Workflow Progress Control

This section explains how to manage the execution order of nodes within a workflow, including parallel processing and using control structures like loops and conditions.

Execution order

The execution order of nodes is determined by the data flow connections between their ports. The workflow progresses from an output port of one node to the input port of the next. However, you can customize this order for more complex scenarios.

Parallel execution

Megaladata can run multiple workflow branches at the same time without any extra configuration. If you connect nodes in parallel, their branches will execute concurrently, which can reduce the total processing time by making better use of system resources. For example, in the following workflow the `ABC#1`, `RFM #1`, and `XYZ #1` branches will run in parallel:

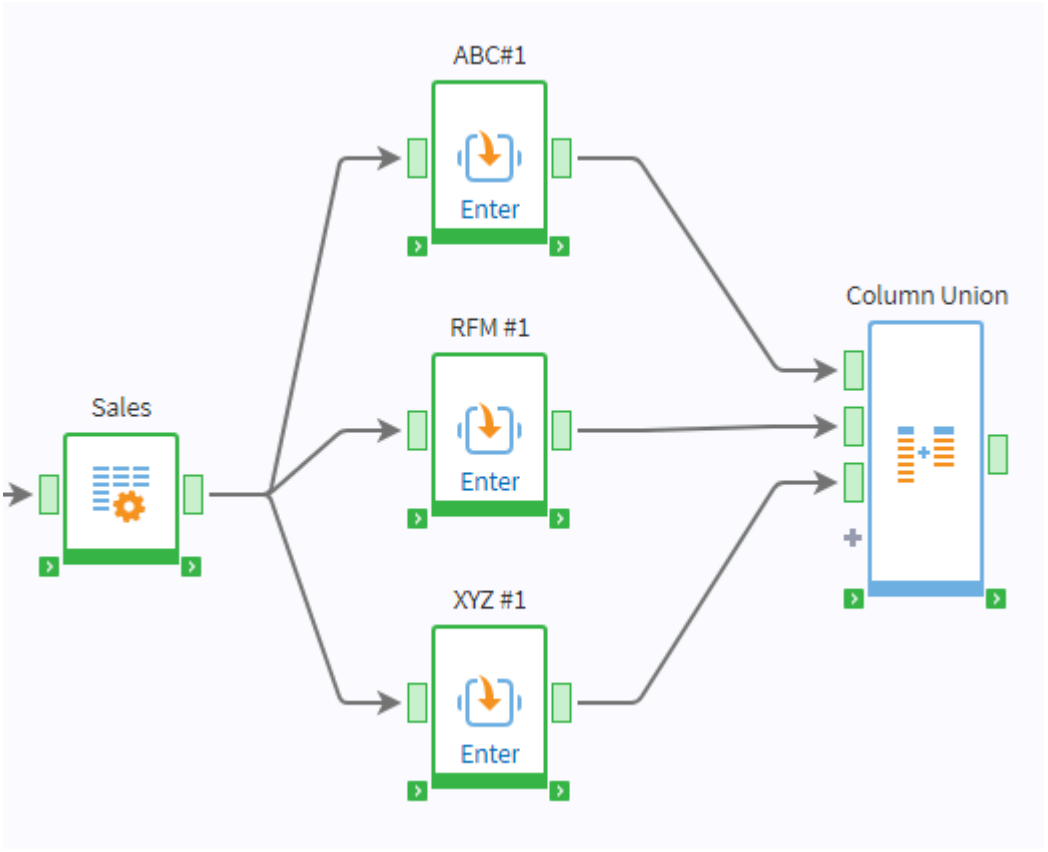


Figure 1. Concurrent execution of three workflow branches

Manual execution order setup

For situations where you need to control the sequence explicitly—for example, writing data to a database and then reading the modified data back, before proceeding with the workflow—you can use the **execution order port** (see service ports) It allows you to define a strict order that isn't based on data flow.

The workflow below shows how this port can force one database operation to complete before the next one begins.

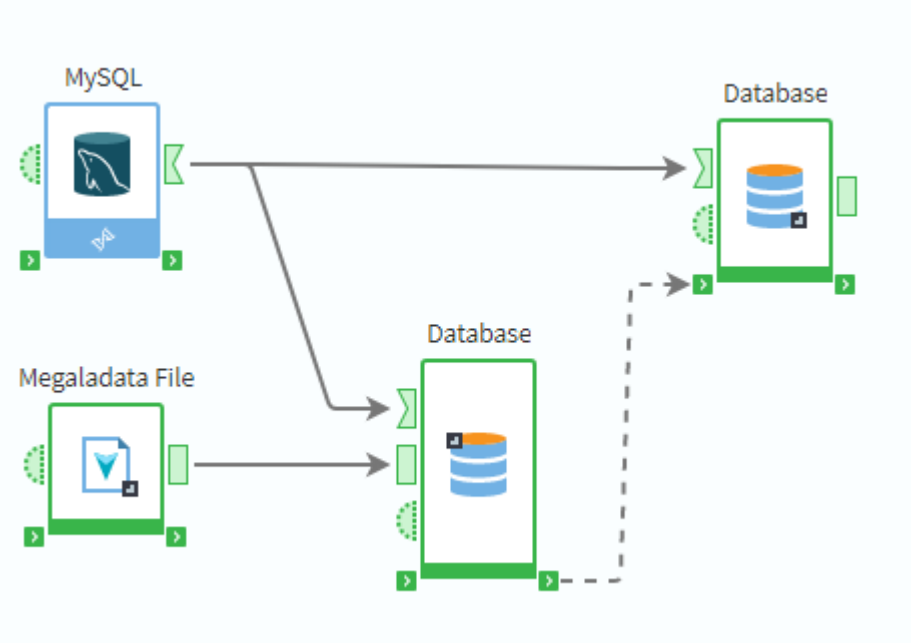


Figure 2. Using service ports to set the nodes execution order.

> **Note:** To see the service ports on the nodes, click the **Customize execution order** button on the workflow area toolbar.

Control structures

You can use control structures to create more advanced logic in your workflows, such as repeating a set of actions or choosing a specific execution path.

Loop

A Loop node repeatedly runs a set of instructions. This is useful for batch processing or iterative calculations. Megaladata supports several types of loops:

- **Counter loop**: Repeats a task a specific number of times (as in `FOR... TO...`).
- **Postcondition loop**: Repeats a task as long as a condition is true (as in `DO...WHILE...`).
- **Group processing**: Repeats a task for each entry in a dataset (as in `FOR EACH`).

You can create nested loops using **supernodes** and run loops in multiple threads to significantly reduce processing time.

Condition (Branching)

A Condition node directs the workflow down a specific path based on the value of an expression, similar to a `SWITCH...CASE` statement in programming. The workflow follows only the branch that matches the condition, as shown in the image:

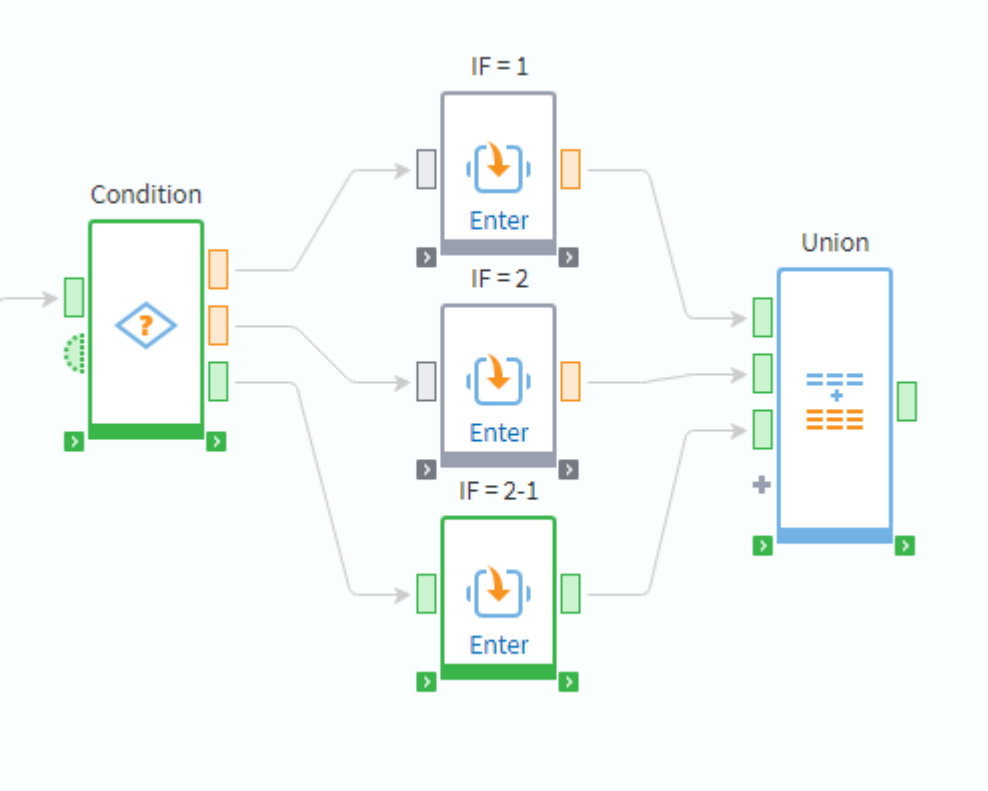


Figure 3. The workflow follows the branch that satisfies the condition

Read on: Node Activation Mode