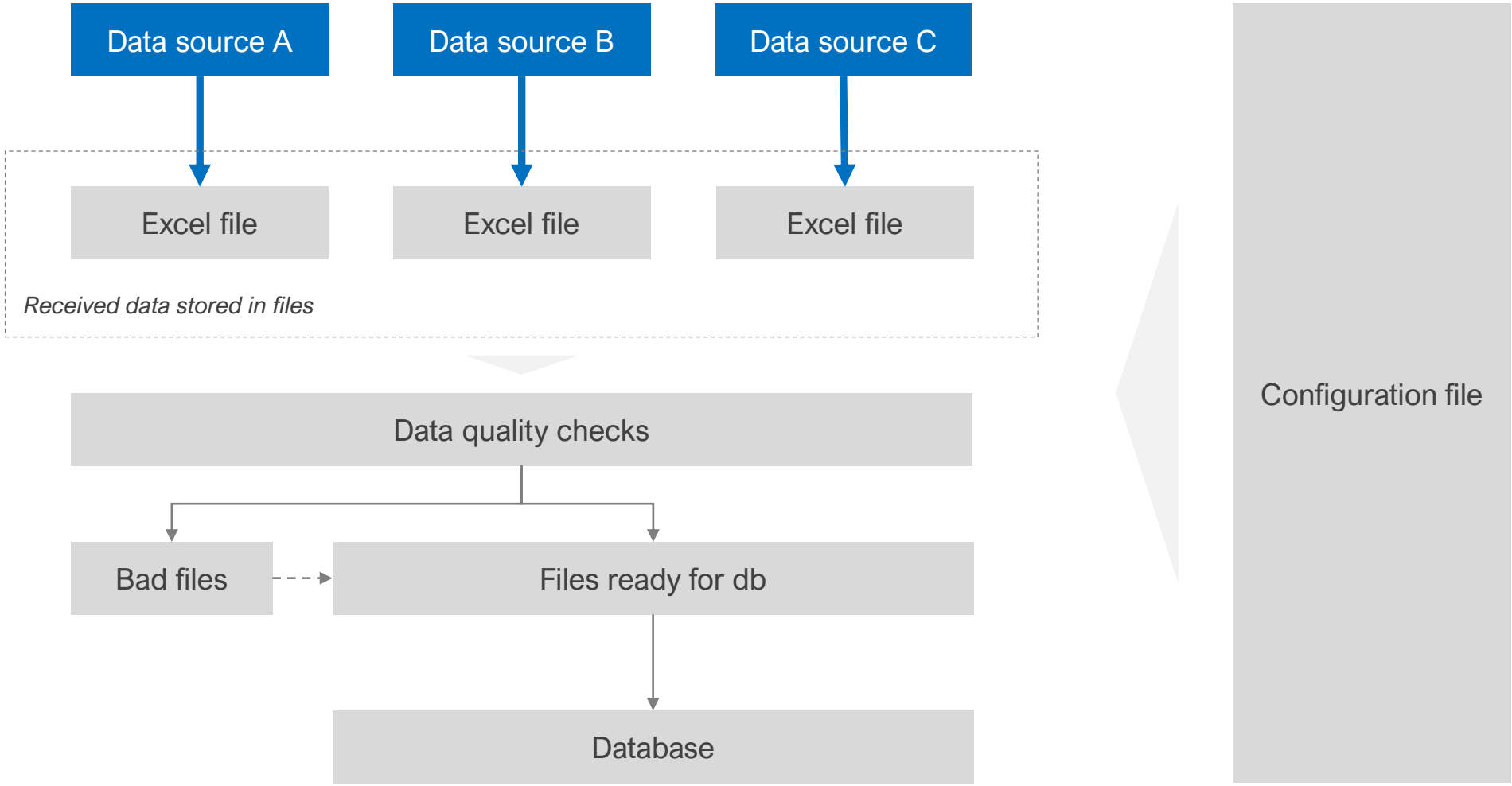
Table of contents:

Key building blocks of the solution



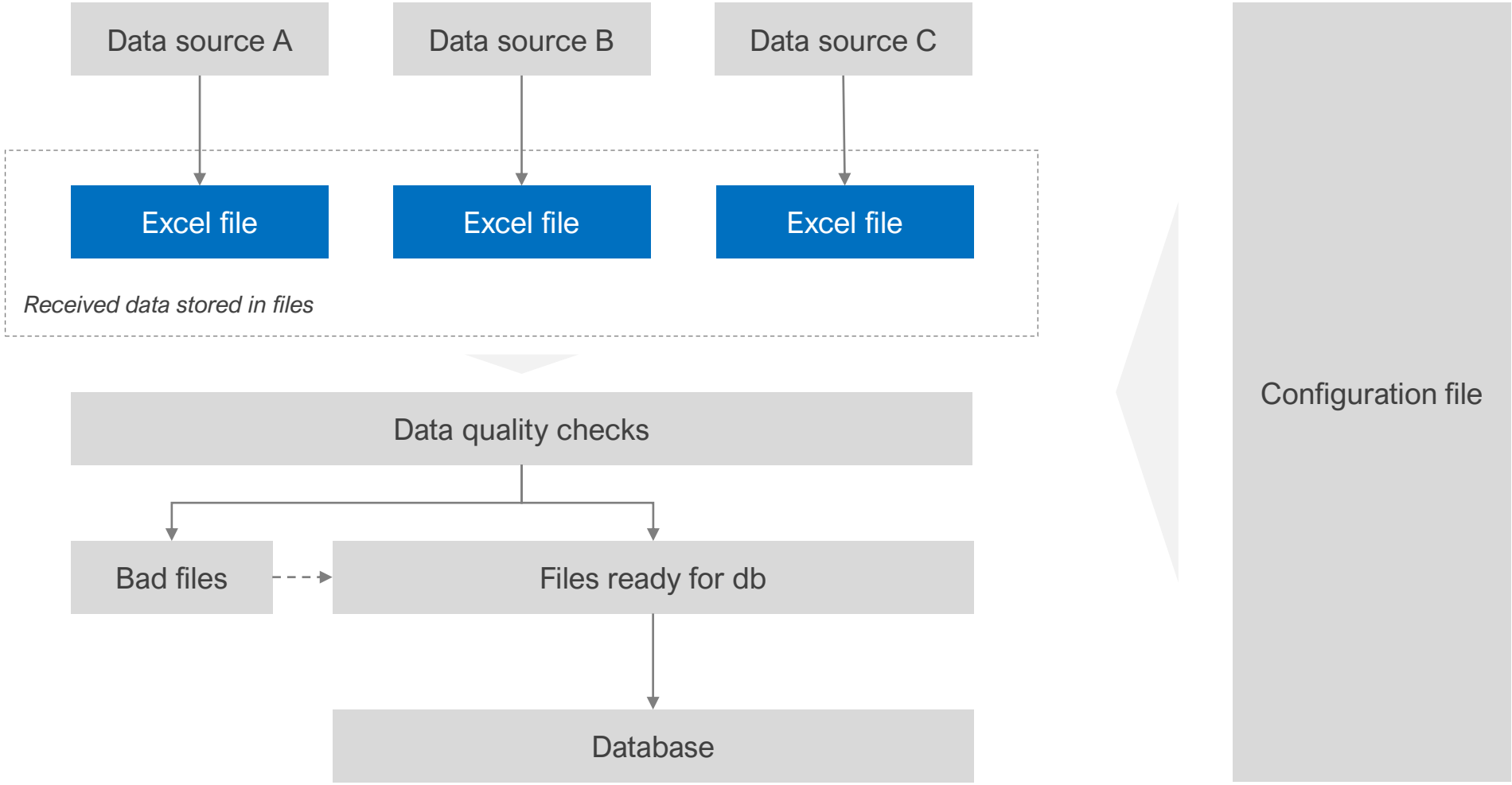
All processing logic of these building elements are managed through connection_processor.py and connection_config.xlsx

Key building blocks of the solution



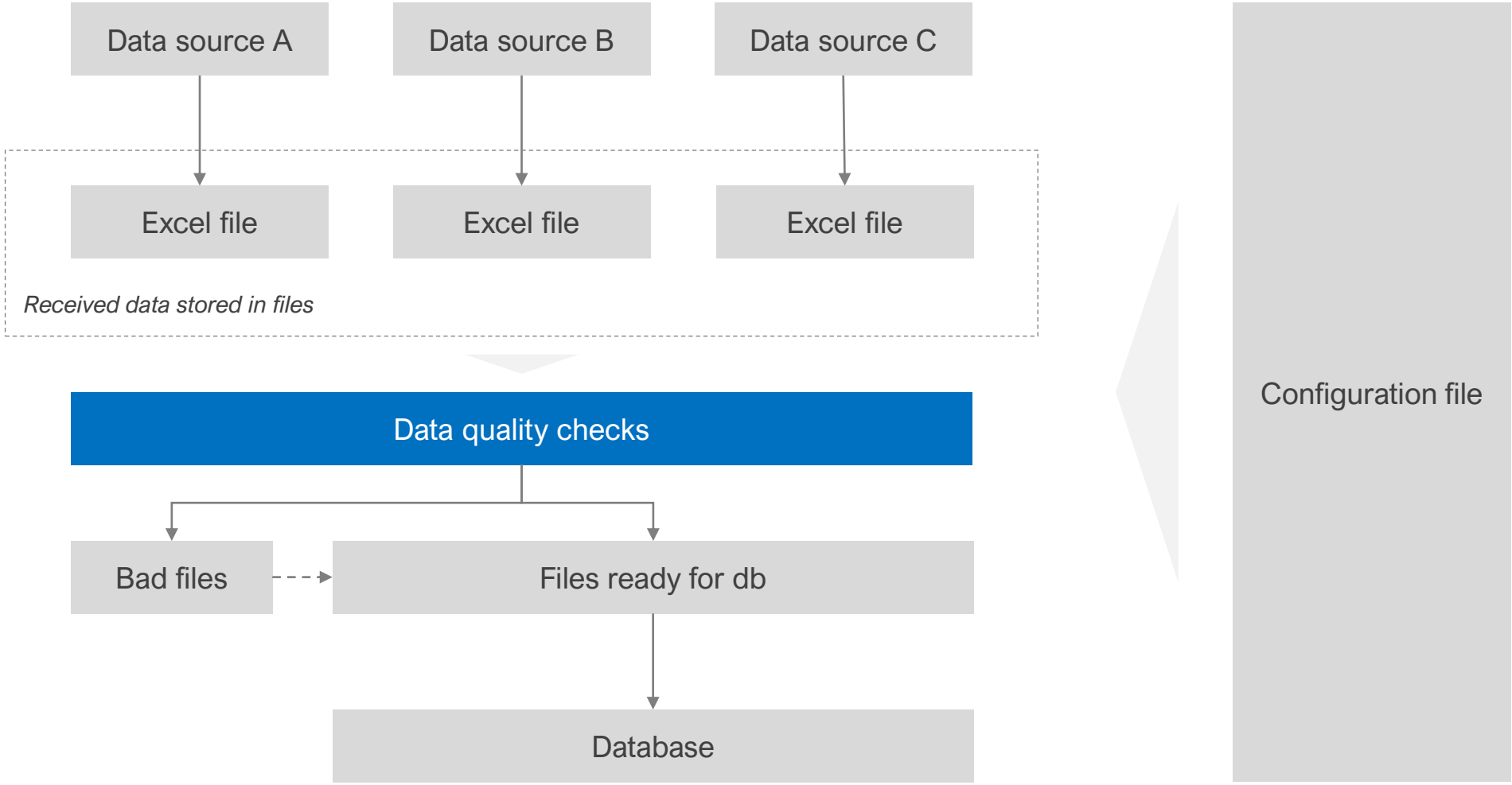
The current assumption is that each connection needs a fairly custom solution, so each data source needs a separate connection and processing logic. Current idea is that each connection has a python module for processing and a configuration file (sql table) which manages which connections and when need to be processed.

Key building blocks of the solution



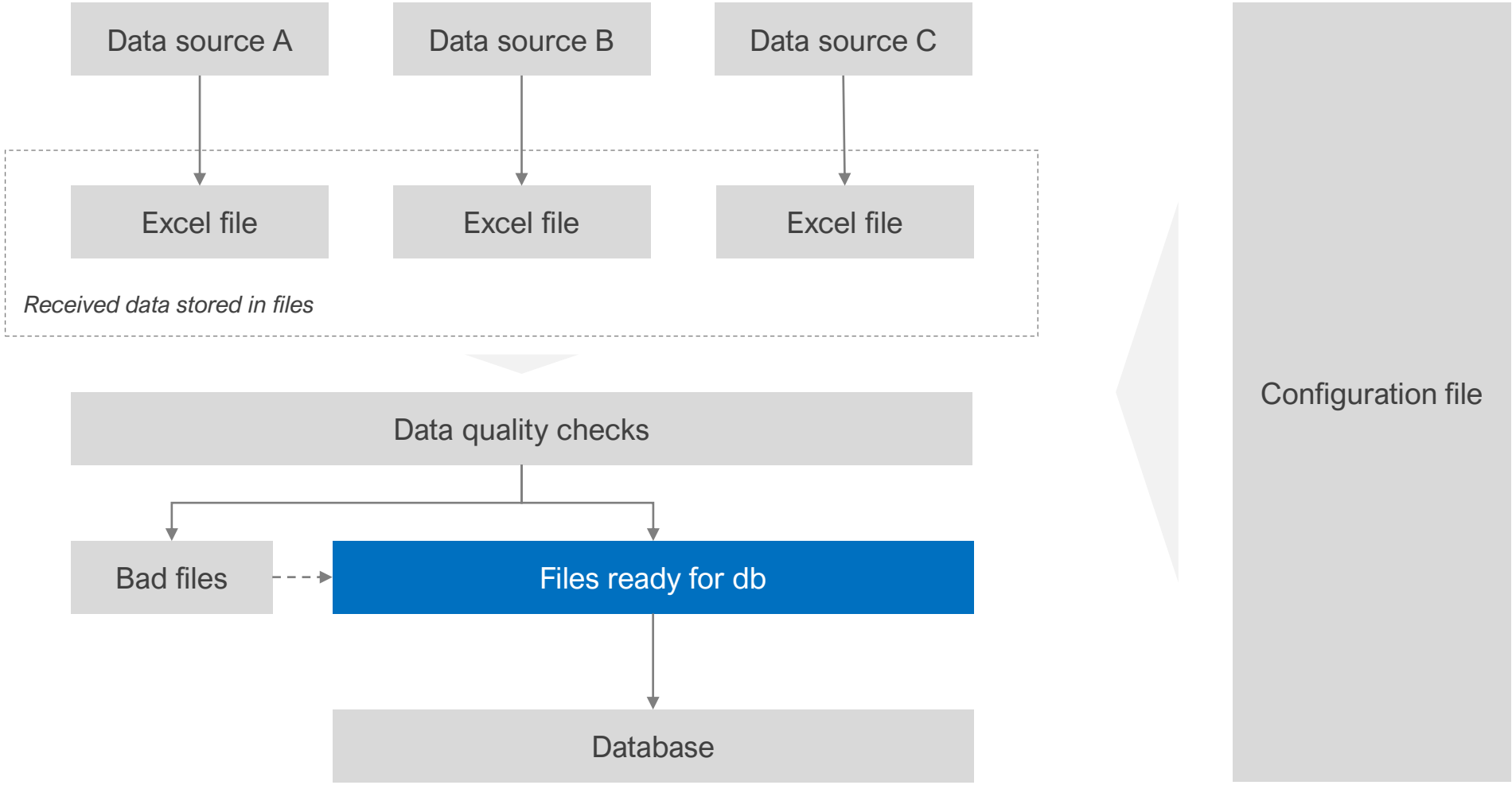
All data from connections are initially stored in excel files (depending on needs a text file might be a better option, for example, in some cases excel may need explicit definition how some data should be stored in a column). Excel files are stored in “received_files” folder for further processing.

Key building blocks of the solution



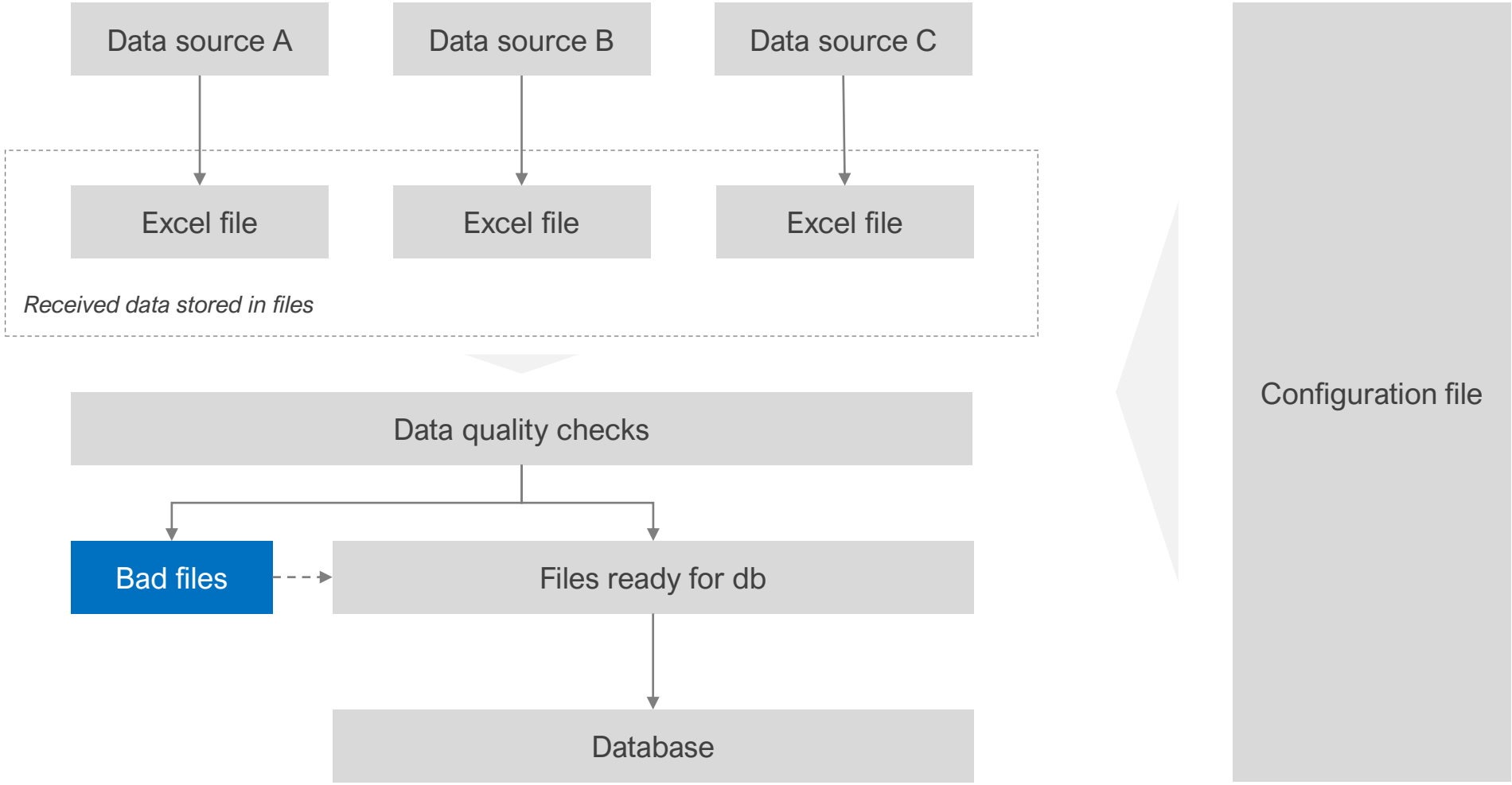
Files in "received_folder" are checked for any data quality issues (e.g. data completeness, formatting, datatypes, integrity etc.). A separate python module manages standardized controls. Through connection's configuration it is possible to manage standard control requirements – for example, skip standard controls.

Key building blocks of the solution



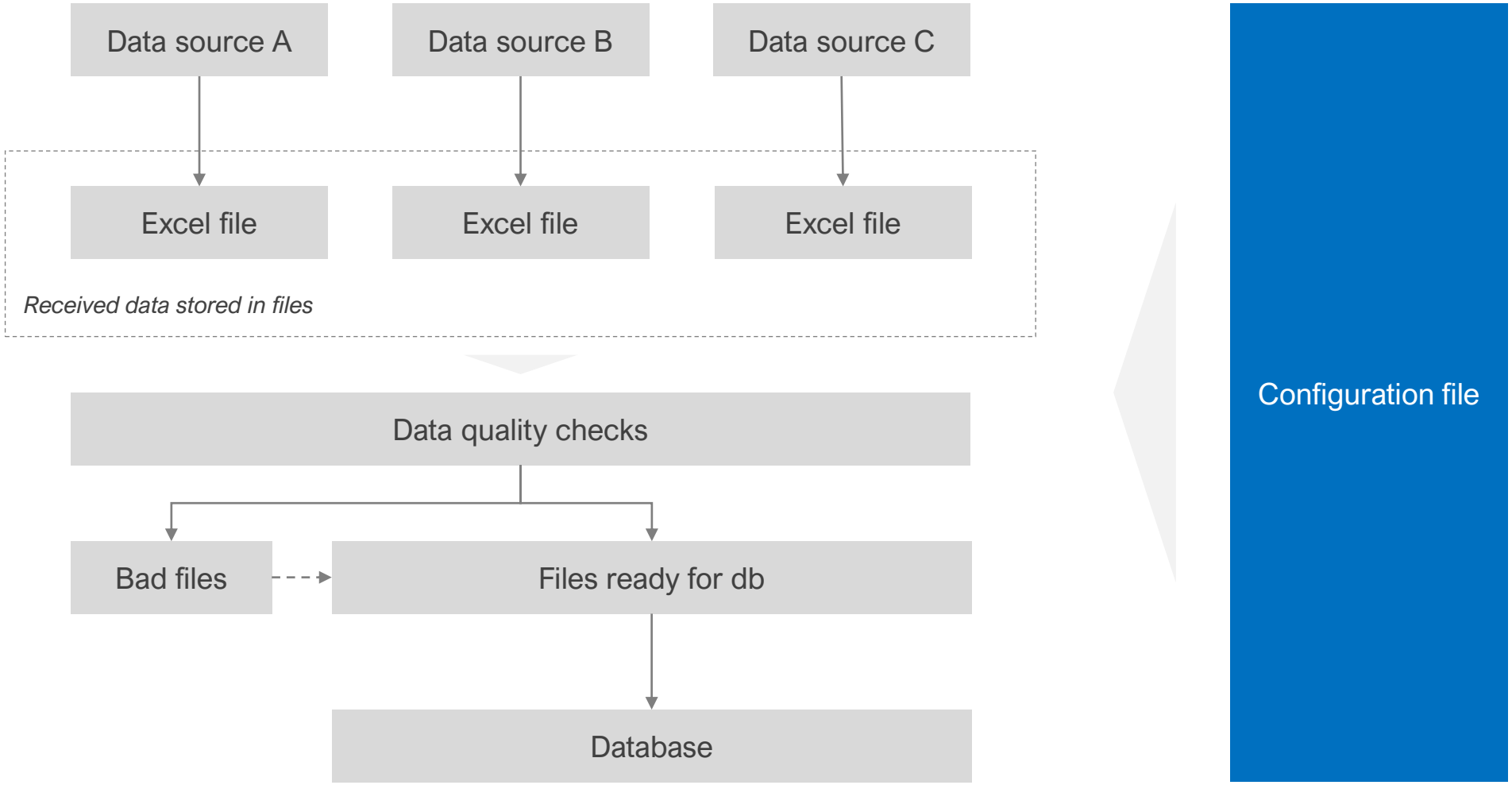
Files that have passed data quality checks are placed in “ready_for_db” folder and are uploaded to database.

Key building blocks of the solution



Files that have not passed data quality checks are placed in "bad_files" folder and will not be uploaded to database. It is expected that a manual user input is required – assess the data quality issues and make a decision if the file is acceptable and can be uploaded to database.

Key building blocks of the solution



Configuration file manages all connection and their processing parameters. For example, only connections that are marked as "active" are being processed. Currently the configuration file is an excel file, generally, it should be stored in an SQL table and a simple UI can be put on top of it to manage the configuration of connections and monitor their statuses (last updated, next update scheduled etc.)

Identify and describe how you would automate this process to update this data every year on 5th of March.

Current solution is limited to “initial data load” processing, because the connection module contains only “custom data” requests.
Here are the technical needs in order to facilitate data updates with a predefined frequency for a selected connection:
1. Implement field in configuration file which would store information on update frequency [yearly, quarterly, monthly., weekly, daily, hourly] and when the first update should be executed.
2. Based on these inputs a “next_update” timestamp fields in the configuration file would be returned.
3. The connection_processor.py would check if “next_update” value is equal or smaller to the current time. If that is true, the processing of connections’s module would start.
4. An update function (but not limited to – other functions that may be consumed by this update function may need development) would be developed in the connection’s module and called in connections_processor.py instead of the initial data load function.
5. After a successful data fetch, the “next_update” field would be updated to a new value.