开源软件基础结课论文

对 PostgreSQL 的分析

作者姓名：王虹景，李霄冲

指导教师：任志磊

学位类别：工学学士

专　　业：软件工程

学院（系）：大连理工大学

2026 年 2 月

Analysis of PostgreSQL

A thesis submitted to

University of Chinese Academy of Sciences

in partial fulfillment of the requirement

for the degree of

Bachelor of Engineering

in Software Engineering

By

Wang Hongjing

Supervisor: Ren Zhilei

Dalian University of Technology

February, 2026

摘 要

本项目围绕 PostgreSQL master 分支（2022-2025）的提交历史，构建“提交行为统计—Bug 修复识别—静态结构演化—动态执行追踪—形式化验证”的多层分析链路，系统刻画 Bug 修复行为规律与代码演化特征，并对关键修复进行可验证性证明。在提交层面，基于规则化标注与时间/作者/模块统计，发现修复节奏与发布周期相关，高风险模块（storage/replication/WAL 等）长期高占比，且核心开发者承担关键模块的修复工作。在静态结构层面，Bug 修复呈现出一致的结构性演化模式：前置条件显式化、控制流分裂、错误路径显式化与条件语义增强等，体现防御式编程的系统性强化。在动态分析层面，通过 PySnooper 追踪 PostgreSQL Python 工具脚本，定位出配置同步脚本的“静默逻辑失效”，并通过回归测试验证修复有效性，建立了从异常发现到验证的闭环流程。在形式化验证层面，使用 Z3 对该修复进行约束建模，证明原逻辑存在反例（SAT），修复后逻辑在规约域内无反例（UNSAT），从而实现“证明修复有效”的强结论。整体而言，本研究以统计分析、结构分析、动态追踪与形式化建模互证，为理解大型系统的 Bug 修复规律与提升修复可信度提供了可复用的方法框架。

关键词：PostgreSQL，提交历史分析，Bug 修复识别，代码结构演化，动态追踪，形式化验证

Abstract

This thesis builds a multi-layer analysis pipeline over the PostgreSQL master branch commit history (2022–2025), integrating commit-level statistics, bug-fix identification, static structural evolution analysis, dynamic execution tracing, and formal verification. At the commit level, using rule-based labeling and statistics by time, author, and subsystem, we observe that fix activity correlates with release cycles; high-risk subsystems (e.g., storage, replication, and WAL) consistently account for a large share of fixes, and core developers contribute disproportionately to critical-module maintenance. At the static level, bug fixes exhibit recurring structural evolution patterns, including explicit preconditions, control-flow splitting, explicit error paths, and enriched conditional semantics, reflecting systematic strengthening of defensive programming. At the dynamic level, we trace PostgreSQL Python utility scripts with PySnooper to pinpoint a silent logic failure in a configuration synchronization script, validate the fix via regression tests, and close the loop from anomaly discovery to verification. Finally, we model the fix in Z3: the original logic admits counterexamples (SAT), whereas the repaired logic has no counterexample within the specification domain (UNSAT), yielding a strong, proof-backed validity claim. Overall, the proposed framework triangulates statistical analysis, structural analysis, dynamic tracing, and formal modeling to provide reusable methods for understanding bug-fix behaviors in large systems and improving the trustworthiness of fixes.

Keywords: PostgreSQL, commit history mining, bug-fix identification, structural evolution, dynamic tracing, formal verification

目 录

第 1 章 绪论 ····· 1
1.1 研究背景与问题定义 ····· 1
1.2 研究目标与贡献 ····· 1
1.3 研究方法与技术路线 ····· 1
1.4 论文结构 ····· 1
第 2 章 数据集构建与 Bug 修复识别 ····· 2
2.1 PostgreSQL 提交数据范围与来源（2022–2025） ····· 2
2.2 Bug 修复提交识别规则（关键词/结构/测试信号） ····· 2
2.3 标注流程与质量控制（Y/M/N 与复核机制） ····· 2
2.4 描述性统计（时间、作者、模块、规模） ····· 2
第 3 章 Bug 修复结构演化分析（静态） ····· 3
3.1 AST 与 libcst 分析方法 ····· 3
3.2 典型修复模式：前置条件显式化与错误路径显式化 ····· 3
3.3 控制流与条件复杂度变化 ····· 3
3.4 静态分析小结 ····· 3
第 4 章 动态执行轨迹与案例复现 ····· 4
4.1 动态分析对象与实验环境 ····· 4
4.2 generate_editorconfig.py 的异常定位（Silent Failure） ····· 4
4.3 修复策略与回归测试结果 ····· 4
4.4 动态证据对静态结论的补充价值 ····· 4
第 5 章 形式化建模与修复正确性验证 ····· 5
5.1 Z3 建模对象与状态抽象 ····· 5
5.2 修复前模型：反例可满足性（SAT） ····· 5
5.3 修复后模型：反例不可满足性（UNSAT） ····· 5
5.4 从“测试发现”到“证明正确”的方法提升 ····· 5
第 6 章 综合讨论 ····· 6
6.1 三类证据链（静态–动态–形式化）的互证关系 ····· 6
6.2 高风险模块治理启示（storage/replication/WAL） ····· 6
6.3 方法局限与有效性威胁 ····· 6

第 7 章　结论与未来工作 ····· 7
7.1 主要研究结论 ····· 7
7.2 工程实践建议 ····· 7
7.3 后续工作（自动检测、自动修复、CI 集成） ····· 7
附录 A　大学学位论文撰写要求 ····· 8
A.1 论文无附录者无需附录部分 ····· 8
A.2 测试公式编号 $\Lambda, \lambda, \theta, \bar{\Lambda}, \sqrt{S_{NN}}$ ····· 8
A.3 测试生僻字 ····· 9
参考文献 ····· 10
致谢 ····· 10
作者简历及攻读学位期间发表的学术论文与研究成果 ····· 11

符号列表

字符

Symbol	Description	Unit
R	the gas constant	$\mathrm{m}^2 \cdot \mathrm{s}^{-2} \cdot \mathrm{K}^{-1}$
C_v	specific heat capacity at constant volume	$\mathrm{m}^2 \cdot \mathrm{s}^{-2} \cdot \mathrm{K}^{-1}$
C_p	specific heat capacity at constant pressure	$\mathrm{m}^2 \cdot \mathrm{s}^{-2} \cdot \mathrm{K}^{-1}$
E	specific total energy	$\mathrm{m}^2 \cdot \mathrm{s}^{-2}$
e	specific internal energy	$\mathrm{m}^2 \cdot \mathrm{s}^{-2}$
h_T	specific total enthalpy	$\mathrm{m}^2 \cdot \mathrm{s}^{-2}$
h	specific enthalpy	$\mathrm{m}^2 \cdot \mathrm{s}^{-2}$
k	thermal conductivity	$\mathrm{kg} \cdot \mathrm{m} \cdot \mathrm{s}^{-3} \cdot \mathrm{K}^{-1}$
S_{ij}	deviatoric stress tensor	$\mathrm{kg} \cdot \mathrm{m}^{-1} \cdot \mathrm{s}^{-2}$
τ_{ij}	viscous stress tensor	$\mathrm{kg} \cdot \mathrm{m}^{-1} \cdot \mathrm{s}^{-2}$
δ_{ij}	Kronecker tensor	1
I_{ij}	identity tensor	1

算子

Symbol	Description
Δ	difference
∇	gradient operator
$\delta^{\pm}$	upwind-biased interpolation scheme

缩写

CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy
EOS	Equation of State

JWL	Jones-Wilkins-Lee
WENO	Weighted Essentially Non-oscillatory
ZND	Zel'dovich-von Neumann-Doering

第 1 章　绪论

1.1　研究背景与问题定义

本文聚焦 PostgreSQL 在 2022–2025 年间的提交演化，关注其中具有代表性的 Bug 修复行为。研究问题是：在真实工程中，Bug 修复是否存在可归纳的结构模式，以及这些模式能否通过静态、动态和形式化三类证据被一致支持。

1.2　研究目标与贡献

本文的目标是构建一个可复现的数据与分析流程，并围绕“识别–分析–验证”形成完整证据链。主要贡献包括：

- 构建 PostgreSQL Bug 修复样本集，并建立可执行的标注规范；
- 提炼修复结构演化规律，分析控制流与条件复杂度变化；
- 在典型案例上结合动态执行轨迹与形式化约束验证修复正确性。

1.3　研究方法与技术路线

技术路线分为三层：

1. 数据层：提交抓取、规则识别、人工复核与统计分析；
2. 证据层：AST/libcst 静态分析与运行轨迹复现；
3. 验证层：基于 Z3 的状态抽象与可满足性检验。

1.4　论文结构

全文共七章：第二章介绍数据集和识别流程；第三章给出静态结构演化结果；第四章给出动态案例复现；第五章进行形式化验证；第六章综合讨论三类证据及治理启示；第七章总结并展望。

第 2 章　数据集构建与 Bug 修复识别

2.1　PostgreSQL 提交数据范围与来源（2022–2025）

本章数据来源于 PostgreSQL 官方代码仓库提交记录，时间窗口设定为 2022 年 1 月至 2025 年 12 月。通过统一脚本抽取提交元数据、变更文件、提交说明和测试相关信号，形成原始样本池。

2.2　Bug 修复提交识别规则（关键词/结构/测试信号）

识别规则由三类特征组成：

- 文本关键词：如 fix、bug、regression；
- 结构信号：条件判断新增、错误分支增强、边界检查补全；
- 测试信号：回归测试新增或断言强化。

2.3　标注流程与质量控制（Y/M/N 与复核机制）

采用三级标注：Y（明确 Bug 修复）、M（疑似修复）、N（非修复）。流程包括初标、交叉复核和冲突仲裁，确保标注一致性。针对争议样本保留复核意见与决议依据。

2.4　描述性统计（时间、作者、模块、规模）

统计维度覆盖时间分布、作者贡献、模块归属（如 storage/replication/WAL）和变更规模。通过描述性统计识别高风险区域与修复密集阶段，为后续章节分析提供采样基础。

第 3 章　Bug 修复结构演化分析（静态）

3.1　AST 与 libcst 分析方法

本章使用 AST 与 libcst 双路径建模代码变更：AST 强调语义结构，libcst 保留语法细节。通过前后版本对齐提取语句级与表达式级修复动作。

3.2　典型修复模式：前置条件显式化与错误路径显式化

从样本中提炼两类高频模式：

1. 前置条件显式化：补充空值、范围、状态有效性检查；
2. 错误路径显式化：新增异常分支、错误码返回或日志说明。

3.3　控制流与条件复杂度变化

通过控制流图（CFG）和条件表达式复杂度指标分析修复前后差异，考察分支数量、嵌套深度和路径可达性变化，评估修复对可维护性与可验证性的影响。

3.4　静态分析小结

静态证据表明，大多数 Bug 修复并非“局部语句替换”，而是“条件与错误路径的可解释性增强”。该结论将在后续动态与形式化章节中继续验证。

第 4 章　动态执行轨迹与案例复现

4.1　动态分析对象与实验环境

本章选择具有代表性的修复样本进行动态复现，记录输入、路径、日志与输出状态。实验环境包含固定版本解释器、依赖与测试脚本，以保证结果可重复。

4.2　generate_editorconfig.py 的异常定位（Silent Failure）

针对 generate_editorconfig.py 的 Silent Failure 问题，构造触发输入并复现实验。通过执行轨迹定位异常传播链路，识别“错误被吞没但无显式告警”的关键节点。

4.3　修复策略与回归测试结果

依据定位结果提出修复策略：补充显式检查、增强异常分支与日志，并将修复纳入回归测试集合。测试结果显示目标缺陷可稳定复现与消除，且无新增失败用例。

4.4　动态证据对静态结论的补充价值

动态分析补充了静态分析难以覆盖的运行时上下文，证明“前置条件显式化”和“错误路径显式化”能够在真实执行路径上降低静默失败风险。

第 5 章　形式化建模与修复正确性验证

5.1　Z3 建模对象与状态抽象

本章将关键修复逻辑抽象为可判定约束系统，使用 Z3 对输入状态、路径条件和错误触发条件建模，建立“修复前/修复后”可比较模型。

5.2　修复前模型：反例可满足性（SAT）

在修复前模型中，求解器可找到满足约束的反例（SAT），说明存在输入组合会触发异常行为，验证缺陷确实存在且可形式化描述。

5.3　修复后模型：反例不可满足性（UNSAT）

在修复后模型中，将同类反例目标加入约束后得到 UNSAT，表明对应缺陷路径在约束空间内已不可达，为修复有效性提供证明级证据。

5.4　从“测试发现”到“证明正确”的方法提升

与仅依赖测试相比，形式化验证可覆盖更多输入空间并给出可解释结论，实现从“经验性发现问题”到“可证明修复正确”的方法学升级。

第 6 章　综合讨论

6.1　三类证据链（静态–动态–形式化）的互证关系

静态分析回答“改了什么结构”，动态分析回答“运行时发生了什么”，形式化分析回答“是否仍存在可达反例”。三者构成互补证据链，提升结论可信度。

6.2　高风险模块治理启示（storage/replication/WAL）

针对高风险模块（storage/replication/WAL），建议建立分层治理机制：高频提交监测、关键路径断言增强、回归用例优先配置以及形式化检查前置。

6.3　方法局限与有效性威胁

本文方法存在若干局限：样本时间窗口有限、标注仍受人工判断影响、形式化模型存在抽象损失。外部有效性方面，不同项目的开发流程差异可能影响结论迁移。

第 7 章　结论与未来工作

7.1　主要研究结论

本文围绕 PostgreSQL Bug 修复演化给出系统研究：构建数据集与识别规则，提炼静态修复模式，完成动态复现，并通过 Z3 提供修复正确性验证。

7.2　工程实践建议

建议在工程流程中落地三项实践：

- 将修复识别与模块风险统计纳入例行质量看板；
- 将动态回归与关键路径日志作为发布前强制检查；
- 对高风险逻辑引入轻量级形式化验证。

7.3　后续工作（自动检测、自动修复、CI 集成）

后续可从三方面扩展：基于历史修复模式训练自动检测器；探索面向条件与错误路径的自动修复策略；将静态、动态和形式化检查进一步集成到 CI 流水线。

附录 A　大学学位论文撰写要求

学位论文是研究生科研工作成果的集中体现，是评判学位申请者学术水平、授予其学位的主要依据，是科研领域重要的文献资料。根据《科学技术报告、学位论文和学术论文的编写格式》（GB/T 7713-1987）、《学位论文编写规则》（GB/T 7713.1-2006）和《文后参考文献著录规则》（GB7714—87）等国家有关标准与实际情况，特制订本规定。

A.1　论文无附录者无需附录部分

A.2　测试公式编号 $\Lambda, \lambda, \theta, \bar{\Lambda}, \sqrt{S_{NN}}$

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{V}) = 0 \\ \frac{\partial (\rho \boldsymbol{V})}{\partial t} + \nabla \cdot (\rho \boldsymbol{V}\boldsymbol{V}) = \nabla \cdot \boldsymbol{\sigma} \\ \frac{\partial (\rho E)}{\partial t} + \nabla \cdot (\rho E \boldsymbol{V}) = \nabla \cdot (k \nabla T) + \nabla \cdot (\boldsymbol{\sigma} \cdot \boldsymbol{V}) \end{cases} \quad \ldots (A.1)$$

$$\frac{\partial}{\partial t} \int\limits_{\Omega} u \, \mathrm{d}\Omega + \int\limits_{S} \mathbf{n} \cdot (u \boldsymbol{V}) \, \mathrm{d}S = \dot{\phi} \quad \ldots (A.2)$$

$$\mathcal{L}\{f\}(s) = \int_{0^-}^{\infty} f(t) e^{-st} \, \mathrm{d}t, \ \mathscr{L}\{f\}(s) = \int_{0^-}^{\infty} f(t) e^{-st} \, \mathrm{d}t$$

$$\mathcal{F}\big(f(x + x_0)\big) = \mathcal{F}\big(f(x)\big) e^{2\pi i \xi x_0}, \ \mathscr{F}\big(f(x + x_0)\big) = \mathscr{F}\big(f(x)\big) e^{2\pi i \xi x_0}$$

mathtext: $A, F, L, 2, 3, 5, \sigma$, mathnormal: $A, F, L, 2, 3, 5, \sigma$, mathrm: $\mathrm{A, F, L, 2, 3, 5, \sigma}$.

mathbf: $\mathbf{A, F, L, 2, 3, 5, \sigma}$, mathit: $\mathrm{A, F, L, 2, 3, 5, \sigma}$, mathsf: $\mathsf{A, F, L, 2, 3, 5, \sigma}$.

mathtt: $\mathtt{A, F, L, 2, 3, 5, \sigma}$, mathfrak: $\mathfrak{A}, \mathfrak{F}, \mathfrak{L}, 2, 3, 5, \sigma$, mathbb: $\mathbb{A}, \mathbb{F}, \mathbb{L}, \mathbb{2}, \mathbb{3}, \mathbb{5}, \sigma$.

mathcal: $\mathcal{A}, \mathcal{F}, \mathcal{L}, 2, 3, 5, \sigma$, mathscr: $\mathscr{A}, \mathscr{F}, \mathscr{L}, 2, 3, 5, \sigma$, boldsymbol: $\boldsymbol{A}, \boldsymbol{F}, \boldsymbol{L}, 2, 3, 5, \boldsymbol{\sigma}$.

vector: $\boldsymbol{\sigma}, \boldsymbol{T}, \boldsymbol{a}, \boldsymbol{F}, \boldsymbol{n}$, unitvector: $\boldsymbol{\sigma}, \mathbf{T}, \mathbf{a}, \mathbf{F}, \mathbf{n}$

matrix: $\boldsymbol{\sigma}, \mathbf{T}, \mathbf{a}, \mathbf{F}, \mathbf{n}$, unitmatrix: $\boldsymbol{\sigma}, \mathbf{T}, \mathbf{a}, \mathbf{F}, \mathbf{n}$

tensor: $\boldsymbol{\sigma}, \mathsf{T}, \mathsf{a}, \mathsf{F}, \mathsf{n}$, unittensor: $\boldsymbol{\sigma}, \mathsf{T}, \mathsf{a}, \mathsf{F}, \mathsf{n}$

A.3　测试生僻字

霜蟾盥薇曜灵霜颸妙鬘虚霏淩澌菀枯菡萏泬寥窅冥磑毰毸濩落雪雪便嬛岧峣瀺灂姽婳愔嫕飒纚棽俪緾冤莩甲摛藻卮言倥侗椒觞期颐夜阑彬蔚倥偬澄廓簪缨陟遐迤逦缥缃鹣鲽憯懔闺闼璀错媕婀噌吰澒洞阛阓覼缕玓瓅逡巡諓諓琭琭瀌瀌踽踽叆叇氤氲瓠犀流眄蹀躞赟嬛茕頔璎珞螓首蘅皋惏悷缱绻昶皴皱颟顸愀然菡萏卑陬纯懿犇麤掱暒墌墍墎墏墐墒墒墓墔墕墖墘墖墚墛坠墝增墠墡墢墣墤墥墦墧墨墩墪樽墬墭堕墯墰墱墲坟墴墵垯墷墸墹墺墙墼墽垦墿壀壁壂壃壄壅壆坛壈壉壊垱壌壍埙壏壐壑壒压壔壕壖壗垒圹垆壛壜壝垄壟坜壣壤壥壦壧壨坝塆圭嫶嫷嫸嫹嫺娴嫼嫽嫾婳妫嬁嬂嬃嬄嬅嬆嬇娆嬉嬊娇嬍嬎嬏嬐嬑嬒嬓嬔嬕嬖嬗嬘嬙嬚嬛嬜嬞嬟嬠嫒嬢嬣嬥嬦嬧嬨嬩嫔嬫嬬奶嬬嬮嬯嬰嬱嬲嬳嬴嬵嬶嬷婶嬹嬺嬻嬼嬽嬾嬿孀孁孂娘孄孅孆孇孆孈孉孊娈孋孊孍孎孏嫫婿媚嵭嵮嵯嵰嵱嵲嵳嵴嵵嵶嵷嵸嵹嵺嵻嵼嵽嵾嵿嶀嵝嶂嶃崭嶅嶆岖嶈嶉嶊嶋嶌嶍嶎嶏嶐嶑嶒嶓嵚嶕嶖嶘嶙嶚嶛嶜嶝嶞嶟峤嶡峣嶣嶤嶥嶦峄峃嶩嶪嶫嶬嶭崄嶯嶰嶱嶲嶳岙嶵嶶嶷嵘嶹岭嶻屿岳帋巀巁巂巃巄巅巆巇巈巉巊岿巌巍巎巏巐巑峦巓巅巕岩巗巘巙巚帠帡帢帣帤帨帩帪帬带帯帱帲帴帵帷帹帺帻帼帽帾帿幁幂帏幄幅幆幇幈幉幊幋幌幍幎幏幐微幒幓幖幙幚幛幜幝幞帜幠幡幢币幤幥幦幧幨幩幪幭幮幯幰庁庎庑庖庘庛庝庠庡庢庣庤庥庨庩庪庬庮庯庰庱庲庳庴庵庹庺庻庼庽庿廀厕廃厩廅廆廇廋廌廍廎廏廐廑廒廔廕廖廗廘廙廛廜廞庑廥廦廧廨廭廮廯廰痈廲廵廸廹廻廼廽廿弁弃弅弆弇弉弖弙弚弜弝弞弡弢弣弤弨弩弪弫弬弭弮弰弲弪弴弶弸弻弽弿彖彗彘彚彛彜彝彞彟彴彵彶彷彸役彺彻彽彾佛徂徃徆徇徉后徍徎徏径徒従徔徕徖徙徚徛徜徝从徟徕御徢徣徤徥徦徧徨复循徫旁徭微徯徰徱徲徳徴徵徶德徸彻徺徻忂惔愔忇忈忉忔忕忖忚忛応忝忞忟忪挣挦挧挨挩挪挫挬挭挮挰掇授掉掊掋掍掎掐掑排掓掔掕掗掚挂掜掝掞掟掠采探掣掤掦措掫掬掭掮掯掰掱掲掳掴掵掶掸掹掺掻掼掽掾掿拣揁揂揃揅揄揆揇揈揉揊揋揌揑揓揔揕揗揘揙揜揝揞揟揠揢揤揥擎捂揰揱揲揳援揵揶揷揸揻揾揿搀搁搂搃搄搅搆搇搈搉搊搋搌搎搏搐搑搒搓搔搸擦揪搟搪摛掼搋摆搃搌斫斩斮斱斲斳斴斵斶斸叶旫旮旯晒晓晔晕晖晗晘晙晛晜晞晟晠晡晣晣晤晥晦晧晪晫晬晭晰晲晳晴晵晷晸晹晻晼晽晾晿暀暁暂暃暄暅暆暇晕晖暊暋暌暍暎暏暐暑暒暓暔暕暖暗旸暙暚暛暜暝暞暟暠暡暣暤暥暦暧暨暩暪暬暭暮暯暰昵暲暳暴暵

致　谢

感激 casthesis 作者吴凌云学长，gbt7714-bibtex-style 开发者 zepinglee，和 ctex 众多开发者们。若没有他们的辛勤付出和非凡工作，LaTeX 菜鸟的我是无法完成此国科大学位论文 LaTeX 模板 ucasthesis 的。在 LaTeX 中的一点一滴的成长源于开源社区的众多优秀资料和教程，在此对所有 LaTeX 社区的贡献者表示感谢！

作者简历及攻读学位期间发表的学术论文与研究成果

本科生无需此部分。

作者简历：

已发表（或正式接受）的学术论文：

申请或已获得的专利：

参加的研究项目及获奖情况：

[plain]