The use of NoSQL in product traceability system construction

Chunsheng Hu, Xuejun Zhu*, Yang Zhou
School of Mechanical Engineering
Ningxia University
Yinchuan, China
jeadean@163.com

***Abstract*—Product traceability system is one significant part of the modern production system, and it supports the product quality assurance by providing traceable life-cycle information about one product. By using entity-relationship model and relational database to construct a product traceability system, some problems arise when facing complex products. Three problems include entity/table explosion, query disaster and sorting problem are analyzed, then NoSQL is adopted to overcome these disadvantages. Finally, a case study about a product traceability system for pleurotus eryngii cultivation is introduced. The result demonstrates that the document type NoSQL can fix the entity/table explosion, query disaster and sorting problem easily. This solution can be used as references to those products that have plenty of unstructured production components, stages and process.**

Keywords- product traceability system; NoSQL; agricultural product

I. INTRODUCTION

According to ISO 8402, traceability is "the ability to trace the history, application or location of an entity by means of recorded identifications" [1]. It is no doubt that the product traceability system is one significant part of the modern production system [2]. The product traceability system supports the product quality assurance by providing traceable life-cycle information about one product. Moreover, it not only saves the history information, but also provides basic and essential data for the adjustment and optimization of the production process of a product [3].

To build a product traceability system, a sub-system of life-cycle information recording should be built first according to the production process model, and this work usually means building entity-relationship model and create tables in a relational database, usually an SQL software. This mode can work well when facing simple product or simple life-cycle information structure, for example a traceability system only to show one product's batch, production date, quality check, performance parameters and so on. But when facing a complex product or complex life-cycle information structures, some problems arises which include entity/table explosion, query disaster and sorting problem.

II. DEFECTION OF RELATIONAL DATABASE

A. Entity/Table Explosion

One product traceability system promises one key traceability number links to the process steps information, raw materials information, components information, employees information and machines information. Just consider that one product composed of hundreds of different components, each of which have different data structure, and each of which need one table to record its information, that is a piece of hard work to deal with but not invincible. There are multiple implementations and Fig. 1 shows one classic data table model.

When components number increases to thousands, it becomes a work too desperate to turn and face, not to mention that these components can compose various parts and they are associated with different production stages.

The situation here can be characterized as "Entity/table explosion" with the increasing complexity of the production process. The root cause of this phenomenon is that the relational database technology needs all information be formatted to various fixed data structures, so when facing hundreds of different components, hundreds of different entities and corresponding tables are generated.

There are two common solutions to deal with this situation: 1. Common properties of all components are extracted and unique properties are discarded, and one data table fits all components can be got; 2. Common properties of all components are extracted to build a universal data table, then one field is added into this table, in this field all unique properties of one component are recorded in a special format usually string.

The deficiency of solution 1 is losing of information. While solution 2 keeps all information but losing the opportunity to use the retrieval function of database software to retrieve these unique properties.

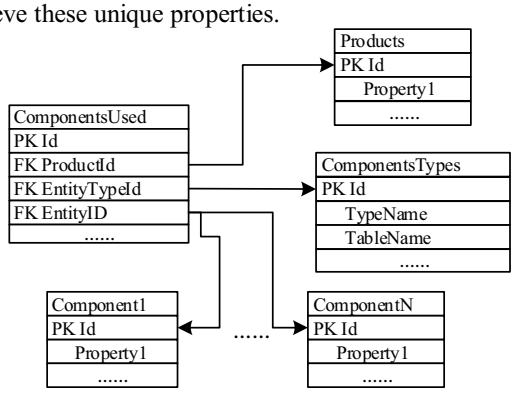

Figure 1. A relational model to record components' information

978-1-5386-5500-9/18/$31.00 ©2018 IEEE
DOI 10.1109/ICISCE.2018.00124

B. Query Disaster

A direct result of entity/table explosion is that the information query becomes too complex. If all components' detailed information about one product need to be listed, hundreds of tables need to be queried, and each one returns one type of specific data structure. If all components from Europe need to be found, each table has to be traversed as well. Even if we don't consider table-join query, complex query statements are difficult to construct. If some components composed one part, new field and tables have to be constructed to record these relations, and inevitably new complex query statements need to be constructed if we want to get this part's detailed information. Beyond that, data tables' maintenance can be another big disaster for developers. The root cause for this disaster also lays upon fixed structured data.

C. Sorting Problem

For linear production process, the sorting problem is not so obvious. Usually, information is naturally sorted according to the production stages listed one by one. But for nonlinear productive process, trouble arises. Consider one agricultural productive process in which watering action, fertilizing action and weeding action are not followed one by one in order, they are irregularly carried out. The watering action involves employee and watering device; the fertilizing action involves employee, fertilizer and fertilizer device; the weeding action involves employee and weeding device. A typical data table model may like Fig. 2.

We can easily get three sets of actions for one specific product, each one set corresponds to one kind of action, and in each set, actions can be easily sorted by the field "ExecutionTime". But if we want to get all three actions showed together and sorted by ExecutionTime, we have to do extra work to sort three sets with different structure. When the action number increases to hundreds or thousands, the sorting working becomes one new disaster. The root cause for this problem also lays upon fixed structured data.

Because all three problems are attributed to fixed structured data, NoSQL which can store and retrieve unstructured data, seems can provide one potential solution to solve these problems.

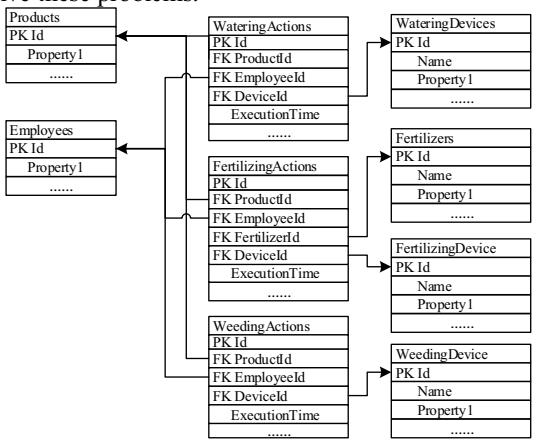

Figure 2. One relational model to record agricultural production process

III. ADVANTAGES OF NOSQL

A. NoSQL

A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases [4].

Non-relational is one of its key points, which means all different data structures linked to one kind of product can be collected together and stored in one "Table". This characteristic can solve the Entity/table explosion problem.

Although the query function of NoSQL software is not as powerful as some commercial relational SQL software, it is strong enough to retrieve all detailed information. Meanwhile, because the Entity/table explosion is eliminated, the query disaster is avoided too.

Sorting based on specific filed for one "Table" is one basic function of NoSQL database, and because different data structure can be stored in one “Table”, the Sorting problem is fixed easily.

There are 4 basic types of NoSQL databases classified by the data model. They are Column type, Document type, Key-value type and Graph type, each of which has specialty and suitable application scenario. The Document-type-NoSQL database is the most suitable one for product traceability system [5,6].

B. Convertion from SQL to NoSQL

In a Document type NoSQL database, a "Collection" is a set of "Document", like a "Table" is a set of "Row" in relational SQL database. And a document is one Key-Value pair in JSON format, in which Value can be different type of structured data.

Take Figure 1 as an example, all different types of components' can be stored as documents in one collection named "CompCollection", and each one document has its unique data structure, as shown below:

```
{
"_id" : ObjectId("xxxxxxxxxxxxxx"),
"ProductId":"20180409124",
"ComponentName":"Comp1",
"ComponentType":"type1",
"ComponentSerialNumber":"2021546",
"Cost":20,
"Length":3.6,
"Height":0.6,
……}
{"_id" : ObjectId("xxxxxxxxxxxxxx"),
"ProductId":"20180409124",
"ComponentName":"Comp2",
"ComponentType":"type2",
"ComponentSerialNumber":"0xfe77cd",
"Cost":450,
"Material":"Aluminium alloy",
……}
……
{"_id" : ObjectId("xxxxxxxxxxx"),
"ProductId":"20180409124",
"ComponentName":"Compn",
"ComponentType":"typek",
"ComponentSerialNumber":"785756",
"Cost:"53,
"Weight":3.7,
"PoreNumber":1,
"PoreSize":0.63,
```

……}

Notice that all these components have one same filed "ProductId" which means that they all belong to one same product.

Only one simple query statement can get all components' information:

db.CompCollection.find({"ProductId":"20180409124"})

If we want get all components used in one production and belong to type1, here is the query statement:

db.CompCollection.find({"ProductId":"20180409124","ComponentType":"type1"})

If we want to get all components used in one production that has a pore with size<0.5,weight<20 and cost>5, here is the query statement:

db.CompCollection.find({"ProductId":"20180409124", "PoreNumber":1, "PoreSize:{$lt:0.5}, "Weight" : {$lt : 20}, "Cost" : {$gt : 5} })

Give a thought to the same query statement in relational SQL database that has hundreds of different components tables, comparison seems so self-evident.

Take Figure 2 as another example, all three actions can be stored as documents in one collection named "Actions", as shown below:

{"_id" : ObjectId("xxxxxxxxxxxxx"),
"ProductId":"20180466",
"ActionType":"Watering",
"EmployeeId":"24824",
"DeviceId":"202",
"ExecutionTime":ISODate("2018-04-01 T00:00:00.00Z"),
……}
{"_id" : ObjectId("xxxxxxxxxxxxxx"),
"ProductId":"20180466",
"ActionType":"Fertilizing",
"EmployeeId":"24843",
"FertilizerID":"23",
"DeviceId":"107",
"ExecutionTime":ISODate("2018-04-02 T00:00:00.00Z"),
……}

The above two documents still remain SQL style, because we need to retrieve a device's information from another collection by using DeviceId. But benefited from NoSQL, a document's value can be another document, so these two documents can be stored like this:

{"_id" : ObjectId("xxxxxxxxxxxxx"),
"ProductId":"20180466",
"ActionType":"Watering",
"Employee":
{"ID":"24824",
"Property 1":"xxx",
"Property 2":"xxx",
..... },
"Device":
{"ID":"202",
"Property 1":"xxx",
"Property 2":"xxx",
..... },
"ExecutionTime":ISODate("2018-04-01 T00:00:00.00Z"),
……}
……
{"_id" : ObjectId("xxxxxxxxxxxx"),
"ProductId":"20180466",
"ActionType":"Fertilizing",
"Employee":
{"ID":"24843",
"Property 1":"xxx",
"Property 2":"xxx",
..... },
"Fertilizer":
{"ID":"23",
"Property 1":"xxx",
"Property 2":"xxx",
..... },
"Device":
{"ID":"107",
"Property 1":"xxx",
"Property 2":"xxx",
..... },
"ExecutionTime":ISODate("2018-04-02 T00:00:00.00Z"),
……}

If we want to get all actions for one product ordered by ExecutionTime, sorting Problem can be easily fixed:

db.Actions.find({"ProductId":"20180466"}).sort({"ExecutionTime":-1})

As we can see, in product traceability system construction, all three problems can be solved by using a document type NoSQL database.

IV. CASE STUDY

We have built a product traceability system for Pleurotus eryngii (one kind of mushroom) cultivation, and this system is based on relational Database [7-9]. The database is planned to record life-cycle information about different batch of mushroom, and each one batch has the same traceability number coded in EAN/UCC-14 style. Information about staffs, raw materials, working rooms, sensors, tools, devices, environmental adjustments and production actions need to be recorded. Here is the simplified E-R model for this system.

This design made several compromises. To avoid Entity/Table explosion, operation actions are formatted into one kind of fixed structure. However, not all actions need room, device, tool and material at the same time, and some actions may need more than these four kinds of things. The compromise causes redundant data table structure for some actions and information loss for some other actions. The same compromises are made for the environment adjustment actions and environment data.

When facing sorting of operation history, environment adjustment history and environment data, problem appears because these three things have different structures.

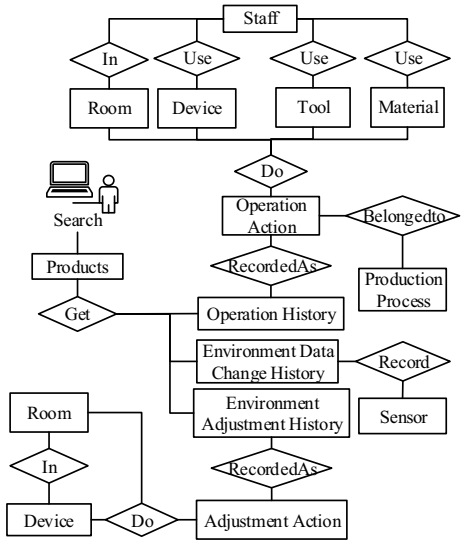

Figure 3. Simplified E-R model of a product traceability system for Pleurotus eryngii cultivation.

To improve this product traceability system, a NoSQL database (MongoDB) is added while the original SQL database is still working. The SQL database records information about the staffs, devices, tools, sensors, products, materials and rooms, because the information needs to interact with other existing information management systems. While the NoSQL database records all the actions. Each action is generated according to its unique feature, like:

```
{"_id" : ObjectId("xxxxxxxxxxxxx"),
"ProductId":"20180466",
"ActionType":"OperationAction",
"Employee":
        {"ID":"24824",
         "Property 1":"xxx",
         "Property 2":"xxx",
        ..... },
"UseDevice":
        {"ID":"202",
         "Property 1":"xxx",
         "Property 2":"xxx",
        ..... },
"ActionName":"Move",,
"InRoom":
        {"ID":"202",
         "Property 1":"xxx",
         "Property 2":"xxx",
        ..... },

"ExecutionTime":ISODate("2018-04-01 T00:00:00.00Z"),
……}
```

Just take "get all actions be done to one batch of product and order them in ascending order" as an example, table 1 shows the main benefits be brought by using of NoSQL.

In this case, because all the information stored in NoSQL Database comes from the relational dataset, the information integrity of the two is exactly the same. No matter what kind of database is used, the final displayed history records have no difference. For users that use this system, the biggest feeling may be that the retrieval time is shorter. But for users that design this system, the implementation details have been changed a lot. Moreover, all the records can be adjusted and various of new data structures can be added easily in NoSQL database compared to Relational Database.

TABLE I. BENEFITS BY USING OF NOSQL

Comparative Items	Relational Database	NoSQL
Information Integrity	yes	yes
Database Tables number	13	1
Query statements length(lines)	23	1
Query time (ms)	156	273
Sort different structured results in ascending order	Customized algorithm	No need
Sort time (ms)	782	0(No need)
Query results integration (to a JSON string) time (ms)	237	0(No need)
Adjust existing data structure	Complex	Simple
Expand new data	Complex	Simple

V. CONCULISION

Building product traceability system for modern complex product by using relational SQL database becomes harder and harder, and the NoSQL database provides a reasonable solution. For it can fix the Entity/Table Explosion, Query Disaster and Sorting Problem easily. In a product traceability system, the NoSQL database is not the substitution of relational SQL database, but a supplement.

This paper presents one case of application of NoSQL in product traceability system for Pleurotus eryngii cultivation, which is not particularly complicated. The result shows the document type NoSQL is very suitable to construct a product traceability system. This solution can be applied to other products that have plenty of unstructured production components, stages and process, the more complex the products are, the more potential advantages will be got.

Future work includes:1) improve the query efficiency of this solution; 2) improve the database structure for further data analysis; 3) try to create growth parameters database and model relations among actions, environment data and growth parameters.

ACKNOWLEDGMENT

This work was supported by National Science &Technology Pillar Program (2013BDA16B04).

REFERENCES

[1] International Organization for Standardization. (1994). ISO 8402: 1994: Quality Management and Quality Assurance-Vocabulary. International Organization for Standardization.

[2] A. Parreño-Marchante , A. Alvarez-Melcon , M. Trebar, ,& P. Filippin, (2014). Advanced traceability system in aquaculture supply chain. Journal of food engineering, 122, 99-109.

[3] R. Badia-Melis, P. Mishra, & L. Ruiz-García (2015). Food traceability: New trends and recent advances. A review. Food Control, 57, 393-401.

[4] http://nosql-database.org/

[5] S. Sharma, R. Shandilya, S. Patnaik, & A. Mahapatra, (2016). Leading NoSQL models for handling Big Data: a brief review. International Journal of Business Information Systems, 22(1), 1-25.

[6] A. Haseeb, & G. Pattun, (2017). A review on NoSQL: Applications and challenges. International Journal of Advanced Research in Computer Science, 8(1).

[7] Y. Zhou, C. Hu, & X. Zhu, (2017, July). The data analysis and the database construction for Pleurotus eryngii product traceability system. In Information Science and Control Engineering (ICISCE), 2017 4th International Conference on (pp. 712-716). IEEE.

[8] R. Qi, & Z. Xuejun (2016). Design of pleurotus eryngii production traceability system based on ZigBee technology. Journal of Chinese Agricultural Mechanization, 8, 042.

[9] L. Zhao, & X. Zhu, (2015, August). The development of remote monitoring system for cultivation environment of Pleurotus eryngii. In Information and Automation, 2015 IEEE International Conference on (pp. 2643-2648). IEEE.