# PostgreSQLのバグを見つけた！ そんな時どうする？

澤田　雅彦
日本PostgreSQLユーザ会
2023/7/20 OSC2023 Online/Kyoto

# 自己紹介

- 澤田　雅彦 @masahiko_sawada
- PostgreSQL Major Contributor
- PostgreSQL Committer
- レプリケーション、Vacuum周りの新機能開発
- バグ修正

# 注意事項

- PostgreSQLに実在したバグを用いて、バグ発見→バグ報告/修正までの流れを解説しますが、バグはすでに修正済みです
  - 12.9、13.5、14.1にて修正済み(2021年リリース)
  - 15.0以降でも発生しません
- PostgreSQLは最新バージョンを使いましょう
- 実際にバグが発見・修正された流れとは異なります（オリジナルストーリーです）

# 登場するバグについて

- ALTER TABLE … RENAME TO文に関するバグ
- すでに修正済みです(12.9, 13.5, 14.1以降)
- PostgreSQL 12.8を使ってバグを見ていきます

# ALTER TABLE … RENAME TOの変わった仕様

```
=# \d test_tbl
              Table "public.test_tbl"
 Column |  Type   | Collation | Nullable | Default
--------+---------+-----------+----------+---------
 a      | integer |           |          |
Indexes:
    "test_tbl_idx" btree (a)

=# ALTER TABLE test_tbl_idx RENAME TO test_tbl_idx_new;
ALTER TABLE
=# ALTER INDEX test_tbl RENAME TO test_tbl_new;
ALTER INDEX
=# \d test_tbl_new
             Table "public.test_tbl_new"
 Column |  Type   | Collation | Nullable | Default
--------+---------+-----------+----------+---------
 a      | integer |           |          |
Indexes:
    "test_tbl_idx_new" btree (a)
```

# ALTER TABLE … RENAME TOの変わった仕様

- ALTER TABLE … RENAME TOは、「ALTER XXXのXXX」とオブジェクトの種類が一致していなくてもOK
  - 例）ALTER INDEX … RENAME TOで、テーブル名を変更できる
- この動作自体はバグではない

# RENAME中にINSERTはできる？

```
-- Session-1
=# BEGIN;
BEGIN

=# ALTER TABLE test_tbl RENAME TO test_tbl_new;
ALTER TABLE









=# COMMIT;
```

```
-- Session-2
=# BEGIN;
BEGIN

=# INSERT INTO test_tbl VALUES (1);




ERROR:  relation "test_tbl" does not exist
LINE 1: INSERT INTO test_tbl VALUES (1);
                    ^
```

# RENAME中にINSERTはできる？

```
-- Session-1
=# BEGIN;
BEGIN

=# ALTER TABLE test_tbl RENAME TO test_tbl_new;
ALTER TABLE









=# COMMIT;
```

あるトランザクションで、テーブル名を変更

```
-- Session-2
=# BEGIN;
BEGIN

=# INSERT INTO test_tbl VALUES (1);




ERROR:  relation "test_tbl" does not exist
LINE 1: INSERT INTO test_tbl VALUES (1);
                    ^
```

# RENAME中にINSERTはできる？

```
-- Session-1
=# BEGIN;
BEGIN

=# ALTER TABLE test_tbl RENAME TO test_tbl_new;
ALTER TABLE


=# COMMIT;
```

別のトランザクションで、
そのテーブルにINSERT。
でも、待たされる。

```
-- Session-2
=# BEGIN;
BEGIN

=# INSERT INTO test_tbl VALUES (1);



ERROR:  relation "test_tbl" does not exist
LINE 1: INSERT INTO test_tbl VALUES (1);
                    ^
```

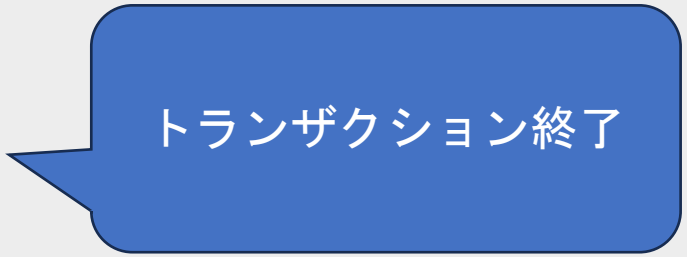
# RENAME中にINSERTはできる？

```
-- Session-1
=# BEGIN;
BEGIN

=# ALTER TABLE test_tbl RENAME TO test_tbl_new;
ALTER TABLE




=# COMMIT;
```




```
-- Session-2
=# BEGIN;
BEGIN

=# INSERT INTO test_tbl VALUES (1);




ERROR:  relation "test_tbl" does not exist
LINE 1: INSERT INTO test_tbl VALUES (1);
                    ^
```

# RENAME中にINSERTはできる？

```
-- Session-1
=# BEGIN;
BEGIN

=# ALTER TABLE test_tbl RENAME TO test_tbl_new;
ALTER TABLE




=# COMMIT;
```

INSERT処理が続行できるが、
すでにテーブル名が変わっているので、
見つからずエラー

```
-- Session-2
=# BEGIN;
BEGIN

=# INSERT INTO test_tbl VALUES (1);



ERROR:  relation "test_tbl" does not exist
LINE 1: INSERT INTO test_tbl VALUES (1);
                    ^
```

# ロックレベル

- ALTER TABLE … RENAME TOはテーブルにAccessExclusiveLockを取得する
- INSERTはテーブルにRowExclusiveLockを取得する
- Session-2（INSERTするTx）はテーブルロックを取得する段階（INSERTを実施する前）で待たされていた
- Session-1はCOMMIT時に保持していたロックを開放

| 要求するロックモード | 既存のロックモード | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | ACCESS SHARE | ROW SHARE | ROW EXCL. | SHARE UPDATE EXCL. | SHARE | SHARE ROW EXCL. | EXCL. | ACCESS EXCL. |
| ACCESS SHARE | | | | | | | | X |
| ROW SHARE | | | | | | | X | X |
| ROW EXCL. | | | | | X | X | X | X |
| SHARE UPDATE EXCL. | | | | X | X | X | X | X |
| SHARE | | | X | X | | X | X | X |
| SHARE ROW EXCL. | | | X | X | X | X | X | X |
| EXCL. | | X | X | X | X | X | X | X |
| ACCESS EXCL. | X | X | X | X | X | X | X | X |

https://www.postgresql.jp/document/15/html/explicit-locking.html

# RENAME TO中のINSERTが成功する…？

```
-- Session-1
=# BEGIN;
BEGIN

=# ALTER INDEX test_tbl RENAME TO test_tbl_new;
ALTER TABLE









=# COMMIT;
```

```
-- Session-2
=# BEGIN;
BEGIN

=# INSERT INTO test_tbl VALUES (1);
INSERT 1

=# COMMIT;
COMMIT
```

# RENAME TO中のINSERTが成功する…？

- RENAME TOに限ってはALTER TABLEもALTER INDEXも動作は同じはず
- しかし、ALTER INDEXにした時はRENAME中にINSERTが成功した

# なぜINSERTが成功する？

- INSERTが成功する＝INSERTが待たされていない
- ロックレベルが関係してそう
- pg_locksビューで確認してみる

# なぜINSERTが成功する？

- test_tblに対してShareUpdateExclusiveLockを取得していることがわかる
- test_tblのOIDは16397

```
=# SELECT locktype, relation, mode, granted FROM pg_locks WHERE pid = 2764118;
   locktype    | relation |           mode            | granted
---------------+----------+---------------------------+---------
 virtualxid    |          | ExclusiveLock             | t
 relation      |    16397 | ShareUpdateExclusiveLock | t
 transactionid |          | ExclusiveLock             | t
(3 rows)

=# SELECT 16397::regclass;
 regclass
----------
 test_tbl
(1 row)
```

# DDLとロック（再掲）

- ALTER TABLE … RENAME TOはテーブルにAccessExclusiveLockを取得する
- INSERTはテーブルにRowExclusiveLockを取得する
- Session-2（INSERTするTx）はテーブルロックを取得する段階（INSERTを実施する前）で待たされていた
- Session-1はCOMMIT時に保持していたロックを開放

| 要求するロックモード | 既存のロックモード | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | ACCESS SHARE | ROW SHARE | ROW EXCL. | SHARE UPDATE EXCL. | SHARE | SHARE ROW EXCL. | EXCL. | ACCESS EXCL. |
| ACCESS SHARE | | | | | | | | X |
| ROW SHARE | | | | | | | X | X |
| ROW EXCL. | | | | | X | X | X | X |
| SHARE UPDATE EXCL. | | | | X | X | X | X | X |
| SHARE | | | X | X | | X | X | X |
| SHARE ROW EXCL. | | | X | X | X | X | X | X |
| EXCL. | | X | X | X | X | X | X | X |
| ACCESS EXCL. | X | X | X | X | X | X | X | X |

https://www.postgresql.jp/document/15/html/explicit-locking.html

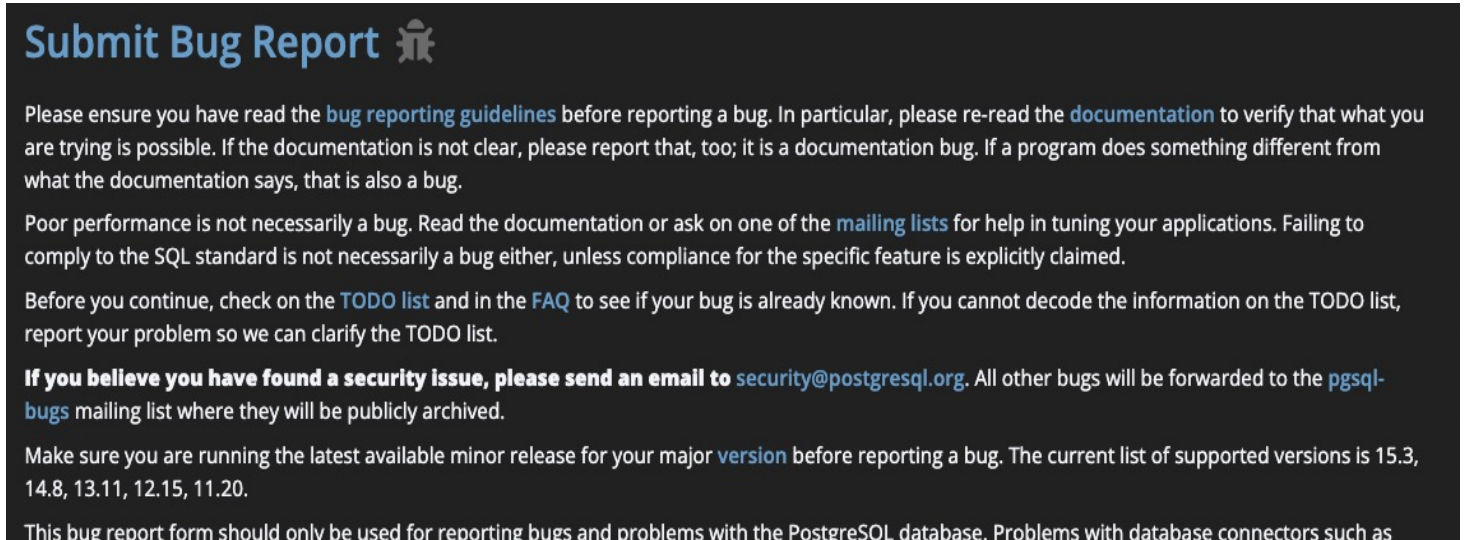
# ここまでのまとめ（バグ発見）

- ALTER XXX … RENAME TOは、オブジェクトの種類が異なっていても動く（これはOK）
- ALTER TABLE … RENAME TOは、テーブルに対してAccessExclusiveLockを取得する
  - これにより、INSERTやSELECTも同時に実行できない
- 一方で、ALTER INDEX … RENAME TOはテーブルに対して少し弱いロック（ShareUpdateExclusiveLock）を取得する
  - これにより、INSERTやSELECTが同時に実行可能

ALTERで指定するオブジェクトの種類によって、同じオブジェクトへの同じ操作でもロックレベルが異なる。

# バグを報告する

- コミュニティにバグを報告してみよう
- PostgreSQLにはバグ管理システムはない
- Submit Bug Report
  - https://www.postgresql.org/account/submitbug/



## Submit Bug Report

Please ensure you have read the bug reporting guidelines before reporting a bug. In particular, please re-read the documentation to verify that what you are trying is possible. If the documentation is not clear, please report that, too; it is a documentation bug. If a program does something different from what the documentation says, that is also a bug.

Poor performance is not necessarily a bug. Read the documentation or ask on one of the mailing lists for help in tuning your applications. Failing to comply to the SQL standard is not necessarily a bug either, unless compliance for the specific feature is explicitly claimed.

Before you continue, check on the TODO list and in the FAQ to see if your bug is already known. If you cannot decode the information on the TODO list, report your problem so we can clarify the TODO list.

**If you believe you have found a security issue, please send an email to** security@postgresql.org. All other bugs will be forwarded to the pgsql-bugs mailing list where they will be publicly archived.

Make sure you are running the latest available minor release for your major version before reporting a bug. The current list of supported versions is 15.3, 14.8, 13.11, 12.15, 11.20.

This bug report form should only be used for reporting bugs and problems with the PostgreSQL database. Problems with database connectors such as

# バグレポート

- 「○○となるべきなのに□□となる挙動を見つけました。これはバグですか？」だけでもOK
- バージョン、再現手順は必須
  - 必要に応じてサーバログ、スタックトレースや設定パラメータも
  - PostgreSQL開発者自身が問題を再現できることが重要
- 必須ではないけれど、もらえると嬉しい情報は、
  - どのバージョンで起こるか、起こらないか
  - どのコミットが原因か
  - どこのコードに原因があるか

# 本文の例

Hi,

I noticed that ‘ALTER TABLE <table> RENAME TO’ acquires an AccessExclusiveLock on the table whereas ‘ALTER INDEX <table> RENAME TO’ acquires a ShareUpdateExclusiveLock on the table. I think we should acquire AccessExclusiveLock on the table even if users uses ALTER INDEX to rename the table name.

(再現手順を載せる)

:

=# BEGIN;

=# ALTER INDEX tbl1 RENAME TO tbl2;

:

=# SELECT locktype, relation, mode, granted FROM pg_locks …

Regards,

# Thanks For Your Contribition!

- バグ報告ありがとうございます！
- pgsql-bugsメーリングリストにて議論が始まります
- 修正された場合は、コミットログに名前が載ります

```
commit e578c17d81662b4f4f63a2797bc1be64af3c8f93
Author: Michael Paquier <michael@paquier.xyz>
Date:   Mon Oct 12 20:34:55 2020 +0900

    Fix compilation warning in unicode_norm.c

    80f8eb7 has introduced in unicode_norm.c some new code that uses
    htonl().  On at least some FreeBSD environments, it is possible to find
    that this function is undeclared, causing a compilation warning.  It is
    worth noting that no buildfarm members have reported this issue.

    Instead of adding a new inclusion to arpa/inet.h, switch to use
    the equivalent defined in pg_bswap.h, to benefit from any built-in
    function if the compiler has one.

    Reported-by: Masahiko Sawada
    Discussion: https://postgr.es/m/CA+fd4k7D4b12ShywWj=AbcHZzV1-OqMjNe7RZAu+tgz5rd_11A@mail.gmail.com
```

# バグ報告についての注意点

- すべてのバグがすぐに修正されるとは限らない
- 議論の結果、バグとみなされないケースもある
  - バグ＝ドキュメント等で記載されている仕様とは異なる挙動
- セキュリティ問題に関連するバグは、報告先が異なるので注意
  - security@postgresql.orgに直接送る
  - 非公開で議論される

# ソースコードを入手

- tarballを入手
  - https://www.postgresql.org/ftp/source/v12.8/
- githubからでもOK
  - https://github.com/postgres/postgres/releases/tag/REL_12_8
- ブラウザ上でも見れます
  - https://doxygen.postgresql.org/
- gitリポジトリをcloneしてもOK
  - git://git.postgresql.org/git/postgresql.git

```
$ wget https://ftp.postgresql.org/pub/source/v12.8/postgresql-12.8.tar.bz2

$ tar xjf postgresql-12.8.tar.bz2

$ cd postgresql-12.8
```

# ソースコードの構造 (src)

- `src/backend`
  - サーバ側のコード（SQL実行、WAL、レプリケーションなど）
- `src/bin`
  - クライアントのコード（psql, pg_dump, pg_upgradeなど）
- `src/common`
  - サーバとクライアント両方で使えるライブラリ的なもの
- `src/test`
  - リグレッションテスト

# ソースコードの構造 (src/backend)

- src/backend/tcop
  - SQLを処理するプロセス（Backendプロセス）の主な処理
- src/backend/commands
  - DDLを実行するコード
- src/backend/postmaster
  - postmasterプロセスやautovacuumプロセスなど

```
% ls src/backend
Makefile    commands    jit     nls.mk      partitioning  regex        statistics  utils
access      common.mk   lib     nodes       po            replication  storage
bootstrap   executor    libpq   optimizer   port          rewrite      tcop
catalog     foreign     main    parser      postmaster    snowball     tsearch
```

# ソースコードを読むときに知っておくと便利な関数

- SQLを受け取り処理する所
  - `src/backend/tcop/postgres.c`の`exec_simple_query()`
  - SQLをパースして、実行計画を作成して、実行する、という流れが見れる

- COPYやALTER TABLE等のDDLを実行する所
  - `src/backend/tcop/utility.c`の`standard_ProcessUtility()`と`ProcessUtilitySlow()`の2つ
  - 大きなswitch文があり、コマンド毎に実行する関数を変えている

# ソースコードを読むときに知っておくと便利な関数

- palloc()とpfree()
  - PostgreSQL版のmalloc()、free()
  - PostgreSQLでは独自のメモリ管理機構を持っており、palloc()したメモリは（トランザクション終了時等に）まとめて開放される
- ereport(), elog()
  - ログを書く関数
  - printfデバッグに便利
  - ログレベルがERROR以上だと、それ以降のコードは実行されないので注意
    - この場合、トランザクションは自動的にロールバックされます

# ALTER TABLE … RENAME TOをどうやってパースしているのか？

- RENAME TOは内部的には他のALTER TABLEコマンドとは別に解釈されているので注意
- これを確認するためには、パーサのコード（src/backend/parser/gram.y）を確認する必要がある

```
/*****************************************************************************
 *
 * ALTER THING name RENAME TO newname
 *
 *****************************************************************************/

RenameStmt: ALTER TABLE relation_expr RENAME TO name
                {
                    RenameStmt *n = makeNode(RenameStmt);
                    n->renameType = OBJECT_TABLE;
                    n->relation = $3;
                    n->subname = NULL;
                    n->newname = $6;
                    n->missing_ok = false;
                    $$ = (Node *)n;
                }
```

# RenameStmtの調査

- DDLなので、ProcessUtilitySlow()関数を見ていく
- ExecRenameStmt()が見つかる

```
case T_DropStmt:
    ExecDropStmt((DropStmt *) parsetree, isTopLevel);
    /* no commands stashed for DROP */
    commandCollected = true;
    break;

case T_RenameStmt:
    address = ExecRenameStmt((RenameStmt *) parsetree);
    break;

case T_AlterObjectDependsStmt:
    address =
        ExecAlterObjectDependsStmt((AlterObjectDependsStmt *) parsetree,
                                   &secondaryObject);
    break;
```

# ExecRenameStmt()を見る

```
ObjectAddress
ExecRenameStmt(RenameStmt *stmt)
{
    switch (stmt->renameType)
    {
        case OBJECT_TABCONSTRAINT:
        case OBJECT_DOMCONSTRAINT:
            return RenameConstraint(stmt);

        case OBJECT_DATABASE:
            return RenameDatabase(stmt->subname, stmt->newname);

        case OBJECT_ROLE:
            return RenameRole(stmt->subname, stmt->newname);

        case OBJECT_SCHEMA:
            return RenameSchema(stmt->subname, stmt->newname);

        case OBJECT_TABLESPACE:
            return RenameTableSpace(stmt->subname, stmt->newname);

        case OBJECT_TABLE:
        case OBJECT_SEQUENCE:
        case OBJECT_VIEW:
        case OBJECT_MATVIEW:
        case OBJECT_INDEX:
        case OBJECT_FOREIGN_TABLE:
            return RenameRelation(stmt);
```

# RenameRelation()を見る

- is_indexによって異なるロックレベルを指定している
- `is_index = stmt->renameType == OBJECT_INDEX;`

```
relid = RangeVarGetRelidExtended(stmt->relation,
                                 is_index ? ShareUpdateExclusiveLock : AccessExclusiveLock,
                                 stmt->missing_ok ? RVR_MISSING_OK : 0,
                                 RangeVarCallbackForAlterRelation,
                                 (void *) stmt);

if (!OidIsValid(relid))
{
    ereport(NOTICE,
            (errmsg("relation \"%s\" does not exist, skipping",
                    stmt->relation->relname)));
    return InvalidObjectAddress;
}

/* Do the work */
RenameRelationInternal(relid, stmt->newname, false, is_index);
```

# 動作チェックしてみる

- ALTER XXXによってロックレベルが変わるのかを確認する

```
diff --git a/src/backend/commands/tablecmds.c b/src/backend/commands/tablecmds.c
index 2e43a32..9b2ecfa 100644
--- a/src/backend/commands/tablecmds.c
+++ b/src/backend/commands/tablecmds.c
@@ -3418,6 +3418,9 @@ RenameRelation(RenameStmt *stmt)
     * Lock level used here should match RenameRelationInternal, to avoid lock
     * escalation.
     */
+
+   elog(NOTICE, "renameType %d, is_index %d", stmt->renameType, is_index);
+
    relid = RangeVarGetRelidExtended(stmt->relation,
                                     is_index ? ShareUpdateExclusiveLock : AccessExclusiveLock,
                                     stmt->missing_ok ? RVR_MISSING_OK : 0,
```

# ソースコードをビルド

```
$ ./configure --prefix=/home/masahiko/12.8 \
    --enable-debug --enable-cassert CFLAGS=-O0
$ make –j `nproc`
$ make install
```

# データベースを起動

```
$ initdb -D data
$ pg_ctl -D data start
$ psql
psql (12.8)
Type "help" for help.

postgres=#
```

# 早速ALTER TABLE RENAME TOを実行してみる

```
=# CREATE TABLE tbl1 (a int primary key);
CREATE TABLE

=# ALTER TABLE tbl1 RENAME TO tbl2;
NOTICE:  renameType 39, is_index 0
ALTER TABLE

=# ALTER INDEX tbl2 RENAME TO tbl1;
NOTICE:  renameType 20, is_index 1
ALTER INDEX

=# ALTER TABLE tbl1_pkey RENAME TO tbl2_pkay;
NOTICE:  renameType 39, is_index 0
ALTER TABLE
```

# ここまでのまとめ

ProcessUtilitySlow()

-> ExecRenameStmt()

-> RenameRelation()

とコードを追った。

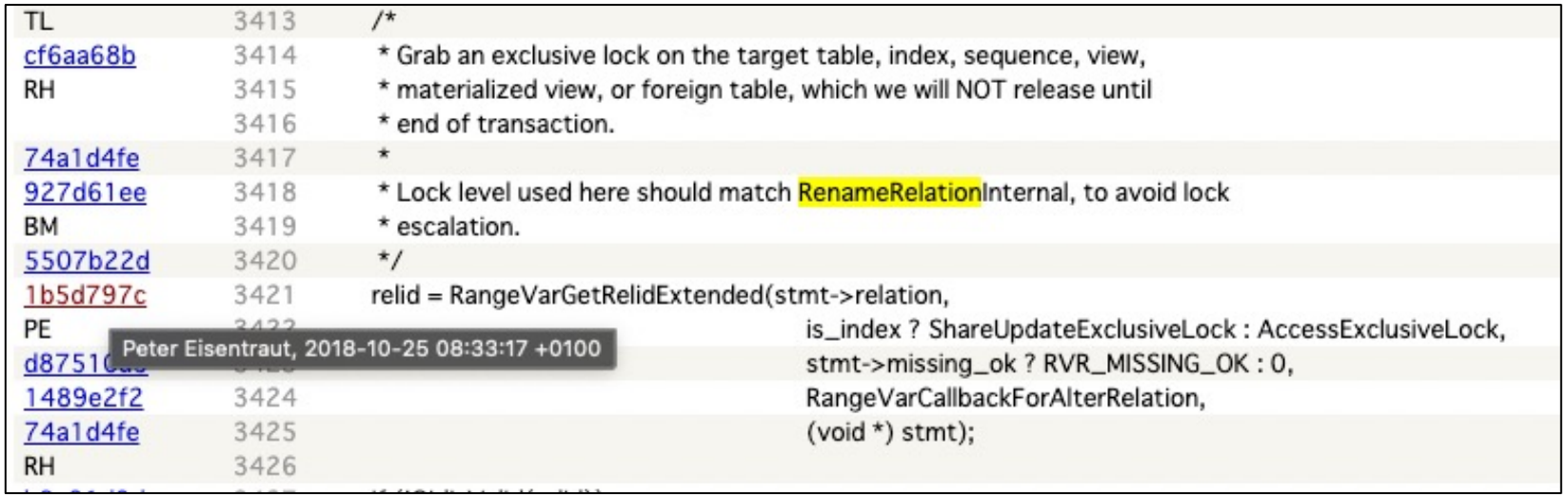
そして、RenameRelation()では、ALTER INDEXだったら、ShareUpdateExclusiveLockを指定していた。

でも…ALTER INDEXでもテーブルをRENAMEすることができるよね？

このコードに原因がありそう

# git blameでコミットを特定する

- ブラウザ上でも確認可能
  - https://git.postgresql.org/gitweb/?p=postgresql.git;a=blame;f=src/backend/commands/tablecmds.c;h=2e43a32ae55db2df7b2c36817da5996026b25b2e;hb=de835071fda945fb5e40340d3ea8dd2ca13e725c



# PG12で入った改善が原因だった

- インデックス改名で必要なロック取得を削減しました。 (Peter Eisentraut)

**Lower lock level for renaming indexes**

| | | |
|---|---|---|
| author | Peter Eisentraut <peter_e@gmx.net> | |
| | Thu, 25 Oct 2018 07:33:17 +0000 (08:33 +0100) | |
| committer | Peter Eisentraut <peter_e@gmx.net> | |
| | Wed, 14 Nov 2018 16:09:54 +0000 (17:09 +0100) | |
| commit | 1b5d797cd4f7133ff0d18e123fcf41c67a5a7b0b | |
| tree | 955a8bd049710d8e3df7c99bfcb49f07b074535c | tree |
| parent | b4721f39505b56dd7b556aef5428a0850230ca59 | commit \| diff |

Lower lock level for renaming indexes

Change lock level for renaming index (either ALTER INDEX or implicitly via some other commands) from AccessExclusiveLock to ShareUpdateExclusiveLock.

# バグのまとめ

- コミットc2c618ff1によって、インデックスのRENAMEに必要なロックレベルが下がった
- しかし、指定されたオブジェクトではなくALTER **INDEX**かどうかでロックレベルを判断していた
- なので、ALTER INDEX … RENAME TOでテーブルを指定すると、本来必要なレベルよりも弱いレベルのロックを取得してしまっていた
- PG12以降で発生する
- （当時の）最新バージョンであるPG14でも発生する
- ALTER INDEX … RENAME TOでも、インデックス以外が指定された場合は、AccessExclusiveLockを取得するべき

# どう直す？

- 基本的には、コミュニティにいる開発者が修正するのが確実で早い
  - バグの報告だけでもOK
  - 勉強のために、自分でパッチを書いてみるのもOK
- 初めての場合は、バグ報告、パッチのテストがおすすめ
- このバグの修正には少し工夫が必要だった
  - 指定されたオブジェクトの種類を確認してロックレベルを決めたいけど、オブジェクトの種類を確認するためにはオブジェクトをロックしてメタ情報を見る必要がある
- 興味がある方はこちらをチェック
  - https://github.com/postgres/postgres/commit/c2c618ff113

# ソースコードレベルでの調査や修正

- 一般的なDBの知識や、PostgreSQLの知識が必要
  - The Internals of PostgreSQL
    - https://www.interdb.jp/pg/
  - PostgreSQL Internals
    - https://www.postgresqlinternals.org/
- コミュニティでの動き方も知っておくと◎
  - PostgreSQL開発コミュニティに参加しよう！
    - https://www.slideshare.net/nttdata-tech/postgresql-global-development-group-osc2022-online-kyoto-nttdata
  - PostgreSQLのバグとの付き合い方
    - https://www.slideshare.net/nttdata-tech/postgresql-bugs-guideline-postgresql-conference-japan-2022-nttdata

# 相談できる場所

- 日本PostgreSQLユーザ会のSlack
  - 1500人以上いる
  - PostgreSQL開発者も多数
- PostgreSQLアンカンファレンス
  - 毎月開催しているオンラインカンファレンス
  - 発表内容はPostgreSQLに関連しているものであれば何でもOK
  - connpassにて「PostgreSQLアンカンファレンス」で検索

# Thank you

@masahiko_sawada