Advances in Engineering Research, volume 164
8th International Conference on Manufacturing Science and Engineering (ICMSE 2018)

Research on Data Redundancy and Query Efficiency of Helicopter Database System

Junhong Guo[1,a,*], Yi Huang[1], Hong Li[1] and Xiang Li[1]
1. Information and Intelligence Unit, Army Aviation Research Institute, Beijing 101121, China
[a]emtfemtf@163.com

Keywords: Data Redundancy, Query Efficiency, Helicopter, Database System.

Abstract. When designing database table structure, it is necessary to balance the design normal forms against properly retain redundant data. Based on design normal forms of relational database, a method called "Record relationship mark" is introduced in this article. The method helps to reduce program complexity and improve the query efficiency.

Introduction

When designing database table structure [1-3], the following three normal forms should be generally followed:

The first normal form: atomicity constraints. Requires attributes with atomic, non-decomposable;

The second normal form: uniqueness constraints. Requires unique identification of records, such as the uniqueness of the entity;

The third normal form: Redundancy Constraints. Any field cannot be derived from other fields, which requires no redundant fields.

A database that does not contain redundancy is not necessarily the best database. Sometimes in order to improve operational efficiency, we should not in accordance with the standard normal form. Proper retention of redundant data to achieve the purpose of space for time.

In this paper, taking a database of helicopter data query system as an example, we introduce a method to improve the query efficiency by preserving redundant data in the database. The method is to add "Record relationship mark" to the table which helps to reduce program complexity and improve the query efficiency[4-7].

Record Relationship Mark

In the helicopter data query system, helicopter data should be classified into many categories according to the tasks helicopters undertake, the weight levels and the structure forms of helicopters. This data structure can be represented using tree data structure, which is stored in a category table. The tree data structure contains a root node, there are several sub-nodes below the root node. Some sub-nodes also have several sub-nodes (Fig. 1). In accordance with the standard design normal form, based on the tree data structure, a data table structure will be designed to store these categories (Table 1).

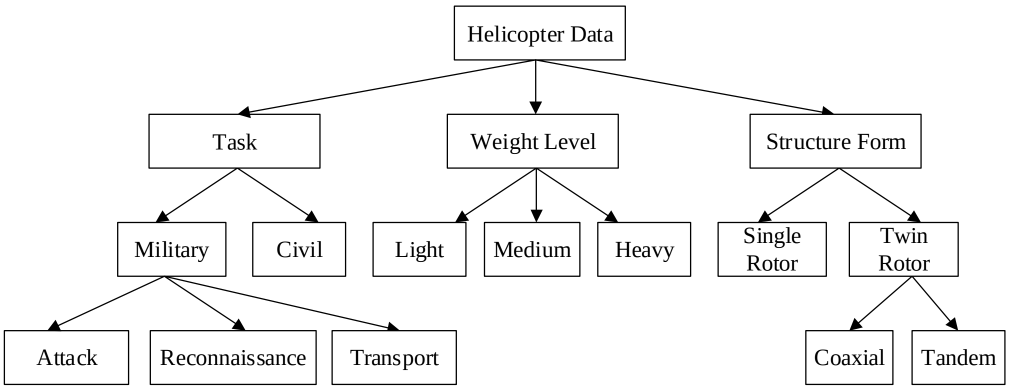

Figure 1. Helicopter data classification structure

Table 1. Data table structure in accordance with the standard design normal form

Name	Variable type	Null Value	Constraint	Description
category_ID	int	No	No repetition	Primary key
category_Name	char(100)	No		Category name
category_parent_ID	int	No		The parent category_ID of this category, set to 0 if it is root node

The above design complies with the third normal form, which correctly represents the classification structure and does not have redundant data. But this design is not the best.

Let's imagine a common usage scenario. Users want to list all the classified data of helicopters, and display them according to the level relationship, as shown in Fig. 2.

Helicopter Data
- 1 Task
    - 1.1 Military
        - 1.1.1 Attack
        - 1.1.2 Reconnaissance
        - 1.1.3 Transport
    - 1.2 Civil
- 2 Weight Level
    - 2.1 Light
    - 2.2 Medium
    - 2.3 Heavy
- 3 Structure Form
    - 3.1 Single Rotor
    - 3.2 Twin Rotor
        - 3.2.1 Coaxial
        - 3.2.2 Tandem

Figure 2. The hierarchical list of all the classified data of helicopters

In order to display the list shown in Fig. 2 (preorder traversal of the tree), the program needs to retrieve the above table many times using the following SQL iteration command, which is inefficient. Therefore, we expand the data table, add a certain amount of redundant data, and limit the maximum number of categories, then we can complete the above hierarchy list by a single search. Table 2 shows the expanded data table structure.

SQL iteration command:

```
WITH CTE AS
(
SELECT category_ID, category_Name, category_parent_ID, rn=Cast(category_ID as varchar(max)) FROM
category_table_Helicopter1 WHERE category_ID in (select category_ID From category_table_Helicopter1 WHERE
category_parent_ID ='0')
UNION ALL

SELECT T.*,rn=CTE.rn+Cast(T. category_ID AS VARCHAR(MAX)) FROM category_table_Helicopter1 T,CTE
WHERE CTE. category_ID =T. category_parent_ID
)
SELECT category_ID, category_Name, category_parent_ID FROM CTE ORDER BY rn
```

Table 2. Data table structure with expanded data

Name	Variable type	Null Value	Constraint	Description
category_ID	int	No	No repetition	Primary key
category_Name	char(100)	No		Category name
category_parent_ID	int	No		The parent category_ID of this category, set to 0 if it is root node
category_level_Mark	char(6)	No	No repetition	Level relationship mark, limited to 3 levels, the initial value is 000000

The optimized database table structure is shown in Table 2. The new table structure adds a level mark field named category_level_Mark, which is a fixed-length string of 6 bits.

The following rules are used for the value of category_level_Mark: 6 bit characters are all numbers. The first two characters are the first level classification number, the middle two are the second level classification number, and the last two are the third level classification number.

For example, "010200 " represents the second sub-category under the first category, namely "1.2 Civil" in Fig. 2

If you want to increase the level of the classification, you can increase the length of the category_level_Mark. In theory, as long as the level of classification is planned in advance, the number of levels can be unlimited. Table 3 is the helicopter category data table of the database.

Table 3. Helicopter category data table of the database

category_ID	category_Name	category_parent_ID	category_level_Mark
1	Helicopter Data	0	000000
2	1 Task	1	010000
3	1.1 Military	2	010100
4	1.1.3 Transport	3	010103
5	1.2 Civil	2	010200
6	2 Weight Level	1	020000
7	2.1 Light	6	020100
8	2.2 Medium	6	020200
9	2.3 Heavy	6	020300
10	3 Structure Form	1	030000
11	3.1 Single Rotor	10	030100
12	3.2 Twin Rotor	10	030200
13	3.2.1 Coaxial	12	030201
14	3.2.2 Tandem	12	030202
15	1.1.1 Attack	3	010101
16	1.1.2 Reconnaissance	3	010102

As can be seen from Table 3, the categories "1.1.1 Attack" and "1.1.2 Reconnaissance" are the records added after the category "3.2.2 Tandem". Use the following SQL command[8] to sort the value of category_level_Mark, the required record set can be obtained (Table 4).

SQL command : SELECT * FROM category_table_Helicopter2 ORDER BY category_level_Mark

Table 4. Required record set of helicopter category data

category_ID	category_Name	category_parent_ID	category_level_Mark
1	Helicopter Data	0	000000
2	1 Task	1	010000
3	1.1 Military	2	010100
15	1.1.1 Attack	3	010101
16	1.1.2 Reconnaissance	3	010102
4	1.1.3 Transport	3	010103
5	1.2 Civil	2	010200
6	2 Weight Level	1	020000
7	2.1 Light	6	020100
8	2.2 Medium	6	020200
9	2.3 Heavy	6	020300
10	3 Structure Form	1	030000
11	3.1 Single Rotor	10	030100
12	3.2 Twin Rotor	10	030200
13	3.2.1 Coaxial	12	030201
14	3.2.2 Tandem	12	030202

The sequence of records listed in table 4 is exactly the result of the first-order traversal. By determining the value of the category_level_Mark, you can control the display position of the category level: Divide each 2 digits into a group. If the value of a group is greater than 0, the display position is shifted to the right by 2 spaces. The limit in this example is a maximum of 3 levels, with up to 99 sub-categories per level. The length and number of the category_level_Mark can be modified to represent more levels.

Conclusions

When designing database table structure, it is an issue that we need to think about carefully to balance the design normal forms against properly retain redundant data. The "Record relationship mark" method can largely reduce program complexity and improve system performance by properly retaining redundant data.

Acknowledgments

This work is financially supported by the Foundation for Basic Research Projects of Army Aviation Research Institute

References

[1] Larry Rockoff. The Language of SQL: How to Access Data in Relational Databases. Course Technology PTR (2010).

[2] Abraham Silberschatz, Henry F.Korth, S.Sudarshan Database System Concepts. China Machine Press (2013).

[3] Paul DuBois MySQL Cookbook: Solutions for Database Developers and Administrators (3rd Revised edition). O'Reilly Media, Inc, USA (2014).

[4] Michael J. Hernandez Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design (3rd Edition). Addison-Wesley Professional (2013).

[5] Thomas Connolly, Carolyn Begg Database systems: a practical approach to design, implementation and management sixth edition. Pearson (2015).

[6] P. Bizarro, N. Bruno, D. De Witt. Progressive Parametric Query Optimization[J]. Trans. on KDE, Apr. 2009.

[7] Maria Carina Roldan Learning Pentaho Data Integration 8 CE - Third Edition: An end-to-end guide to exploring, transforming, and integrating your data across multiple sources. Packt Publishing (2017).

[8] Alan Beaulieu Learning SQL,2 Edition. Post & Telecom Press (2015).