Tool Data Modeling Method Based on an Object Deputy Model

Qianwen Luo, Chen Chen, Song Wang, Rongrong Li, and Yuwei Peng(✉)

School of Computer Science, Wuhan University, Wuhan, Hubei, China
{qwluo17,chenchen33,xavierwang,rrli,ywpeng}@whu.edu.cn

Abstract. With the development of intelligent manufacturing industry, the management of tool data in machine tool processing is becoming more and more important. Due to the richness of machine tool data and the complexity of relationships in them, it's hard for a traditional relational database to manage the tool data. Therefore, a new method should be proposed to manage these data in a better way. In this work, we propose a tool data modeling method based on the object deputy model, which utilizes the characteristics of the class and the objects to express the meaning of the tool data and the various semantic constraints on them. Unlike the traditional relational model, objects are connected with a two-way pointer in the object deputy model where an object can have one or more deputy objects that inherit the properties and methods of the source object, and the deputy objects can have their own properties and methods. Besides, the two-way pointer between the source class and its deputy class makes the cross-class query easier in two aspects: One is to make complex queries expressed in intuitive statements, and the other is to improve query efficiency. We implemented and evaluated our model on an object deputy database. Experiments show that our method is better than the traditional relational ones.

Keywords: Object deputy model · Tool data · Data modeling · Database

1 Introduction

In traditional metal manufacturing, the reasonable use of various tools is of great significance to improve production efficiency, product quality, and process safety, and reduce costs. In today's large-scale use of NC machining, the use of tool data in the programming process is particularly important. Especially the selection of tools and the recommended cutting parameters of the tool when processing the workpieces are of decisive significance. In detail, the tool data has two main characteristics. Firstly, the tool data have rich meanings, including the geometric parameters of the tool, such as diameter, included Angle, cutting edge length, programming length, and material of the tool, such as tool material, tool coating material, and description information of the tool, i.e. tool type, scope of application. Secondly, the tool data contains complex relations, such as the machining

X. Wang et al. (Eds.): APWeb-WAIM 2020, LNCS 12318, pp. 491–499, 2020.
https://doi.org/10.1007/978-3-030-60290-1_37

of the tool, which needs to consider the parameters of the tool, the processing technology, the workpiece material, and the cutting speed of the tool. According to Sandvik's statistics, machine tool operators spend 20% of their time finding suitable tools in the manufacturing process which is a high percentage. Therefore, if the time required to find a suitable tool is reduced, the machining efficiency will be greatly improved. Advanced manufacturing countries attach great importance to tool data management and have successively developed their own tool management software, such as AutoTas tool management system developed by Sandvik of Sweden, TDM Systems developed by Walter Information System of Germany, TMS developed by Zoller of Germany, WinTool system developed by WinTool of Switzerland. The mature tool management system is still absent in China. The existing tool management systems are developed by different enterprises according to their own requirements. Consequently, they lack uniform standards. This situation is incompatible with China's status as the world's largest manufacturing country. Therefore, our work focused on the organization of tool data and the construction of the tool management model on the basis of industrial data processing and industrial Internet technology.

In view of the above data characteristics, all of the above-mentioned foreign tool management systems adopt the relational data model to represent tool entities and their relationships. Since the relational data model is simple and clear [1]. It has a solid mathematical theoretical foundation, and the operating language is descriptive, which greatly improves the efficiency of data storage and system development. However, there are still some shortcomings. We begin with a careful exploration of the tool data model and discuss the limitations of the relational data model concerning the characteristics of the tool data listed above. First, the semantic expression is foggy. In the tool processing scenario, the completion of the workpiece is related to many factors such as the workpiece material and processing use cases. No matter what kind of relationship between entities is connected by the join in the relational data model, the semantic expression is foggy. Second, it's hard for the relational data model to maintain data consistency. During the use of the tool, the life cycle of the tool is limited, so the tool will be updated frequently. Finally, the query efficiency of the relational model in our scenario is low. A large number of join operations are required to determine the information to be found. As we all know, the join operation takes a lot of time. Therefore, to address these problems, we propose a tool data modeling method based on the object deputy model [3]. The object deputy model [2] is a new data model based on the traditional object-oriented model, which enhances the flexibility and modeling ability of the object-oriented model. There are four kinds of relationship types in the object deputy model, namely select, group, union, and join, which make the semantic relation between tool data express accurately and abundantly. The select deputy class indicates whose deputy objects are selected from a class according to a select predicate. The join deputy class indicates that the deputy objects are derived from the results of several classes connected according to a combination predicate, so a join deputy object corresponds to a source object in each source class. The group deputy class

indicates whose deputy objects are derived from a class according to a grouping predicate, so a group deputy object corresponds to a set of source objects. The union deputy class indicates that the source objects of several classes of the same type are merged into one deputy class. When the attribute values of the tool are modified, our object deputy model provides the update propagation mechanism to maintain their consistency. In addition, the cross-class query supported by the object deputy model avoids a large number of join operations in the relational data model. In order to verify the effectiveness of our modeling method, we used the actual tool data to make a comparative experiment between the object deputy model and the relational model, the results of which show our data model is more efficient than the relational data model.

In summary, our contribution are as follows.

- We propose a semantic analysis method for tool application scenarios. We use classification to represent all objects in the tool scene. We use a classification method to represent all objects in the tool scene according to the object characteristics.
- We propose the tool data modeling method based on an object deputy model, which effectively manages the tool data while meeting the most important requirements in machine tool processing.
- We conduct experimental evaluations for our tool data model and existing model based on the relational model. We evaluated the models from both storage space and retrieval time.

The remainder of this paper is organized as follows. We introduce our Tool Data modeling based on the Object Deputy Model in Sect. 2. Section 3 describes our modeling method implementation and evaluates its efficiency, before concluding the paper in Sect. 4.

2 Tool Data Modeling Method

In this section, we introduce the semantics of the tool data, illustrate the specific Tool Data Modeling Method based on the Object Deputy Model, and give cross-class query statements for the tool data scenarios.

2.1 Semantic Analysis

In our tool data model, we use a classification method to analyze data semantics. According to the processing technology, the tools are divided into four categories which are Drill, Mill Cutter, Boring Cutter, and Turning Tool. So we define four source classes to represent the four types of tools in the tool processing scene. In details, a class named Drill is used to define common drill objects which have basic attributes, such as drillid which is the unique identifier of the object, description which is recording the main purpose of the tool, material, Tconnection to describe the interface of the drill, and some geometric parameters that are diameter, included angle, programming length, cutting edge length and total

length. The other three almost have the same attributes as the drill. Therefore, only the drill is used as an example to describe the scene in the following. What's more, when the tools are installed on the machine, they need to be connected to the machine through a tool holder. So we need to design a tool holder source class to describe the tool holder interface information. In tool processing, if the interface of the tool and the interface of the holder are matched, they can be successfully paired and used. The workpiece to be processed is the most important part of tool processing, and the material characteristics of the workpiece play a decisive role in determining the processing parameters. We define a PartMaterial class to record material information. And the use case also affects the choice of cutting parameters, so we define a UseCase class to record the descriptions, indicating roughing or finishing.

2.2 Tool Data Modeling

In the previous chapter, we have introduced several important source classes. We present the complete model in this subsection. As shown in Fig. 1, the model contains four different deputy relationships, covering the main application scenarios of tool processing. Next, we will introduce specific modeling schemes for different data association relationships in the actual tool machining scene. We have adopted the classification standard of tools in the manufacturing field according to processing technology. Actually, each type of tool can be subdivided according to its material. Machine managers usually want to see as few unrelated tools as possible when looking for suitable tools. So we use the select operation to derive a deputy class which only includes the deputy objects of the instances of the source class that satisfy a selection predicate. Take Drill source class as an example, the drill contains many instances including high-speed steel drills, carbide drills, and so on. While processing a workpiece, the managers usually need to find a class of tools for its purpose, so the construction of the select deputy class can make it more convenient and express the relationship of different types of tools accurately. In Fig. 1, the select deputy class depicts the relationship of this situation.

When a tool is selected to machine a workpiece, it needs to be installed on the machine tool via a tool holder. However, the tool and tool holder are paired by the value of Tconnection, an attribute that records the interface criterion of the tool and tool holder. So we also construct a join deputy class to record the paired tools and tool holders. Further, looking up the suitable cutting speed is a very common scenario in tool machining when the other factors such as use case, material, and tool are exact. Therefore, we need to use these factors as filter conditions to query the cutting speed. We use the join operation to derive a deputy class that can store the relationship between them in order to express this relationship. In this way, we can find the cutting speed we need to find through this join deputy class. In Fig. 1, the join deputy class depicts the relationship of this situation. The Tool deputy class is defined with SQL-like statements as:

```
CREATE JOINDEPUTYCLASS Tool  AS (
SELECT  spiraldrill.id, toolholder.id,
toolholder.MConnection as MConnection
FROM spiraldrill, toolholder
WHERE spiraldrill.Tconnection=toolholder.TConnection)
```

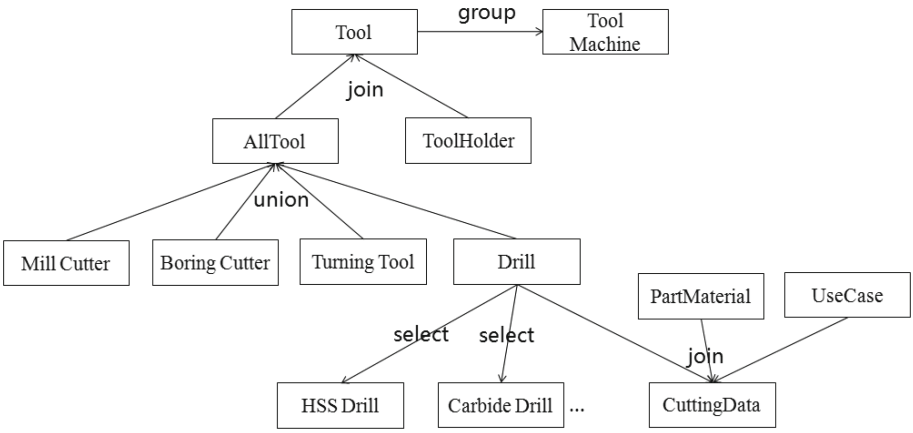


Fig. 1. The tool data model

As mentioned above, we represented four different tools as four different source classes, but we need to check all the tools during the inventory check, so we have to put all the tools together so that the type of tools is no longer single. The union operation in object deputy model can derive a deputy class of which extension consists of deputy objects of instances of more than one source class. Therefore, we use the union deputy class to describe the collection of four different types of tools. In Fig. 1, the union deputy class depicts the relationship of this situation.

During tool processing, the tool needs to be replaced frequently due to wear, and the interface for installing the tool on the machine tool is fixed. In a real-life scenario, when replacing the tools, we need to find the applicable tools for the machine tool through the interface information. Therefore, in order to facilitate the subsequent procurement of tools that can match the specific machine tool, we can know the number of tools applicable to each machine tool in the tool magazine through the tool interface information. The group operation of the object deputy model can derive a deputy class which only includes the deputy objects of the instances of the source class that have the same features. So we need to use the group deputy class to show this message.

2.3 Cross Class Query

In the previous part, we stored the tool data according to their semantic relationships. In the actual application scenario, machine operators hope to quickly retrieve the tool data that they want to query, so we can take advantage of the cross-class query feature in our model. Cross-class query refers to finding an object in a source class or deputy class based on the related information of the other source classes or deputy classes. The objects and their deputy objects are linked in two directions through pointers. So there is a path between any two objects in our model to link them. For example, in the tool data scenario, we usually need to find the optimal cutting speed of the tool when we already know the type of tool, the material of the workpiece, and so on. We can define the following cross-class query:

```
SELECT (spiraldrill{diameter='16.0'}->cuttingdata).cuttingspeed
FROM drill WHERE cuttingspeed IN
(SELECT(usecase{description='roughing'}->cuttingdata).cuttingspeed
FROM usecase WHERE cuttingspeed IN
(SELECT(partmaterial{description='GG'}->cuttingdata).cuttingspeed
FROM partmaterial);
```

In this cross-class query, we can see that using arrows to represent associations between classes is very simple and intuitive. The system will first scan the CuttingData class, and then check its connected source objects in the source class Drill, and see whether its diameter is equal to 16. If the filter condition is satisfied, then the system will judge whether the next filter condition is met until all filter conditions are met, and its attribute cuttingid will be returned, along with the value of its local attribute cutting speed. So the result of this query will be cutting speed. The cross-class queries use bidirectional pointers to find information in associated classes avoiding a large number of join operations in relational types. In general, the cross-class query makes the complex queries involving many filter conditions be presented clearly and executed efficiently.

2.4 Consistence Maintenance

In this section, we introduce how to achieve consistency maintenance by the update propagation mechanism [5]. The update propagation mechanism is that when an object is modified, the modification is automatically reflected its deputy object through a switching operation. There are usually three types of modifications in the tooling scenario. First, when a tool object is added, its deputy objects may be created automatically which satisfies the selection predicate so that the tools can be presented in different machining applications. Second, when an old tool is damaged, that is, a tool object is deleted, its deputy objects will also be deleted. Third, when a tool object is modified, some of its deputy objects may be deleted and some other deputy objects may be added. Thus, our tool data model can maintain the consistency of the tool data by the update propagation mechanism.

3 Experiments

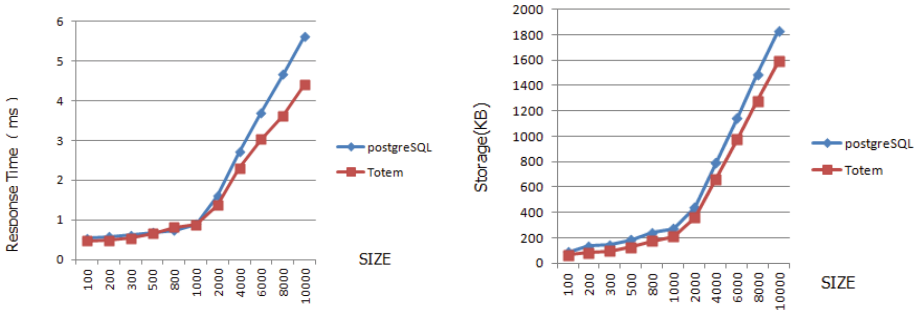
In this section, we introduce our performance testing experiments on the model from three aspects: platform, data set, and experimental results.

3.1 Platform

We report the different experiments we run to evaluate the performance of our tool data model. We evaluate the performance of each data model when processing the queries. We execute the query on two databases: Totem as a representative for the object deputy model, and PostgreSQL represents the relational model. Totem has been developed as a database management system (DBMS) based on the object deputy model [4]. In Totem, queries are expressed with a declarative query language which is named object deputy query language (ODQL) [6]. A query is executed on both of them, and meanwhile, the retrieval time and storage space are recorded. The experiments were performed using object deputy database Totem and relational database PostgreSQL under such a platform: Intel(R)Core(TM)i7-2320 3.0 GHz CPU with 4 GB memory, 500 GB hard disk and Ubuntu16.04.

3.2 Data Set

In the experiments, we used two types of data to evaluate the tool data models. One type was collected from the tool catalog which is from real-world use cases. But there are only a few hundred of such data. The other type was generated by simulation based on the attribute type of the real data. In this way, we get the data sets of different orders of magnitude. Every record in each data set represents a tool application instance which includes the unique identifier of the tool and its cutting speed corresponding to different machining conditions. We choose the main application scenario of tool data, namely query cutting speed, as our experimental scenario. We will compare the storage space of the tool data and the retrieval time needed when querying the same amount of tool objects between two data models based on two databases. We have done several tests with the different data sets. The test results of the response time and storage size are shown in Fig. 2. From the figures, we can see that the response time and the consumed storage spaces in Totem outperform the PostgreSQL.



(a) Comparison of Response Time. (b) Comparison of Storage Consumption.

Fig. 2. The experimental results

4 Conclusion

We presented the tool data modeling method based on the object deputy model. To the best of our knowledge, this model is the first implemented and evaluated in tool data management. And the results of the experiments show that the performance of the object deputy model is better compared to the relational model in the retrieval time and consumed storage space. We also achieved consistency maintenance of the tool data by the update propagation mechanism to make the tool management efficient. As future work, we will further improve query performance and optimize our model to meet more scenarios.

Acknowledgement. This work is supported by the Key Research and Development Program of China (2016YFB1000701) and the key projects of the National Natural Science Foundation of China (No. U1811263).

References

1. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM **13**(6), 377–387 (1970)
2. Kambayashi, Y., Peng, Z.: Object deputy model and its applications. In: Proceedings of the 4th International Conference on Database Systems for Advanced Applications (DASFAA), Singapore, pp. 1–15 (1995)
3. Peng, Z., Kambayashi, Y.: Deputy mechanisms for object-oriented databases. In: Proceedings of the Eleventh International Conference on Data Engineering, pp. 333–340 (1995)
4. Peng, Z., Peng, Y., Zhai, B.: Using object deputy database to realize multi-representation geographic information system. In: Proceedings of the 15th ACM International Symposium on Geographic Information Systems, ACM-GIS 2007, Seattle, Washington, USA, 7–9 November 2007, pp. 43–46 (2007)

5. Wang, L., Wang, L., Peng, Z.: Probabilistic object deputy model for uncertain data and lineage management. Data Knowl. Eng. **109**, 70–84 (2017)
6. Zhai, B., Shi, Y., Peng, Z.: Object deputy database language. In: Fourth International Conference on Creating, Connecting and Collaborating through Computing, C5 2006, Berkeley, California, USA, 26–27 January 2006, pp. 88–95 (2006)