# libev+OpenSSL =EvServer (PostgreSQL編)

Future Versatile Group/MyDNS.JP
T.Kabu
2020.11.02 #pgunconf

# ▶ふと思い立って…

- Postgres用マルチメインDBの構築方法を考察
  →PgBouncerの内部構造を解析してみた
- クライアントからの接続とデータを受け付けて
  PostgreSQLに対してリレー＝PgなProxyか
- 接続/切断を繰り返さない＝負荷が軽いらしい

…けっこうスパゲッティで挫折しかける

# ▶やりたいことは…


Clientからのクエリにより、取得(SELECTとか)なら最も速いDBから、
その他変更系なら全ての対象DBにトランザクションでクエリを発行して、
全結果によりコミットかロールバックが決める、みたいなのでどうよ？
(もちろん処理の内容によりこの辺は細かく制御しないとね)

# ▶すでにないの？


↑こいつと似たようなことをしているのが

- PgPool-II　　多機能
- PgBouncer　シンプル＆高速

# ▶お金払えば…


まさにPostgreSQLでマルチメインできるのが

・PostgreSQL BDR

(1ライセンス$10,000～とか)

# ▶PgBouncer

・とりあえずソースをGitHUBで眺める
・C言語で記述されてる(なんとか読める!)
・libeventを使用(イベントドリブンなのねー)
・記述の統一感が!?(切った貼ったスパゲッチ!?)
・ともあれLinux上でVSCでデバッグ環境構築
・ブレークさせながらソースにコメント付けて読解

# ▶libevent

・イベント処理を汎用的に記述させるライブラリ
・selectやepollとかをラップして移植性を上げる
・その後libuvとかlibevとかより良いものが登場
・PgBouncerで、libeventの動作概要を把握
　→実際ソース起こすなら改善版のlibevだよな

…でも実際libevでSSL/TLSまでやってる例が見当たらず、libeventと同様にできるか不安

# ▶やってから悩もう!

・PgBouncerの動作をだいたい理解したけど…
・ナウいイベントドリブンはやったことないから不安
・とはいえ通信の世界では状態＆イベントは常識
（イベントマトリックス、関数の多次元配列化）

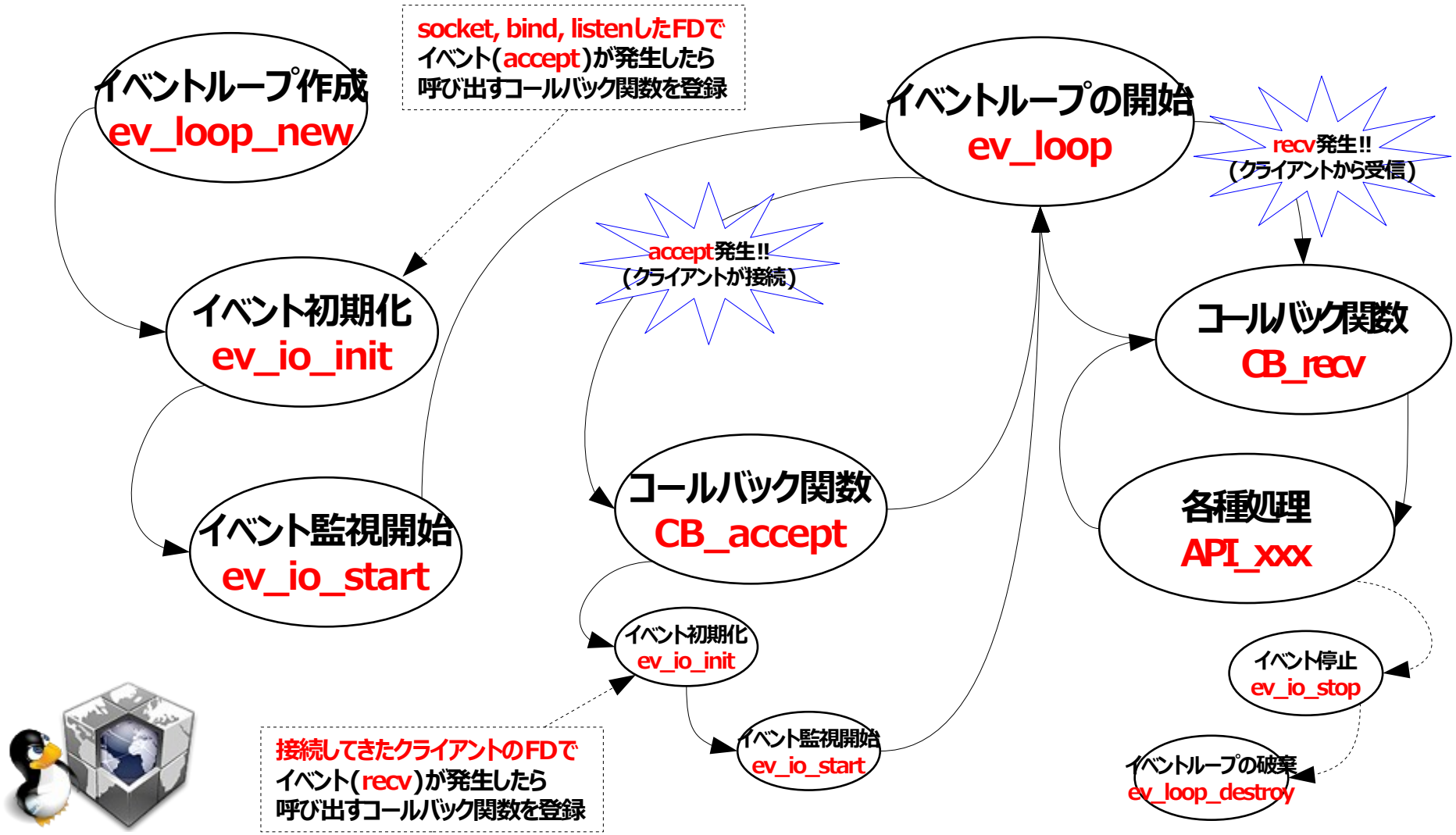
とりまスケルトン作ってlibevを把握しよう!!

# ▶libevの基本

・扱えるイベント(＝割込)の種類が豊富
・イベントの処理(初期化やコールバック関数の登録)が簡単
  1) イベントループ作成 ⇒ ev_loop_new
  2) イベント初期化 ⇒ ev_io_init
  3) イベント監視開始 ⇒ ev_io_start
  4) イベントループの開始 ⇒ ev_loop
  5) イベントループの破棄 ⇒ ev_loop_destroy

割込＝イベントが発生した時に呼び出す
コールバック関数をev_io_initで登録するだけ！

# ▶libevのイメージ


イベントループ作成
ev_loop_new
socket, bind, listenしたFDで
イベント(accept)が発生したら
呼び出すコールバック関数を登録
イベントループの開始
ev_loop
recv発生!!
(クライアントから受信)
accept発生!!
(クライアントが接続)
イベント初期化
ev_io_init
コールバック関数
CB_recv
イベント監視開始
ev_io_start
コールバック関数
CB_accept
各種処理
API_xxx
イベント初期化
ev_io_init
イベント停止
ev_io_stop
イベント監視開始
ev_io_start
イベントループの破棄
ev_loop_destroy
接続してきたクライアントのFDで
イベント(recv)が発生したら
呼び出すコールバック関数を登録

# ▶OpenSSLの基本

CentOS7はOpenSSLが1.0.x、CentOS8は1.1.x…
(ただでさえ、みんな泣いている話しか聞かないのに)

1) OpenSSLの各種初期化処理
→OpenSSL 1.1.0以降、明示的な初期化は必要ないヨ
2) SSL設定情報を作成＆設定
→許可プロトコルとかサーバー証明書とかは1.1.xでも必要

…で、普通に、socket→bind→listenまでは同じだが、
このあとのacceptでの接続受付からのrecvで受信という
流れが非SSL通信とちょっと違う。

# ▶OpenSSLの基本

まずクライアントから接続が来た(accept)なら…

3) SSL_newでSSL接続情報を作成

4) ファイルディスクリプタを、SSL_set_fdでSSL接続情報に紐づけ(bindみたいなもん)

で、クライアントからデータが来た(recv)なら

5) SSL_accept()でSSLセッションを確立させる

6) SSLセッションが確立したらSSL_readでデータ受信

7) SSLセッションで必要に応じてSSL_writeでデータ送信

※復号化/暗号化処理は勝手にやってくれるのでラクチン!

# ▶OpenSSLの基本

で、クライアントと切断の際には

8) SSL_get_fdでSSLセッションで使用しているFDを取得

9) SSL_freeでSSL接続情報を破棄

10) そしたら普通にdoseすればOK

こんな感じで、非SSLなセッションを確立したら、その上に改めてSSLなセッションを確立する感じで、大したことはない。

問題は、

SSL_acceptやSSL_readで

イベント発生するの？

# ▶libev＆OpenSSL

・結果から言うと、難しく考えなくてもうまくいった
・「何か」データが来るとイベントが発生するので、recv処理で
その接続がSSL/TLS接続かどうかで受信処理を分ければOK
1) 非暗号化通信なら
→そのまま受信(read)
2) SSL/TLS接続モードなら
→2-1)SSL/TLSハンドシェイクしてSSL/TLS接続を確立
→2-2)接続確立後ならSSL_readで復号データを受信
※STARTTLSのような、非暗号通信から暗号通信に
途中から切り替わる場合も、似たり寄ったり

# ▶スケルトン完成…

・GitHUBで『EvServer』として公開中

https://github.com/disco-v8/EvServer

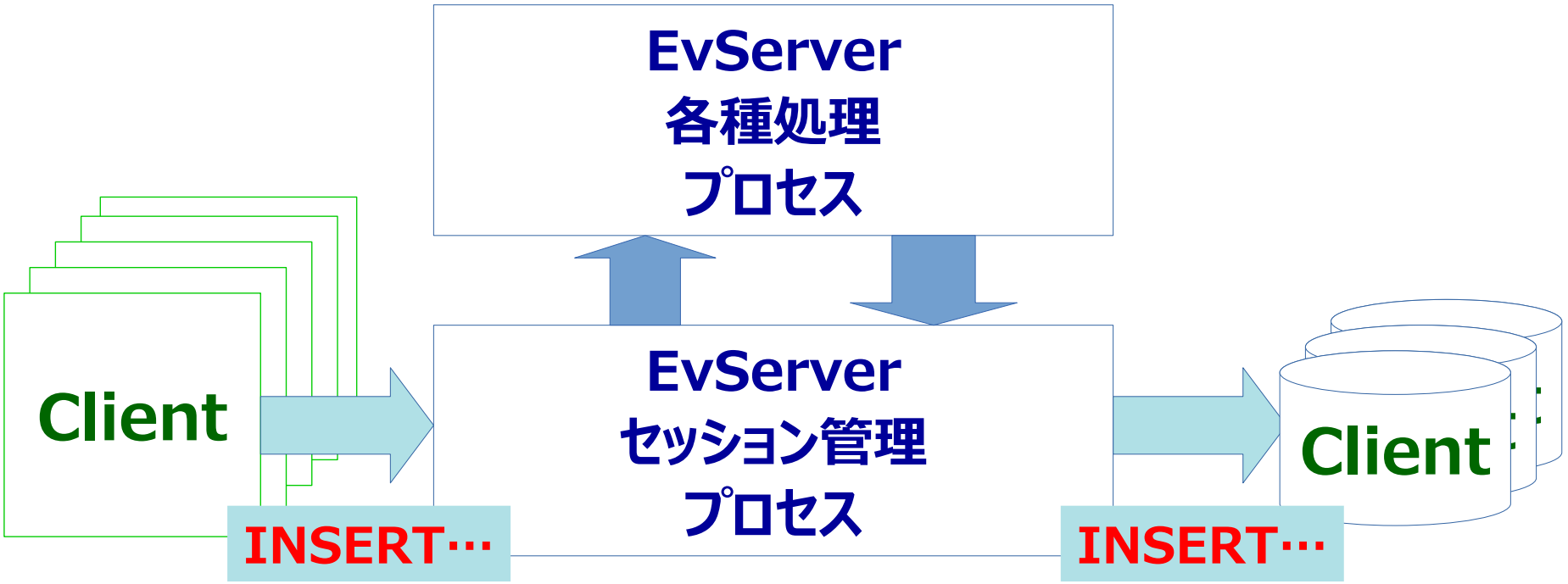
・例によって日本語による、ほぼ一行一コメント
・勉強するにはちょうどいい…のかな？感想歓迎！

# ▶例えばHTTPD

・nginxやApacheみたいなWEBサーバーを作成
・てけとーWEBサーバーですが、FastCGIも実装！
・GETだけでなくPOST(multi-part)にも対応！
・Apacheよりは、1.2～1.3倍速いです(ab測定)
・まさか二十年以上たってからまたFastCGIを触るとは!!

というか、FastCGIプロトコル概念が
古すぎだろ…ブツブツ…64kBだと!?

# ▶そして発展形(本題)




WEBサーバーは一段落したので、本来のPostgreSQL方面を。
面倒な処理は、FastCGI同様に別プロセスとソケット通信で接続し、
そちらに任せた方が、C言語で全部やるよりぜったいに楽かなー(笑)

# ▶PostgreSQL

・『フリーなオープンソースのリレーショナルデータベース管理システム』(https://lets.postgresql.jp/ より)
・CLIからのアクセスツールとしてpsqlがある

```
psql hoge_db -U hogehoge -p 5432 -h localhost
```

・psqlとPostgreSQLのやり取り(プロトコル)を把握
→第52章 フロントエンド/バックエンドプロトコル

https://www.postgresql.jp/document/12/html/protocol.html

# ▶接続手順

・クライアントからの開始メッセージは二種類
  1) SSLRequest（SSL接続を要求）
  2) StartupMessage（実際の接続要求）
・最新のpsqlは標準でSSLRequestから
  →PostgreSQLは"S"(=OK)か"N"(=NG)を返す
・SならSSLハンドシェイク開始(平文から暗号化通信へ)
・Nなら(もしくは暗号化通信確立後)に、2)がくる

# ▶接続確立後は…

・PostgreSQLからAuthenticationOkで接続確立
・接続情報がParameterStatusで色々送られて来る
・最終的にReadyForQueryが来たらクエリを送れる
・そしたらpsqlからクエリを送れるのでQueryで送ると…
・PostgreSQLからは、RowDescription、
DataRow(複数の場合あり)、
CommandCompleteのように応答があり…
・再びReadyForQueryが来たらクエリを送れる

# ▶フォーマットが…

・やり取りは簡単だけど、メッセージのフォーマットが…

1) 開始メッセージ系は最初の四バイトがメッセージ長

xx xx xx xx xx …←int32に変換する必要あり

※この長さでSSLRequestかどうか判定

2) それ以外のメッセージは

xx xx xx xx xx xx xx xx xx xx …

メッセージ本体(メッセージ長-4バイト)

メッセージ長(4バイト、自身を含む)

メッセージタイプ(1バイト)

# ▶とりまさておき…

・メッセージタイプはいろいろあってPostgreSQLの歴史を感じる(笑)
・前述のpsqlとPostgreSQLのやり取りを実際に把握しないと、本題に取り掛かれないよなぁ

まずはEvServerをカスタマイズして
やり取りを可視化するPgAnalyzerを作ってみるか!!
まさにこれというのが↑見当たらなかったので

# ▶PgAnalyzer

・psqlとPostgreSQLの間に入り、メッセージを解析しつつバケツリレーするだけ

実際に動いているところを
見ていただいた方が速いですね

# ▶OpenSSLに泣く

・PostgreSQLではナウでヤングなsha256認証まで
やったら、自分もOpenSSLで泣きました

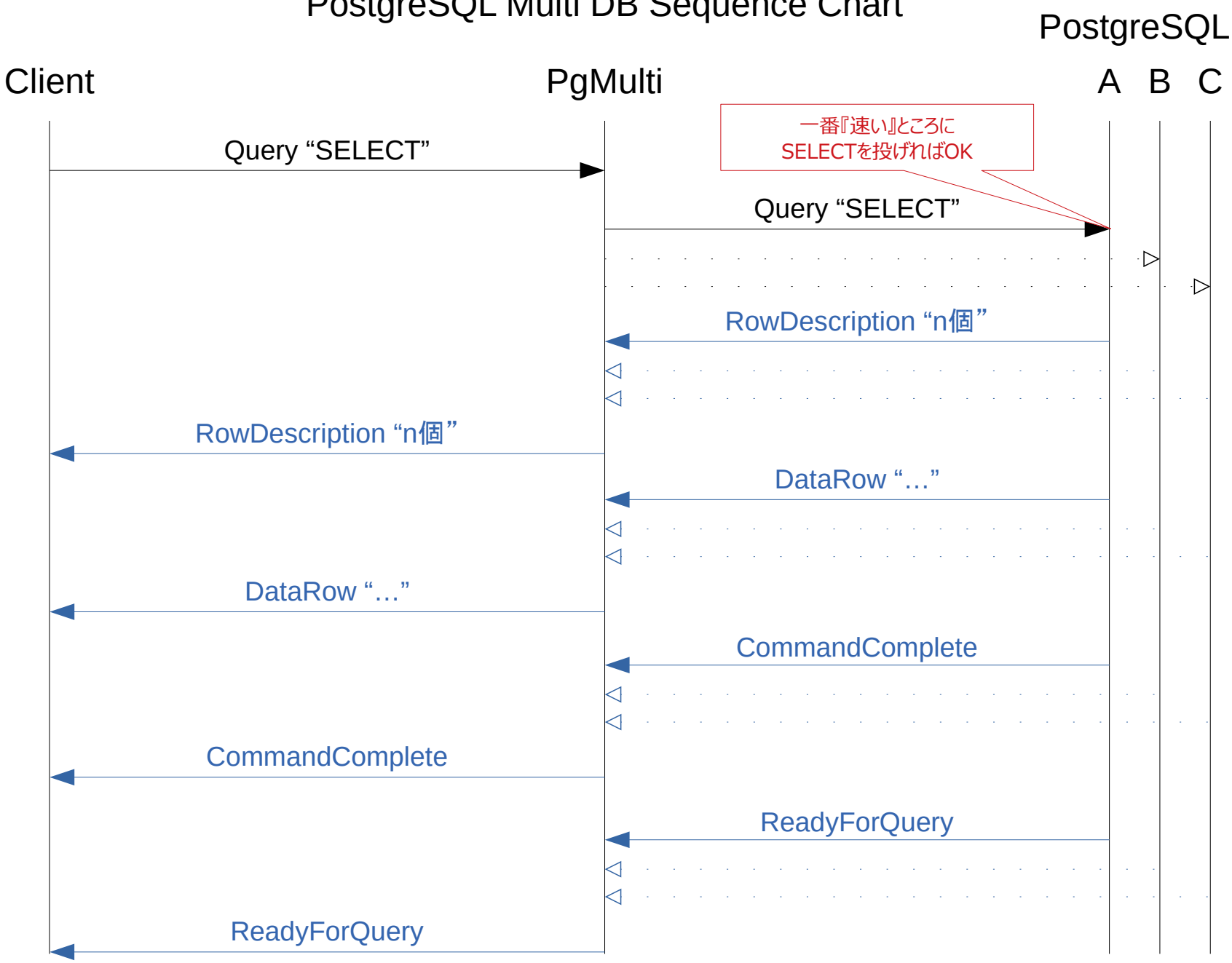
実際に動いているところを
見ていただいた方が速いですね
( ノД`)ｼｸｼｸ…

# ▶マルチメインDB目指して

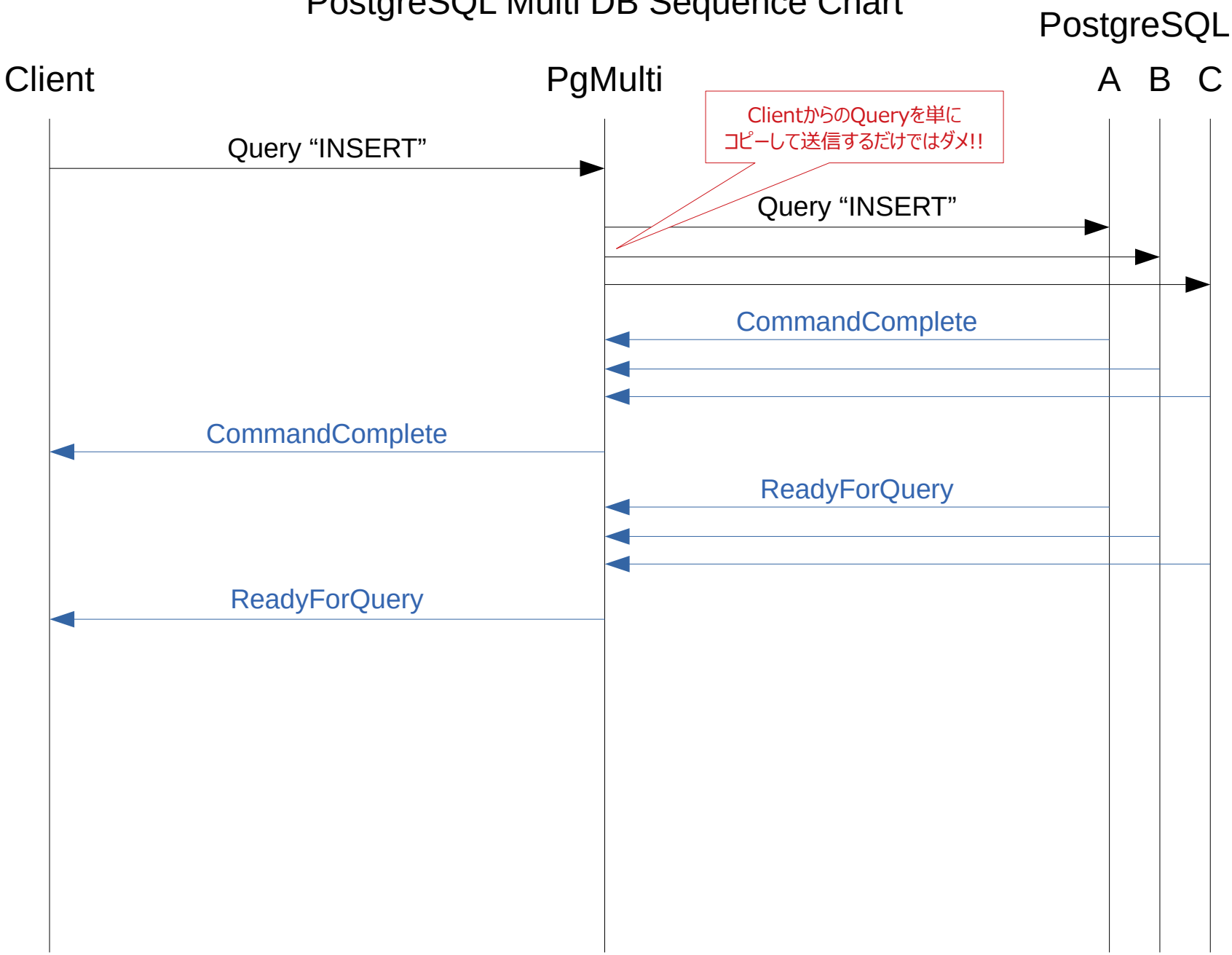

・簡易問い合わせ(Query)の中に以下の命令がある場合、
  ・INSERT
  ・UPDATE
  ・DELETE
  データベースの書き換えが発生する、という事になる。
・上記以外の
  ・SELECT
  については、自分から一番近い(速い)データベースに対してのみ
  クエリを投げればいいのでは？

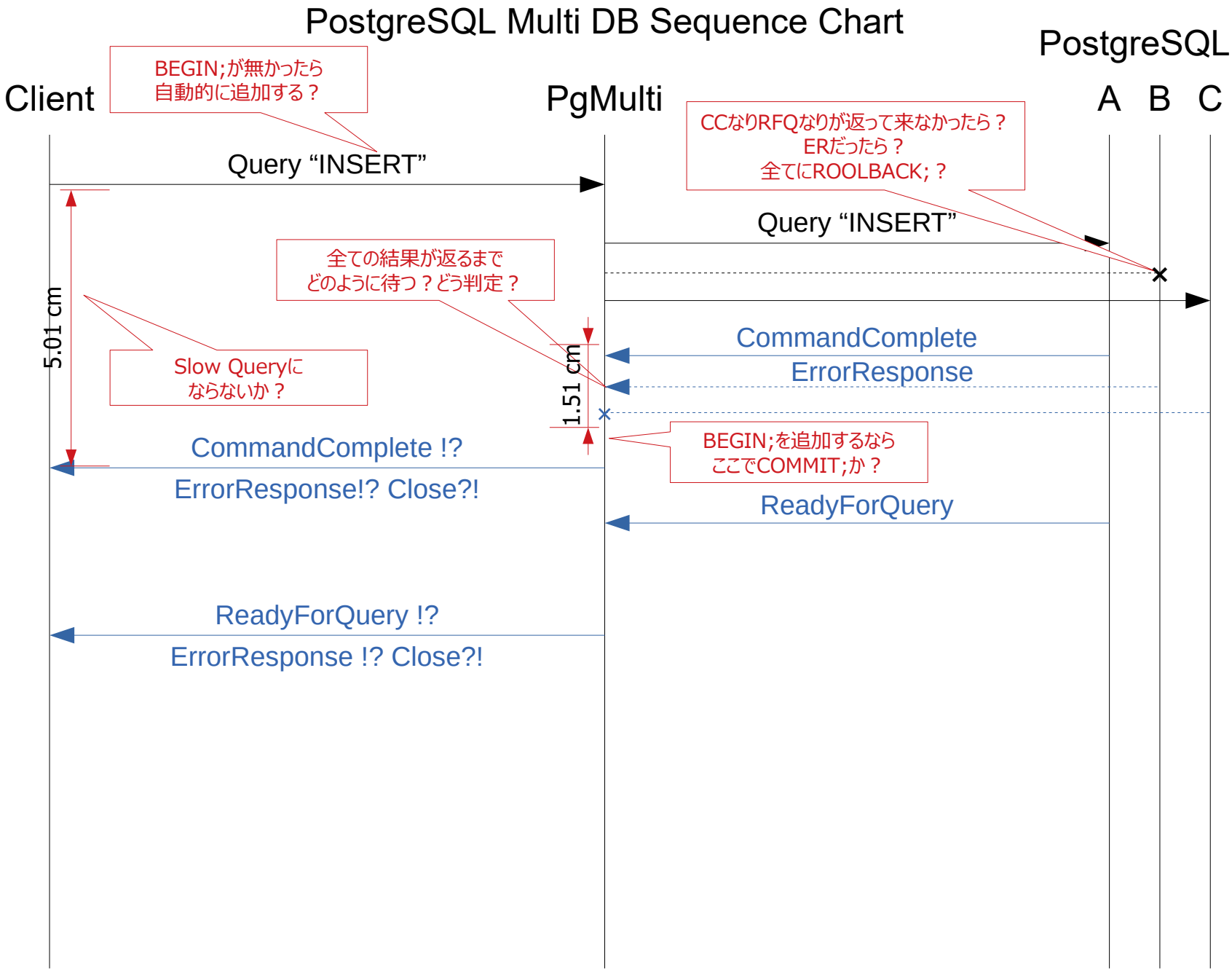
# PostgreSQL Multi DB Sequence Chart

# PostgreSQL Multi DB Sequence Chart

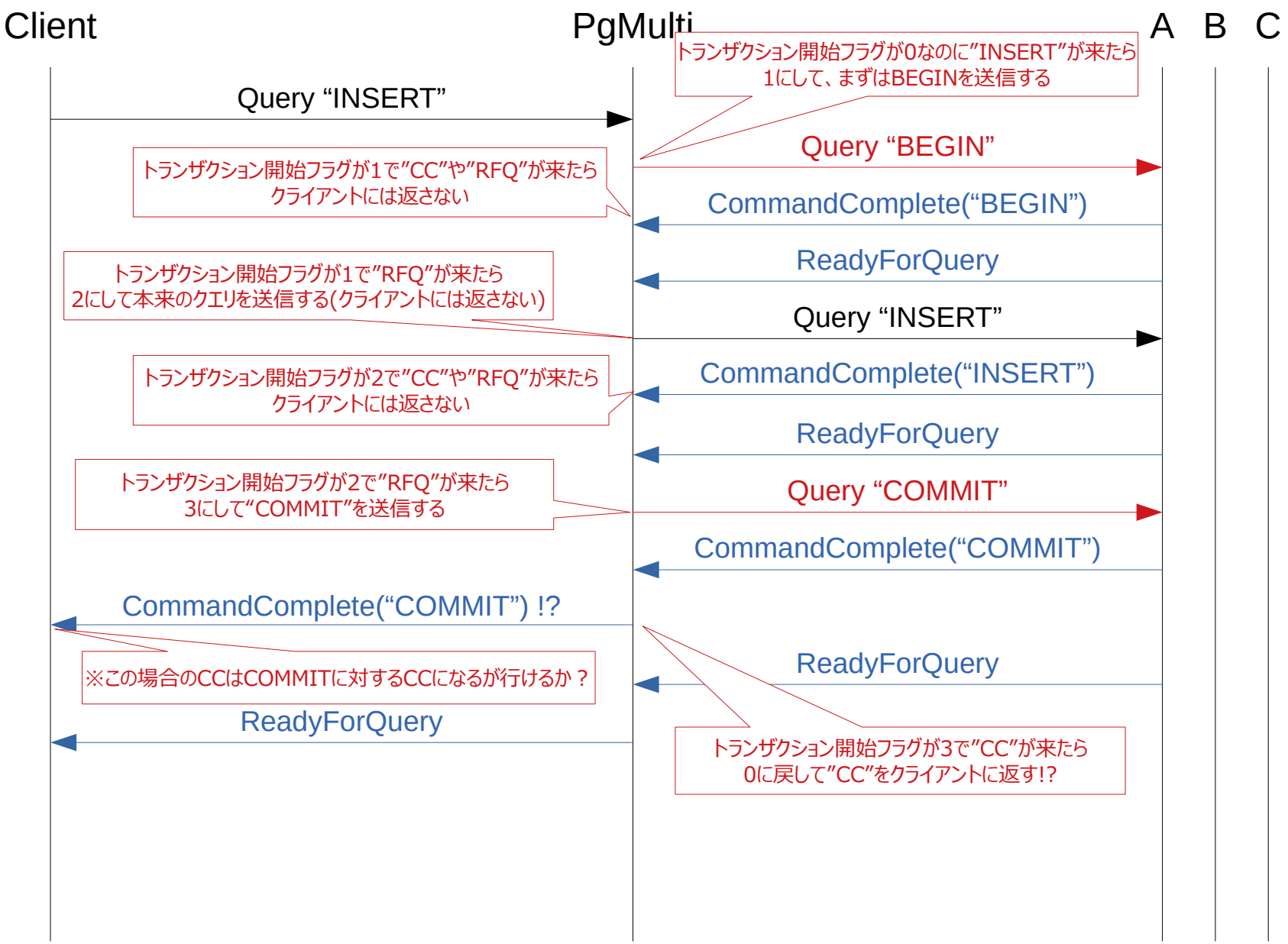


# PostgreSQL Multi DB Sequence Chart

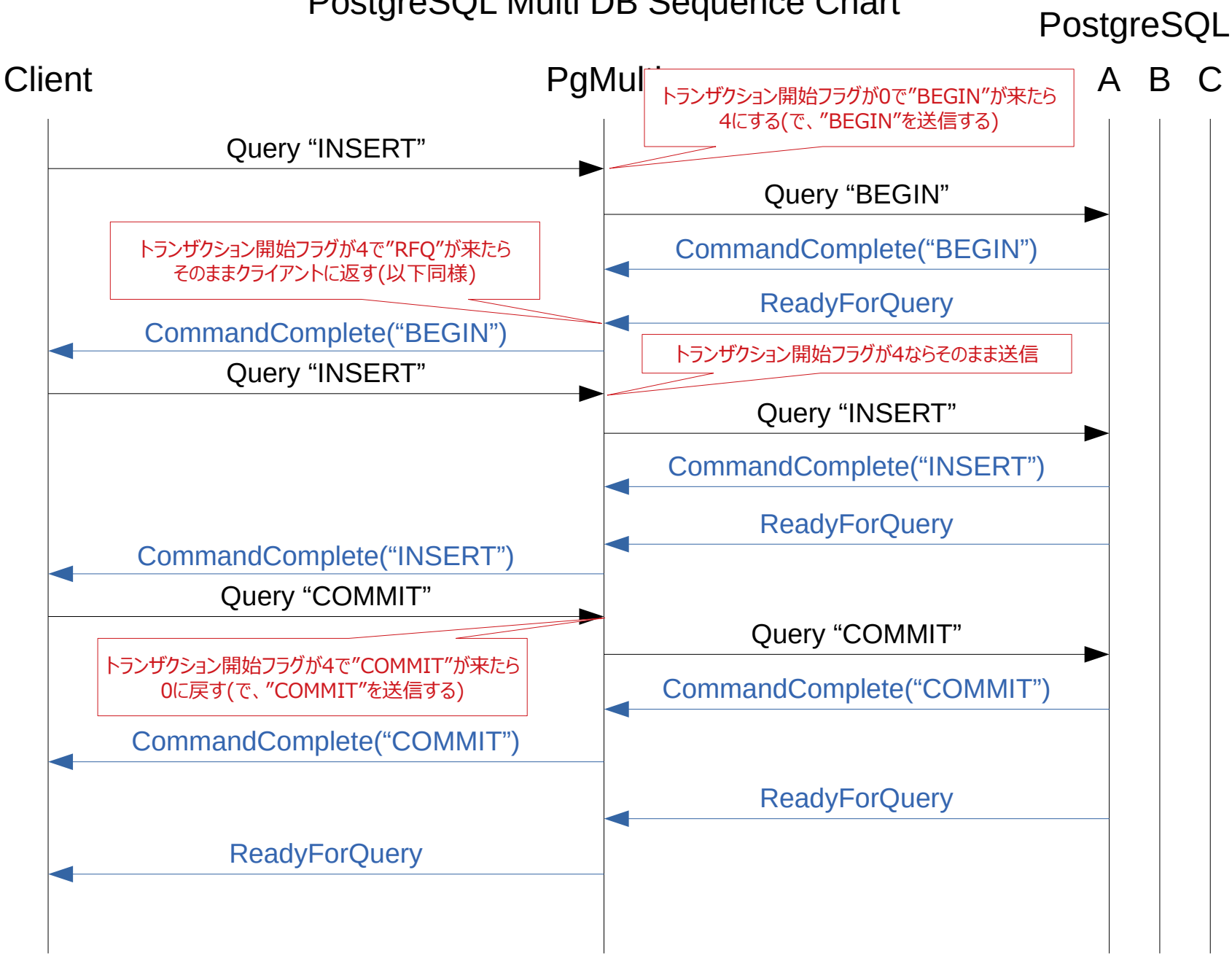



PostgreSQL Multi DB Sequence Chart
Client
PgMulti
PostgreSQL
A
B
C
BEGIN;が無かったら
自動的に追加する？
Query “INSERT”
CCなりRFQなりが返って来なかったら？
ERだったら？
全てにROOLBACK; ？
Query “INSERT”
全ての結果が返るまで
どのように待つ？どう判定？
5.01 cm
Slow Queryに
ならないか？
1.51 cm
CommandComplete
ErrorResponse
BEGIN;を追加するなら
ここでCOMMIT;か？
CommandComplete !?
ErrorResponse!? Close?!
ReadyForQuery
ReadyForQuery !?
ErrorResponse !? Close?!

# PostgreSQL Multi DB Sequence Chart



# PostgreSQL Multi DB Sequence Chart



# ▶マルチメインDB目指して


・PgAnalyzerはGitHubに公開予定
・引き続き、まったりと本題の開発を進める予定

こんな感じで、PostgreSQL用のマルチメインDB
実現用のミドルウェアを模索します。
たぶん機能とかを割り切って実装すれば、
そんなに難しくなく実現できるはず…かなー？

ご期待ください!?

# おしまい

Future Versatile Group/MyDNS.JP
T.Kabu