DOI: 10.14529/jsfi210404

# Representation of Spatial Data Processing Pipelines Using Relational Database

***Igor G. Okladnikov***[1,2]



A methodology for representation of spatial data processing pipelines using relational database within the framework of the computing backend of the online information-analytical system "Climate" (`http://climate.scert.ru`) is proposed. Each pipeline is represented by a sequence of instructions for the computing backend describing how to run data processing modules and pass datasets between them (from the output of one module to the input of another one), including raw data and final computational results obtained in graphical or binary formats. Using relational database for storing descriptions of processing pipelines used in the "Climate" system provides flexibility and efficiency while adding and developing spatial data processing modules. It also provides computing pipelines scaling for further implementation for multiprocessor systems.

*Keywords: spatial data, information systems, databases, workflow, directed multigraph, processing pipeline, climate research.*

## Introduction

Usually, data analysis process represents a set of sequential operations starting from data search and retrieval and ending with the output of results in the required format. Depending on the complexity of the research method chosen, such a sequence might consist of three or more relatively simple computational procedures where intermediate results are passed from one to another. For instance, to calculate a quite simple climatic index "Monthly maximum of minimum daily temperatures", first it is necessary to read data from corresponding files (`https://www.climdex.org/learn/indices/#index-Tnx`), then to find the minimum daily values, then in the obtained values to find the monthly maximum value and finally to display and save the result, for example, in the graphical format. In the case when it is additionally necessary to calculate the trend of the index over several years one more operation is added. Practically in most studies such a procedure is either completely or partially a manual process when each such operation is performed by a researcher independently using various software products, starting from the very beginning every time. To automate this process specialized software products aimed at eliminating the need for regular routine actions and thereby speeding up the research might be used. One of such software products is the information-analytical system "Climate" [1] which allows solving problems of various range of complexity for the Earth science field, so that hiding complex technical and routine operations from the system user. However, even to be able to use the automation tools for computational processes, user (or developer) needs special skills and considerable time to form the necessary sequences of operations, namely, computing pipelines, for each separate type of data processing and analysis. Thus, an urgent task is to formalize the representation of computing (data processing) pipelines in a convenient and standardized form, that makes it possible to facilitate and force the process of their formation, modification, and reuse. This will contribute to the implementation of the current FAIR principles used for the management of scientific data and results (`https://www.go-fair.org/fair-principles/`), within the framework of any information-

[1]Institute of Monitoring of Climatic and Ecological Systems of the Siberian Branch of the Russian Academy of Sciences, Tomsk, Russian Federation
[2]Federal Research Center for Information and Computational Technologies, Tomsk, Russian Federation

analytical system. This paper describes a methodology for representing computing pipelines as modified labeled oriented multigraphs with their subsequent translation to relational database. The methodology is quite universal and might be adapted for other information and analytical systems.

## 1. General Approach

Within the framework of the digital information-analytical system "Climate", the developed computing modules providing unified programming interface are used for spatial data internal batch processing. A computing module is an isolated set of internal data structures and functions (class) that has an application programming interface (API) for input arguments (input data and control parameters), returning results (output data), and running module itself. A computational module can be represented as a function $f$, such that $Y = f(X)$, where $X$ is a vector of arguments, $Y$ is a vector of results, and each element of the vector is a multidimensional array of spatial data. A sequence of computing module calls where the results of one module are passed to the input of another one forms a computing pipeline. The pipeline described using one of the conventional technical formats (XML, JSON, etc.) is passed to the computing backend of the system which sequentially runs modules providing them with the necessary data and parameters. As a rule, the first module in the conveyor receives as an input one or more datasets, representing multidimensional arrays of climatic data obtained as a result of numerical modeling or observations. Subsequently, the datasets, representing intermediate processing results, are transferred along the pipeline from module to module. This procedure, aimed at performing a specialized processing of climatic data by "running" them through the computing pipeline, is known within the framework of the system by the name "processor". Each type of data processing (for example, the calculation of the specific climatic index) has its own processor. Parameters of the processor consist of the set parameters of the corresponding computing modules. Each parameter can take one of several valid values (or range of values). These can be threshold values of some measurable quantity (for example, the daily maximum temperature that must be greater than the threshold value of 25°C for calculating the climatic index "Number of summer days" and less than 0°C for calculating the index "Number of frost days"), or the choice of the one of the predetermined time period types for which an index is defined (day, month, year, etc.). Different values of the processor parameters affect the operation of its constituent computing modules and lead to different results. "Processor configuration" is a set of parameters of the computing pipeline that took one of the valid values. As part of the interaction with the "Climate" system the user selects a processor using a graphical interface and sets its parameters which are then passed to the computing modules thereby forming a specific configuration. Processor parameters have two types: variables (user-defined) and constants. The values of the variable parameters depend on the choice the user made using the interface. Constant parameter values do not change and are the inherent properties of each processor.

In the process of developing the digital information-analytical system "Climate", the first version of the metadata database was developed, built on the basis of the MySQL DBMS and designed to store information about the spatial datasets and processors available to the user [2]. As a result of the evolutionary development of the system it became obvious that the information about the processors in the metadata database should be extended using descriptions of the computing pipelines associated with these processors. The previously used approach to representing computing pipelines as separate XML files [3] proved to be inconvenient and unpromising from

the point of view of further system development. The adding of the computing pipeline descriptions to the metadata database provides flexibility and efficiency to the procedure of extending the computing backend functionality. It also provides the "client-server" model realization at the computing backend level, that, in turn, increases flexibility and reliability of its functioning within the framework of the distributed computing technology implemented in the "Climate" system.

To represent a computing pipeline within the scope of the relational database such as MySQL, initially it is convenient to display it as a graph reflecting the workflow [4]. Such a graph should describe not only the sequence of operations, but also the directions of the datasets transferred between them, thus separating the data flow and the control flow [5]. As it is shown in the number of related works, this task is solved by introducing additional properties to the graph (for instance, state vectors or "messages" of various types to pass the information between operations), labels of vertices and arcs, etc. [5–8]. Within the framework of the digital information-analytical system "Climate" to represent the computing pipeline in the form of a graph with its following translation to relational database, only the necessary additional constructions are introduced: labels of vertices and arcs. Using these, a specific computing module is assigned to each vertex, and dataset passed from one module to another is assigned to each arc. The dataset might represent processor parameters as well as intermediate or final results of calculations. Each arc has additional numeric labels that indicate the position of each dataset in the result data vector and in the vector of computing module arguments (thus defining the order of arguments and results for each dataset and computing module). Then, key-value pairs are associated with some vertices to specify the condition of their presence in the graph. It is generally accepted to use a labeled oriented multigraph [9] as a basis for representing such a workflow. The multigraph should be modified by adding extra labels for vertices and end labels for arcs.

The following is the methodology for representing the workflow of the data processing pipeline using modified labeled oriented multigraph as well as an information model that provides its implementation in the advanced version of the metadata database of the digital information-analytical system "Climate".

## 2. Methodology

### 2.1. Data Processing Pipeline Graph Representation

First, the workflow of an arbitrary computing pipeline might be presented as a simple labeled oriented multigraph $G\,(V, A, s, t, \Sigma_V, \Sigma_A, l_V, l_A)$, where $V$ is the set of the graph vertices; $A$ is a set of arcs connecting them; $s : A \to V$, $t : A \to V$ — two mappings that define the source and target vertices of an arc; $\Sigma_V$, $\Sigma_A$ are two sets of labels of vertices and arcs containing the names of computing modules and datasets, respectively; $l_V : V \to \Sigma_V$, $l_A : A \to \Sigma_A$ are two mappings that assign labels to vertices and arcs. The vertices of such a graph correspond to the associated computing module calls, and the arcs correspond to the operations of transferring control, datasets, and processor parameters between modules. Let us modify this graph so that it fully reflects the computing pipeline of the "Climate" system.

**Conditional vertices.** Let us assume that $K$ is a set of names of processor parameters, $W$ is a set of corresponding allowed parameter values and the mapping $Z : K \to W$ defines which values each parameter can take. Let us introduce the mapping $F : V^c \to Z$ that defines

for graph vertices belonging to the set $V^c \subset V$ the conditions in the form of "key-value" pairs which consist in the equality of the parameter $k \in K$ to the one of the allowed values $Z(k)$. The vertex $v^c \in V^c$ is suggested to be called "conditional", and the set $F(V^c)$ is the "set of graph conditions". A graph $G^G$ containing conditional vertices will be called "generalized". The generalized graph takes its final form only at runtime of the computing backend. Due to this fact, different computing pipelines can be formed from the same generalized graph "on the fly" both for different processors as well as for different configurations of the same processor depending on the user's choice. The graph $G^D \subseteq G^G$ that has taken its final form at runtime of the computing backend will be called "determined". An example of the generalized graph containing a conditional vertex (highlighted by a dotted line) is shown in Fig. 1.

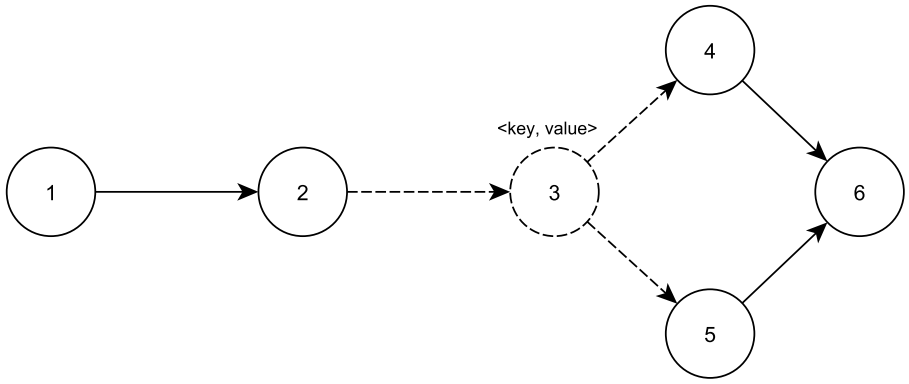


**Figure 1.** Generalized graph containing the conditional vertex 3. The condition of vertex existence in the processor current configuration is the presence of the parameter "key" with the value "value"

Let us introduce a conditional vertex deleting rules: 1) if a conditional vertex is removed from the generalized graph, then the incoming arc of the deleted vertex is redirected to the vertex to which the outgoing arc of the deleted vertex is directed; 2) if there are several incoming arcs, then all of them are redirected to the vertex to which the outgoing arc of the deleted vertex is directed; 3) if there are several outgoing arcs, then the incoming arc is redirected to each vertex where the outgoing arcs belonging to the removed vertex are directed; 4) if a vertex has several incoming and outgoing arcs, then such a vertex cannot be deleted and, therefore, cannot be conditional. Arcs corresponding to data transmission only are not considered and are removed along with the vertex.

If the condition associated with the conditional vertex is met for the processor configuration selected by the user, then at runtime the conditional vertex is preserved in the graph ($G^D \equiv G^G$) (see Fig. 2).

If the condition associated with the conditional vertex is not met for the processor configuration selected by the user (the configuration does not contain the corresponding parameter-value pair), the conditional vertex at runtime is removed from the graph according to the introduced vertex deleting rule thereby forming a subgraph $G^D \subseteq G^G$ (see Fig. 3).

Thanks to this technique different processors can use the same generalized graph, building on its basis the required computing pipeline using their own configuration.

**End labels of arcs.** The arcs $A$ of the labeled oriented multigraph G representing a computing pipeline of the "Climate" system correspond either to the operations of transferring control and datasets between the computing modules or only to the operations of transferring data that determine the configuration of the processor. The vertex of the graph arc comes out

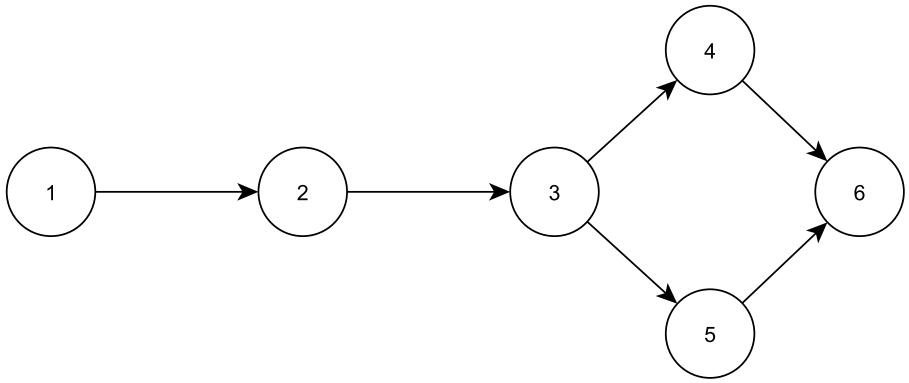


**Figure 2.** Graph with the vertex 3 when the condition is met

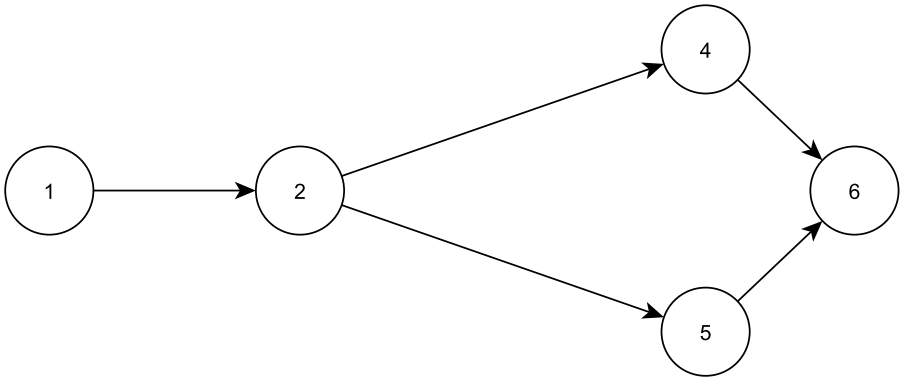


**Figure 3.** Subgraph when the vertex 3 is absent since the condition is not met

from will be called "source", and the vertex into which the arc enters – "target". The set $\Sigma_A$ contains special labels corresponding to different datasets or processor properties. If several datasets are passed from one module to another, multiple arcs with their own identification are introduced between the corresponding vertices of the graph, using one for each dataset. Since the order of the arguments and results is important to the module, each arc should be assigned two additional "end" labels: "output index" and "input index". When considering the dataset corresponding to an arc, the output index is its position in the vector of the source vertex results, and the input index is the position in the vector of the target vertex arguments. To connect arcs with these labels two mappings should be introduced $l_o : A \to N$, $l_i : A \to N$, which determine the source vertex output index and the target vertex input index of each arc, where N is the set of natural numbers. The generalized graph with added arc end labels is shown in Fig. 4.

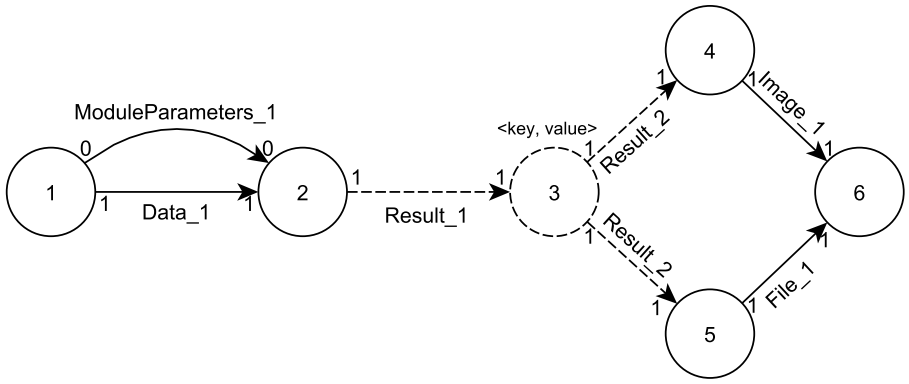


**Figure 4.** Processing pipeline graph containing conditional vertex and arc labels

Thus, the computing pipeline of the "Climate" system might be represented as a generalized modified labeled oriented multigraph $G^G\,(V, A, Z, F, s, t, \Sigma_V, \Sigma_A, l_V, l_A, l_o, l_i)$ that takes its final form (forming the required computing pipeline) only at runtime of the computing backend based on the processor configuration specified by the user interacting with the graphical interface. Additionally, it should be noted that due to the specifics of the task manager of the Climate platform, the presence of loops in graphs is not allowed.

## 2.2. Representation of the Computing Pipeline Graph in the Relational Database

To translate the graph of the computing pipeline to the relational database, an appropriate information model should be developed. The following entities will be required:

- graph vertex,
- computing module,
- graph arc,
- dataset,
- processor,
- processor configuration,
- parameter,
- parameter value,
- computing pipeline.

Several processors might be associated with a single computing pipeline. Each pipeline is represented as a graph consisting of vertices and arcs. Each vertex is associated with the corresponding computational module, and if the vertex is conditional, with a parameter-value pair (combination) specifying the condition. Each arc connects two vertices (source and target), the output index of the source vertex, the input index of the target vertex, and the dataset label associated with the arc. The computing module always has a unique name. The parameter also has a unique name, valid parameter values having unique values. The processor has a unique name and is associated with the processor configuration. The processor configuration is associated with pairs of parameter names and valid values. Each parameter can correspond to several valid values that it can take. And each valid value can be matched by several parameters that can take it. Datasets have unique conditional labels.

The conceptual diagram of the information model displaying a computing pipeline graph is as follows:

1. processor
   (a) processor ID (PK);
   (b) pipeline ID (FK).
2. computing pipeline
   (a) pipeline ID (PK);
   (b) pipeline description.
3. vertex
   (a) vertex ID (PK);
   (b) computing module ID (FK);
   (c) ID of combination specifying the condition (FK).
4. arc
   (a) arc ID (PK);

   (b) pipeline ID (FK);
   (c) source vertex ID (FK);
   (d) target vertex ID (FK);
   (e) source vertex output index;
   (f) target vertex input index;
   (g) dataset ID (FK).
5. computing module
   (a) computing module ID (PK);
   (b) computing module name.
6. parameter
   (a) parameter ID (PK);
   (b) parameter name.
7. parameter value
   (a) parameter value ID (PK);
   (b) parameter value name.
8. dataset
   (a) dataset ID (PK);
   (b) dataset label.
9. processor configuration
   (a) processor configuration ID (PK);
   (b) processor configuration description.
10. combination ("parameter-value" pair)
   (a) combination ID (PK);
   (b) parameter ID (FK);
   (c) parameter value ID (FK).
11. processor has processor configuration
   (a) processor ID (FK);
   (b) processor configuration ID (FK);
12. processor configuration contains a combination
   (a) processor configuration ID (FK);
   (b) combination ID (FK).

Here PK mean primary key attributes while FK mean foreign key attributes. The conceptual ER diagram using Crow's Foot notation [10] is presented in Fig. 5.

## 2.3. Building the Computing Pipeline Based on the Relational Database

To build a computing pipeline based on the information contained in the relational database, it is necessary to execute several SQL queries. The nature and specification of such queries depends on the specific DBMS and SQL language version, so they are not presented in this work. The result of the SQL queries execution is the information about the vertices and associated computing modules as well as the arcs that connect them. Having this information in mind, it is possible to restore a generalized computing pipeline graph in the computer's RAM using any conventional way (for instance, using an adjacency list by the method of Guido van Rossum, `https://www.python.org/doc/essays/graphs/`).

To be able to obtain a generalized graph specific implementation, it is required having a list of parameters selected by the user, to enumerate all the conditional vertices of the built generalized

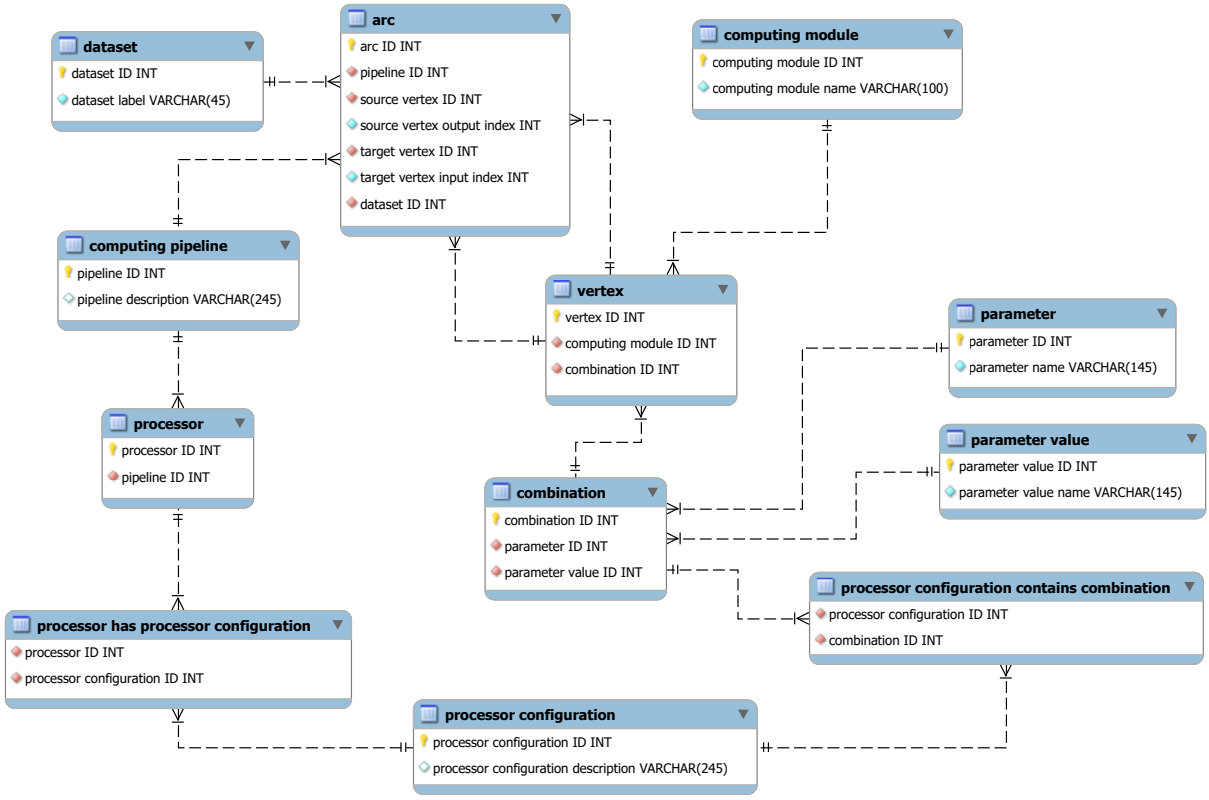


**Figure 5.** ER model representing a processing pipeline generalized graph

graph and then, according to the methodology presented in this work, to delete those that do not meet the conditions. The graph obtained represents a sequence of operations of the computing pipeline that meets the user's requirements. The sequence might be extracted, for example, by performing starting from the starting vertex a complete traversal of the graph using the breadth-first search method [11]. Then the operation sequence should be transformed into description of the computing pipeline with the following transfer to the corresponding computing unit of the system backend. A single pipeline can be associated with several processors thus providing the reuse of already developed pipelines. In this case the mechanism of conditional vertices allows, using various parameters of the processor, to obtain various specific implementations of computing pipelines. This reduces the time required to add new processors to the system and greatly simplifies the work of developers thus speeding up the process of bringing new features to the system users.

## 2.4. Climatic Index Example

Let us consider a simple case based on the example of building a computing pipeline for the "Monthly maximum of daily minimum temperatures" index (Fig. 6).

The figure depicts a generalized graph for a given data processing pipeline. The following constant vertices corresponding to the computing modules can be distinguished: cvcCalcMaximum, which basically calculates the index at each point of the geographic grid, and cvcOutput which outputs the results of the computations in the format required. The yellow dashed line outlines the conditional vertices cvcCalcTrendTM calculating the trend values, and cvcCalcTiMean which associated module calculates the average values of the index in the case the user selects more than one annual period for processing (then the index is calculated for the selected

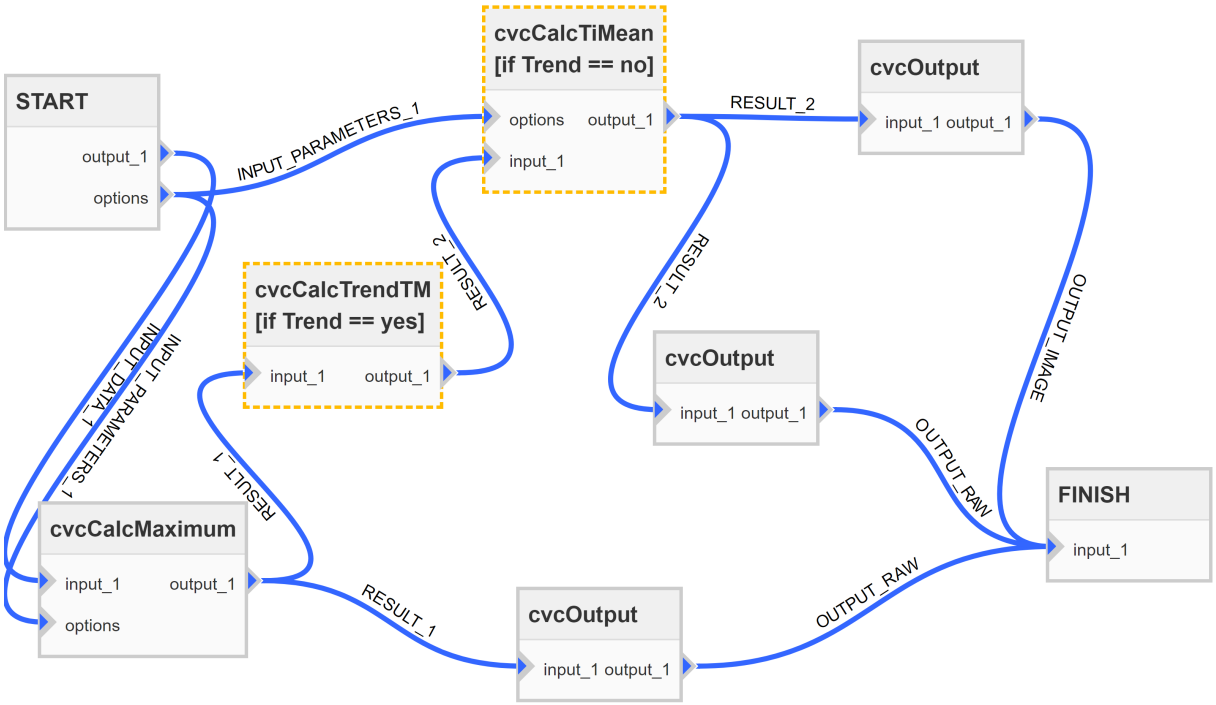


**Figure 6.** Processing pipeline of the climate index "Monthly maximum of daily minimum temperatures"

month of each year, for example, July). The preserving condition for the cvcCalcTrendTM vertex is the presence of the "Trend" parameter with the "yes" value (the user selected the trend calculation using the graphical interface), and for the cvcCalcTiMean vertex – the "Trend" parameter with the "no" value (the user did not select the trend calculation). Thus, a specific pipeline implementation contains only one of these conditional vertices. The INPUT_DATA_1 arc corresponds to the dataset being processed, for example, the ERA-Interim reanalysis [12], the INPUT_PARAMETERS_1 arc corresponds to the configuration parameters of the processor (computing modules), the RESULT_1 and RESULT_2 arcs correspond to the intermediate calculation results in the form of datasets, OUTPUT_IMAGE arc corresponds to the calculation result in the graphical format (GeoTIFF, Shapefile), and OUTPUT_RAW arcs correspond to the calculation results in the format of a multidimensional binary array of spatial data (NetCDF).

## Conclusion

The proposed methodology allows to effectively represent spatial data processing pipelines in the relational metadata database of the digital information-analytical system "Climate". Thanks to the "generalized" graph approach different processors can use the same graph. Based on the graph different computing pipelines are generated for different values of configuration parameters. Translating descriptions of generalized computing pipelines to relational database provides flexibility and efficiency in adding new and revising existing spatial data processing modules as well as provides computing pipelines scaling for further implementation for multiprocessor systems. The methodology is substantially universal and might be adapted for other information and analytical systems, as well as find application in other subject areas.

## Acknowledgements

The reported study was funded by the State project No. 121031300158-9.



## References

1. Gordov, E., Shiklomanov, A., Okladnikov, I., et al.: Development of Distributed Research Center for analysis of regional climatic and environmental changes. IOP Conference Series: Earth and Environmental Science 48, 012033 (2016). https://doi.org/10.1088/1755-1315/48/1/012033
2. Okladnikov, I.G., Gordov, E.P., Titov, A.G.: Development of climate data storage and processing model. IOP Conference Series: Earth and Environmental Science 48, 012030 (2016). https://doi.org/10.1088/1755-1315/48/1/012030
3. Okladnikov, I.G.: Computing core of a software package for "cloud" analysis of climate change and the environment. IOP Conference Series: Earth and Environmental Science 611, 012058 (2020). https://doi.org/10.1088/1755-1315/611/1/012058
4. Swamy, M.N.S., Thulasiraman, K.: Graphs, Networks, and Algorithms. Wiley-Blackwell, New York (1980)
5. Allamanis, M., Brockschmidt, M., Khademi, M.: Learning to represent programs with graphs. International Conference on Learning Representations (ICLR) (2018). https://openreview.net/pdf?id=BJOFETxR-
6. Chang, C.L.: Interpretation and execution of fuzzy programs. In: Zadeh, L.A., et al. (eds.) Fuzzy Sets and Their Applications to Cognitive and Decision Process, pp. 191–218. Academic Press, New York (1975)
7. Averkin, A.N., Batyrshin, I.Z., Blishun, A.F., et al.: Fuzzy sets in models of control and artificial intelligence. Nauka, Moscow (1986)
8. Li, Yu., Tarlow, D., Brockschmidt, M., Zemel, R.: Gated graph sequence neural networks. International Conference on Learning Representations (ICLR) (2015). https://arxiv.org/pdf/1511.05493.pdf
9. Balakrishnan, V.K.: Graph Theory. McGraw-Hill (1997)
10. Halpin, T.: Entity Relationship modeling from an ORM perspective: Part 1. Object Role Modeling. http://www.orm.net/pdf/JCM11.pdf, accessed: 2020-02-25
11. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill, Cambridge (2009)
12. Dee, D.P., Uppala, S.M., Simmons, A.J., et al.: The ERAInterim reanalysis: Configuration and performance of the data assimilation system. Quarterly Journal of the royal meteorological society 137(656), 553–597 (2011). https://doi.org/10.1002/qj.828