Al-Farouk journal of sciences
Volume (2), Issue (3) (JUNE, 2026)
ISSN: 3135-2359

مجلة الفاروق للعلوم
المجلد (2) العدد 3 يونيو 2026
ردمد: 3135-2359

RTTS: A Replicated Temporary Table Strategy for Resolving Temporary Table Reopening Failures in MySQL 5.7 Academic Registration Systems

Lutfia Mohamed Hagrassi
Computer Department, The Higher Institute of Industrial Technology, Tripoli, Libya
lutfia.hagrassi.cs@gmail.com

تاريخ الاستلام: 2026/04/02 تاريخ المراجعة 2026 /05/01 تاريخ القبول:2026/05/14- تاريخ النشر: 2026 /06/02

Abstract

Academic registration systems frequently rely on stored procedures to process prerequisite validation, course eligibility assessment, academic restrictions, and concurrent student enrollment operations. In legacy MySQL environments, particularly MySQL 5.7, temporary tables are commonly used to simplify intermediate query processing. However, MySQL exhibits documented limitations related to reopening temporary tables when the same temporary dataset is accessed through multiple execution paths, nested subqueries, or complex join structures. These limitations often generate runtime errors such as “Can't reopen table” and “Table doesn't exist,” causing registration procedures to fail. This paper presents a real-world case study based on the GetAvailableCourses procedure implemented within an academic registration system serving 1,833 students and maintaining 165,282 historical academic records spanning multiple student cohorts from the 1990s to the present.

The study analyzes the root causes of temporary table reopening failures and proposes a Replicated Temporary Table Strategy (RTTS), in which synchronized temporary table replicas are created and assigned to independent execution paths. Experimental deployment within a production environment running MySQL 5.7.44 eliminated observed reopening failures while preserving existing business logic. The results demonstrate improved execution stability, maintainability, and reliability in academic registration workflows.

Keywords: MySQL 5.7, Temporary Tables, Stored Procedures, Academic Registration Systems, Query Optimization, Database Reliability, RTTS.

1. Introduction

Academic registration systems constitute one of the most transaction-intensive components of higher education information systems. These systems must simultaneously enforce prerequisite requirements, semester restrictions, academic regulations, and seat availability constraints while supporting concurrent student requests.

Many institutions continue to operate on legacy MySQL infrastructures due to deployment stability and cost considerations. In such environments, stored procedures frequently utilize temporary tables to store intermediate datasets and reduce query complexity.

Despite their advantages, temporary tables introduce execution challenges in MySQL 5.7. Under certain execution plans, the optimizer attempts to reopen temporary tables multiple times across nested joins and subqueries. This behavior may produce runtime failures that interrupt critical enrollment operations.

This study investigates a real production incident involving the GetAvailableCourses procedure and proposes a practical mitigation strategy suitable for legacy MySQL environments.

2. Research Motivation and Dataset Characteristics

The investigated registration platform serves as a production academic information system supporting student enrollment, prerequisite validation, course eligibility determination, and semester registration management.

The system operates on a MySQL 5.7.44 database environment and includes historical academic records accumulated over several decades. The database maintains academic histories for both active and archived student cohorts, with records dating back to the 1990s.

The core procedure analyzed in this study, GetAvailableCourses, is responsible for determining student-specific registration eligibility based on academic performance, failed courses, prerequisite satisfaction, pending registrations, and semester restrictions. Table 1 summarizes the primary characteristics of the evaluated dataset.

Table 1. Dataset Characteristics

Dataset Characteristic	Value
Database Engine	MySQL 5.7.44
PHP Version	8.1.34
Students	1,833
Courses	960
Academic Grade Records	165,282
Historical Coverage	1990s – Present
Core Procedure	GetAvailableCourses
Optimization Technique	RTTS

The primary academic-history repository (student_enrollment_grades1) contains 165,282 grade records representing multiple generations of students. Rather than scanning the entire repository during each execution, the GetAvailableCourses procedure extracts student-specific academic histories and constructs temporary datasets used for prerequisite validation, failed-course analysis, pending-course verification, and course eligibility determination.

This historical dataset provides a realistic production environment for evaluating temporary table behavior and optimizer-related execution failures in MySQL 5.7 stored procedures.

3. Related Work

Previous studies and technical reports have documented limitations in MySQL temporary table handling.

MySQL Bug #12198 reported failures involving temporary table aliasing inside stored procedures.

MySQL Bug #72318 documented reopening restrictions associated with temporary tables referenced multiple times within execution plans.

Schwartz et al. discussed temporary-table performance considerations in *High Performance MySQL*.

Winand highlighted optimizer behavior and execution-plan complexity in *SQL Performance Explained.*

However, existing literature focuses primarily on identifying the limitation rather than proposing practical mitigation strategies for production registration systems operating on MySQL 5.7.

4. Problem Description

The original GetAvailableCourses procedure created temporary tables to store student enrollment history and pending-course information. These temporary datasets were subsequently reused across multiple joins, subqueries, and prerequisite validation operations.

Example:

CREATE TEMPORARY TABLE tmp_student_courses AS

SELECT ...

The same temporary table was later referenced in:

LEFT JOIN operations

NOT EXISTS subqueries

prerequisite validation queries

academic-rule filtering operations

This design generated intermittent runtime failures, including:

ERROR 1137: Can't reopen table

and

ERROR 1146: Table doesn't exist

5. Root Cause Analysis

Analysis of the original implementation revealed repeated reuse of the same temporary table across independent execution paths. The optimizer occasionally attempted to reopen the same temporary object during execution planning.

The issue became more pronounced as business rules increased and additional validation branches were added.

6. Proposed RTTS Strategy

6.1 Concept

The proposed Replicated Temporary Table Strategy (RTTS) creates synchronized replicas of temporary tables and assigns each replica to a dedicated execution branch. Instead of repeatedly accessing a single temporary table, multiple replicas are generated immediately after construction.

Example:

```
CREATE TEMPORARY TABLE tmp_student_courses2
AS
SELECT * FROM tmp_student_courses;
```

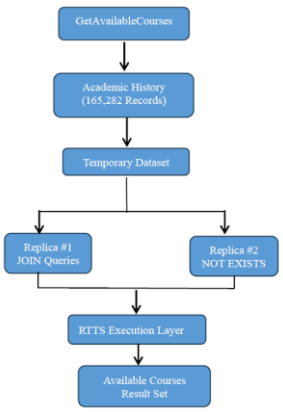
Similarly:

```
CREATE TEMPORARY TABLE tmp_Downloaded_Courses2
AS
SELECT * FROM tmp_Downloaded_Courses;
```

These replicated structures prevent the optimizer from reopening the same temporary object through different execution paths.

6.2 RTTS Architecture

Figure 2. Proposed RTTS Architecture for Temporary Table Reopening Mitigation in MySQL 5.7.



7. Algorithm Design

Algorithm 1: Original Procedure

Create tmp_student_courses

Use tmp_student_courses in JOIN

Reuse tmp_student_courses in NOT EXISTS

Reuse tmp_student_courses in prerequisite validation

Execute registration filtering

Return available courses

Weaknesses

Repeated temporary-table access

Optimizer reopening conflicts

Runtime instability

Algorithm 2: RTTS Procedure

Create tmp_student_courses

Create tmp_student_courses2

Create tmp_Downloaded_Courses

Create tmp_Downloaded_Courses2

Assign JOIN operations to replica 1

Assign prerequisite validation to replica 2

Assign pending-course validation to dedicated replica

Execute academic rules

Return available courses

Advantages

Independent execution paths

No temporary-table reopening

Stable optimizer behavior

Improved maintainability

8. Experimental Evaluation

The proposed RTTS strategy was deployed in the production implementation of GetAvailableCourses.

The evaluation focused on:

execution reliability,

runtime stability,

maintainability,

and procedural complexity.

Metric	Original Design	RTTS Design
Temporary Table Copies	1	Multiple
Runtime Failures	Observed	Not Observed
Reopening Errors	Present	Eliminated
Maintainability	Moderate	Improved
Query Stability	Low	High

The redesigned implementation completed registration operations without reproducing previously observed reopening failures.

9. Discussion

The findings indicate that temporary table reopening limitations remain a significant concern in legacy MySQL environments.

Although upgrading to MySQL 8 or redesigning procedures using CTEs may eliminate some limitations, many institutions continue to operate legacy deployments where such migrations are impractical.

RTTS offers a lightweight solution requiring minimal modifications to existing business logic while preserving compatibility with MySQL 5.7.

10. Threats to Validity

This study is based on a single academic registration system deployed in a higher-education environment.

Results may vary depending on:

optimizer configuration,

workload characteristics,

hardware resources,

and database schema design.

Furthermore, RTTS is intended primarily for legacy MySQL environments and may not provide additional benefits in database engines with more advanced temporary-object management.

11. Conclusion

This paper presented a real-world case study involving temporary table reopening failures in the GetAvailableCourses procedure of an academic registration system.

The study identified repeated temporary-table reuse as the primary cause of runtime failures and proposed the Replicated Temporary Table Strategy (RTTS) as a practical mitigation approach.

Deployment results demonstrated stable execution and elimination of observed reopening failures while preserving compatibility with MySQL 5.7.44.

Future work will investigate automated temporary-table replication techniques and comparative evaluation against MySQL 8 CTE-based implementations.

References

Oracle MySQL Bug #12198. *Temporary Table Aliasing does not work inside stored procedure*.

Oracle MySQL Bug #72318. *Can't reopen temporary table with aliases in stored routines*.

Schwartz, B., Zaitsev, P., & Tkachenko, V. *High Performance MySQL*. O'Reilly.

Winand, M. *SQL Performance Explained*.

Silberschatz, A., Korth, H., & Sudarshan, S. *Database System Concepts*.

Gray, J., & Reuter, A. *Transaction Processing: Concepts and Techniques*.

Oracle Corporation. *MySQL 5.7 Reference Manual*.

Connolly, T., & Begg, C. *Database Systems: A Practical Approach to Design, Implementation, and Management*.

Ramakrishnan, R., & Gehrke, J. *Database Management Systems*.

Date, C. J. *An Introduction to Database Systems*.