Postprint of paper in *Proceedings of 2022 IEEE International Symposium on Software Reliability Engineering Workshops* (*ISSREW '22*), IEEE, Piscataway, NJ, USA (2022), pp. 130–131

A Disjoint-Partitioning Approach to Enhancing Metamorphic Testing of DBMS

Matthew Siu-Hin Tang, T. H. Tse*
Department of Computer Science
The University of Hong Kong
Pokfulam, Hong Kong
tshcat@connect.hku.hk, thtse@cs.hku.hk

Zhi Quan Zhou
School of Computing and IT, University of Wollongong
Wollongong NSW 2522, Australia, and
Alibaba and Ant Group, Hangzhou, China
zhiquan@uow.edu.au

***Abstract*—Owing to big data, DBMS testing faces the oracle problem, that is, it is difficult to verify execution results against expected outcomes. Rigger and Su applied metamorphic testing to alleviate the challenge. We propose a disjoint-partitioning approach to extend their work. We have conducted an empirical case study on OceanBase, the DBMS associated with the world's fastest online transaction processing system. Even though OceanBase has been extensively tested and widely used in the industry, we have unveiled various hidden failures and crashes.**

***Index Terms*—test oracle, metamorphic testing, DBMS, SQL, OceanBase, failure**

I. INTRODUCTION

Because of the popularity and financial impacts of online transaction processing (OLTP) systems, the correctness of the supporting database management systems (DBMS) is essential. However, owing to the scale of large databases, DBMS testing faces the test oracle problem, which refers to the difficulty in verifying the execution results [1]. The metamorphic testing (MT) approach [2][3] was invented in 1998 to alleviate the problem. In 2020, Rigger and Su [4][5][6] applied MT to tackle the issue in DBMS testing. They proposed the query partitioning (QP) and ternary logic partitioning (TLP) techniques. Their empirical studies revealed failures in five DBMS.

* Corresponding author.

TABLE 1 entry_exit TABLE

id	name	entry_exit_count
1	Alice	4
2	Bob	1
...	...	...
10000	Alice	3

TABLE 2 EXTENDED entry_exit TABLE

id	name	entry_exit_count	entry_exit_count & 1
1	Alice	4	0
2	Bob	1	1
...	...	...	...
10000	Alice	3	1

In this project, we conduct an in-depth investigation on Rigger and Su's work. We find a gap between QP and TLP, and introduce the concept of disjoint partitioning to address the issue. We have undertaken an empirical case study on OceanBase [7], which was developed by Alibaba and Ant Group and is associated with the world's fastest OLTP. We have identified various hidden failures and crashes.

II. MOTIVATION EXAMPLE IN DBMS TESTING

Consider Table 1, which records the id, name, and entry_exit_count of 10 000 staff in a large corporation. By examining the values of entry_exit_count for each staff, we can determine that those with odd values (such as the cases of Bob and the second Alice) are inside the building, whereas those with even values (such as the case of the first Alice) are outside.

We can use the SQL bitwise AND (&) operator with the operand 1 to determine whether an entry_exit_count is odd or even. An odd number & 1 returns an integer 1, and an even number & 1 returns an integer 0, as shown in Table 2. Hence, we can write the following SQL statement to retrieve all the staff (with an odd entry_exit_count) *inside* the building:

```
(Q0) SELECT * FROM entry_exit WHERE entry_exit_count & 1;
```

OceanBase returns a list with 3000 staff. However, it is very difficult to determine whether the result is correct or not.

III. METAMORPHIC TESTING

In software testing, a *test oracle* is the mechanism to verify the consistency between the expected outcome and the actual execution result. The *test oracle problem* refers to the situation where such a mechanism is missing, or is extremely difficult to apply [1]. As illustrated in the motivating example, it may be difficult to verify the result of a given query in large databases.

In 1998, T. Y. Chen invented *metamorphic testing* (*MT*) [2][3] to alleviate the test oracle problem. He defines *metamorphic relations* (*MRs*) as necessary properties of the target program with regard to **multiple** inputs and their expected outputs. To implement MT, some program inputs are first constructed as original test cases (called *source* test cases). Based on an MR, new inputs are then constructed as *follow-up* test cases. Unlike traditional testing, which verifies the correctness of each individual test result, MT verifies the relation

among the source and follow-up inputs and outputs with reference to the MR. More than 500 papers on MT have been published. However, research work on the application of MT to DBMS is limited.

When applying MT to DBMS testing, the source test case may be an initial query, whereas the follow-up test case may be a subsequent query constructed based on an MR. The relation between their outputs (resultant lists returned by the DBMS) can then be verified with respect to the specified MR. In the previous example of the entry_exit_count, we may apply an MR such that if we add another condition to our query, the DBMS should return fewer records. For instance, we may refine our query to ask for all the records WHERE entry_exit_count & 1 AND name <> 'Alice'. Upon execution, if the DBMS returns > 3000 results, it is a failure. In this way, failures may be revealed by MT without the need for an ideal test oracle.

IV. Previous Work by Rigger and Su

In 2020, Rigger and Su applied MT to alleviate the oracle problem in DBMS testing [4][5][6]. They proposed a general concept called query partitioning (QP) [5] for revealing DBMS failures. The process is as follows: *1)* Construct an initial query (serving as a source test case). *2)* Construct multiple queries, called partitioning queries (serving as follow-up test cases), each producing partial results of the source query. *3)* Combine all the partial results to give a resultant list. *4)* Compare the resultant list with the original result of the source query.

QP describes a general MR among DBMS queries. Outputs of the follow-up test cases are sublists of the output of the source test case. The MR specifies that the concatenation of the outputs of the follow-up test cases must be equal to the output of the source test case. A violation of this MR indicates a failure.

Rigger and Su further proposed ternary logic partitioning (TLP) [5] as a specific case of QP. An original query is decomposed into three partitioning queries, which are three predicate variants derived from a randomly generated predicate. Following the MR stated in QP, by concatenating the three resultant lists, one should obtain the result of the original query. TLP has revealed failures in various DBMS.

V. Extension of Rigger and Su to Disjoint Partitioning

After thorough investigation, we find that Rigger and Su's proposed QP as a strategic idea is too general for practical MR constructions, while TLP as a special implementation of the strategic idea is too specific. We propose an innovative technique, called *disjoint partitioning*, to fill the gap. The core idea is that resultant lists returned by DBMS can be divided into partitions that are **exhaustive** and **mutually exclusive** using, for instance, the SQL keywords LIMIT and OFFSET.

Consider the motivating example again. We propose follow-up queries that divide the rows in the original table into disjoint partitions with the same WHERE condition. We write the two follow-up queries as follows:

(Q1)
```
SELECT * FROM (SELECT * FROM entry_exit
LIMIT 5000 OFFSET 0) AS offset1
WHERE entry_exit_count & 1;
```

(Q2)
```
SELECT * FROM (SELECT * FROM entry_exit
LIMIT 5000 OFFSET 5000) AS offset2
WHERE entry_exit_count & 1;
```

(Q1) retrieves the staff with an odd entry_exit_count in rows 1 to 5000, whereas (Q2) retrieves those in rows 5001 to 10 000. OceanBase returns 1500 and 1501 staff, respectively.

The MR states that simple **concatenation** of the outputs of follow-up queries (Q1) and (Q2) should produce the same result as the original query (Q0). The MR is therefore violated because there are more rows in the concatenated output from (Q1) and (Q2). Thus, we have revealed a hidden failure.

VI. Empirical Case Study

We have implemented the proposed disjoint-partitioning approach to testing OceanBase Community Edition version 3.1.0. Our tool automatically constructs random source queries, random number of partitions, and random follow-up queries. It generates source and follow-up test cases randomly. It also verifies the results according to the MRs. We have unveiled various hidden failures and crashes. Some of them have been fixed in a subsequent release version 3.1.1 by the OceanBase QA team of the Ant Group. Others are being investigated or scheduled for fixes in future releases.

VII. Summary

We have conducted an in-depth investigation into existing metamorphic testing techniques in DBMS testing. We have identified a gap between query partitioning and ternary logic partitioning in Rigger and Su's work. We address the issue using a new concept of disjoint partitioning. We have applied the approach to OceanBase. We have unveiled various hidden failures and crashes, even though OceanBase has been tested extensively and applied widely in the DBMS community.

Acknowledgments

This project was supported in part by an internship of the first author at Alibaba and Ant Group, China. We would also like to thank their QA team for confirming our failure reports.

References

[1] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE Transactions on Software Engineering*, vol. 41, no. 5, pp. 507–525, 2015.

[2] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. H. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities," *ACM Computing Surveys*, vol. 51, no. 1, pp. 4:1–4:27, 2019.

[3] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortes, "A survey on metamorphic testing," *IEEE Transactions on Software Engineering*, vol. 42, no. 9, pp. 805–824, 2016.

[4] M. Rigger and Z. Su, "Detecting optimization bugs in database engines via non-optimizing reference engine construction," in Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20), New York, NY: ACM, 2020, pp. 1140–1152.

[5] M. Rigger and Z. Su, "Finding bugs in database systems via query partitioning," *Proceedings of the ACM on Programming Languages*, vol. 4, no. OOPSLA, pp. 211:1–211:30, 2020.

[6] M. Rigger and Z. Su, "Testing database engines via pivoted query synthesis," in *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation* (*OSDI '20*), Berkeley, CA: USENIX Association, 2020, pp. 667–682.

[7] *OceanBase*, 2021. [Online]. Available: https://www.oceanbase.com/en