Engineer

# コネクションプール

🕮 開いたデータベース接続を再利用し、同時に利用できる接続数を制限する仕組み

## どのようなものか

コネクションプールは、アプリケーションが開いたまま保持し、繰り返し貸し出すデータベース接続の集まりです。処理のたびに接続を開き、終わったら閉じる代わりに使います。コードはプールから接続を借り、その接続でSQL文を実行してから返します。次に接続が必要になった呼び出し元は、既に開いている同じ接続を受け取れます。SQLAlchemyのドキュメントは、プーリングを「長期間使う接続を効率よく再利用するために、メモリー内に維持する標準的な手法」と説明しています。同じページは、プールにもう一つの役割を与えています。アプリケーションが同時に使える接続数を管理し、急に通信が増えても接続をさらに開くのではなく、空くまで待たせる役割です。

プールの置き場所には二つあります。アプリケーション内では、データベースライブラリーの一部になります。SQLAlchemyの `Engine` には必ず一つ付属し、DjangoはPostgreSQL用に各プロセス内で一つ保持できます。アプリケーション外では、クライアントとデータベースサーバーの間に置く独立したプログラムになります。PgBouncerは、PostgreSQL向けのそうした接続プール管理プログラムです。アプリケーションはデータベースと同じようにPgBouncerへ接続し、PgBouncerが自分のプールからサーバー接続を貸し出します。両方を組み合わせることもでき、Djangoのデータベースリファレンスには、PgBouncerのようなトランザクション単位のプールの背後で動かす場合の節があります。

接続を維持する価値は、サーバー側を見ると分かります。PostgreSQLは、クライアントごとに専用のプロセスを割り当てます。postmasterという監督プロセスが着信を待ち受け、接続要求のたびに新しいサーバープロセスを起動します。各クライアントは、そのうちちょうど一つのプロセスにつながります。つまり、接続を開くことは、データベースサーバー上でプロセスを起動することです。プールなら、その負担を一度だけ払い、結果を再利用できます。

概念を判断に生かす

「コネクションプールとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 年々重くなってきたシステムのクエリと不足するインデックスを調べ、効果のある箇所を改善します。

**サービスを見る** お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 30日分の電力データを180 msで処理 — 180 ms 1か月分のメーター値を集計

**30日分の電力データを180 msで処理** お問い合わせ →

## なぜ重要なのか

第一の理由は、接続の負担です。Djangoのドキュメントは、接続を維持する理由を明快に説明しています。永続接続を使えば、HTTPリクエストごとに新しい接続を確立する負担を避けられます。ただし、既定の動作は逆です。`CONN_MAX_AGE` は0で、各リクエストの終了時に接続を閉じます。正の秒数を指定すると再利用のために開いておき、 `None` なら期限なく開いておきます。リクエストが終わっても残る接続をスレッドごとに一つ持つ、最も単純な再利用の形です。プールは、それを一般化した形です。

第二の理由は、サーバーの上限です。PostgreSQLの `max_connections` は通常、既定値が100ですが、カーネルの設定が対応できなければ、それより小さくなることもあります。変更できるのはサーバー起動時だけです。既定では、そのうち三つの接続枠がスーパーユーザー用に予約されています。上限を増やしても、負担がないわけではありません。PostgreSQLは、共有メモリーなど一部のリソースの大きさを、 `max_connections` から直接決めます。プールがなければ、各ワーカーが専用の接続を保持します。Djangoのドキュメントは、その帰結も明記しています。各スレッドが自分の接続を持つため、データベースは少なくともワーカースレッド数と同じだけの同時接続を受け入れなければなりません。プールを使えば、チームが選び、合計を計算できる接続数に変えられます。

第三の理由は、接続を健全に保つことです。接続は、借り手が替わるたびに必ずプールを通るため、そこで後始末と検査を行えます。SQLAlchemyのプールは、接続が戻ると `rollback()` を呼び、ロックを含むトランザクションの状態を解放します。次の借り手が、未完了のトランザクションを引き継がないためです。PgBouncerの既定のセッションプーリングモードでは、サーバー接続を解放してから次のクライアントに渡すまでに、 `server_reset_query` を実行します。既定値は `DISCARD ALL` です。また、プールは貸し出す前に接続を検査できます。しばらく使われていない接続では、これが重要になります。

# 実際の活用例

SQLAlchemy 2.1では、SQLiteのメモリー内データベースを除き、すべての `Engine` に既定で `QueuePool` が付きます。 `pool_size` は5、 `max_overflow` は10なので、一つのエンジンは同時に最大15接続を持てます。すべて使用中なら、呼び出し元は `pool_timeout` の時間だけ待ってから断念します。既定では30秒です。設定は、 `create_engine()` にキーワード引数として渡します。

```
from sqlalchemy import create_engine

# このエンジンは最大10接続。保持するのは10接続までで、超過は認めない。
engine = create_engine(
    "postgresql+psycopg2://me@localhost/mydb",
    pool_size=10,
    max_overflow=0,
    pool_timeout=10,  # 空き接続を10秒待ち、それでもなければ断念する
    pool_pre_ping=True,  # 接続を貸し出すたびに検査する
    pool_recycle=3600,  # 作成から1時間を超えた接続を置き換える
)
```

`pool_pre_ping` を指定すると、貸し出す際に、ドライバー自身のpingか、 `SELECT 1` に相当する文で、接続がまだ使えるかを確認します。 `pool_recycle` は、一定期間使われなかった接続を閉じるサーバー向けの設定です。3600なら、開いてから一時間を超えた接続を、次に貸し出すときに置き換えます。

Django 6.1には、PostgreSQL用の組み込みプールがあります。psycopgドライバーでのみ動作し、 `psycopg[pool]` パッケージが必要です。データベース設定の `"OPTIONS": {"pool": True}` で有効にするか、psycopgの `ConnectionPool` 用のオプションを辞書で渡します。プールはスレッド間で接続を再利用し、各プロセスが、データベースの別名ごとに独立したプールを持ちます。プーリングが有効な場合、 `CONN_MAX_AGE` は0のままにしなければなりません。プール内の接続の寿命は、代わりにプールの `max_lifetime` オプションで設定します。ASGIでは、Djangoのドキュメントは、永続接続を無効にして、この組み込みプールを使うよう勧めています。

PgBouncerの `pool_mode` は、サーバー接続をいつ別のクライアントに渡せるかを決めます。既定のセッションモードでは、クライアントが切断するまで接続を保持します。トランザクションモードでは、クライアントがサーバー接続を持つのはトランザクションの実行中だけなので、数十のサーバー接続で、はるかに多数のクライアントを処理できます。ステートメントモードでは、クエリのたびに接続を解放し、複数の文にまたがるトランザクションを許しません。各プールは、

ユーザーとデータベースの組み合わせごとに、`default_pool_size` までのサーバー接続を持ちます。既定値は20です。PgBouncerが受け入れるクライアント接続数の上限は `max_client_conn` で、既定値は100です。

当社自身のノートでも、早い段階でプールに取り組むよう勧めています。シャーディングの前にデータベースを最適化するでは、`pg_stat_activity` のアクティブなセッション数を、CPUコア数と比較します。アクティブなセッション数がコア数の何倍にもなると、サーバーは切り替えに時間を使うようになり、ハードウェアを増強する前に、コネクションプールでそれを解決できます。最後の一枚を一度だけ売るは、同じ考えを短時間に注文が集中する販売に適用します。十万人の購入者は、一つのロックを待つ数千のサーバープロセスではなく、PgBouncerを通じた数十の接続に集約されるべきです。

## 問題が起きるところ

**プールの合計がサーバーの上限を超えます。** 各プールが数えるのは、自分自身の接続だけです。SQLAlchemyの既定値では、一つのエンジンが15接続に達するため、十二のプロセスは、既定の `max_connections` が通常100で、そのうち三つをスーパーユーザー用に予約するサーバーに、180接続を求める可能性があります。Djangoは、各プロセスの各データベース別名に独立したプールを持ちます。そのデータベースにつなぐ、すべてのアプリケーションの全プロセスと、すべての定期実行ジョブにあるプールの最大接続数を合計します。合計は、`max_connections` から予約枠を引いた数より小さく保ちます。収まらなければ、前段にPgBouncerを置き、サーバー接続数を管理させます。

**プール内で待つ間に接続が切れます。** 使われていない接続が、プールの知らない間に相手側で閉じられると、次の借り手は最初のSQL文の実行で失敗します。SQLAlchemyの `pool_pre_ping` か、Djangoの `CONN_HEALTH_CHECKS` を有効にします。後者は、再利用する接続をリクエストごとに一度検査します。また、`pool_recycle` またはpsycopgの `max_lifetime` を、サーバーがアイドル接続を閉じるまでの時間より短くします。どちらも、トランザクションの途中で切れた接続には対処できません。SQLAlchemyのドキュメントは、pre-pingがこの場合を処理しないと明言しています。そのため、アプリケーション自身が処理を再試行する必要があります。

**フォークしたワーカーが親の接続を引き継ぎます。** ウェブサーバーやジョブ実行基盤が、エンジンを作成してからワーカーをフォークすると、すべての子プロセスが、親と同じプール内の接続を持つことになります。SQLAlchemyのドキュメントは、フォークしたプロセスとプール内の接続を共有しないことが極めて重要だとしています。共有すると、両方のプロセスが同じネットワーク接続上で通

信してしまうためです。推奨される方法に従い、各子プロセスの初期化処理で `engine.dispose(close=False)` を呼びます。子は空のプールから始め、親の接続には触れません。代わりに、使用のたびに接続を開いて閉じる `NullPool` を使うか、フォークする前に `engine.dispose()` を呼ぶ方法もあります。

**トランザクション単位のプールがセッション状態を壊します。** PgBouncerのトランザクションモードでは、同じクライアントの二つのトランザクションが、異なるサーバー接続で実行されることがあります。そのため、クライアントがセッションに設定したものは、次のトランザクションでも存在するとは限りません。PgBouncerの機能一覧は、 `SET` 、 `LISTEN` 、 `WITH HOLD` を指定したカーソル、SQLレベルの `PREPARE` 、セッション単位のアドバイザリーロックを、このモードでは一切サポートしないとしています。また、このモードは本質的に、クライアントがサーバーについて置く前提の一部を崩すとも明記しています。トランザクション単位のプールを通るコードにはセッション状態を持ち込まず、それが必要なクライアントはセッションモードのプールへ送ります。Djangoでは、トランザクション単位のプールの背後にあるデータベースについて、 `DISABLE_SERVER_SIDE_CURSORS` を `True` にするか、該当するクエリを `atomic()` 内で実行します。

## 参考資料

- コネクションプーリング — SQLAlchemy 2.1の `QueuePool` と既定値、切断への対応、返却時のリセット、フォーク時のプールの扱いを説明します。
- PgBouncerの機能 — セッション、トランザクション、ステートメントの各モードと、それぞれが対応するPostgreSQLの機能を説明します。
- PgBouncerの設定 — `pool_mode` 、 `default_pool_size` 、 `max_client_conn` 、タイムアウトと、その既定値を説明します。
- 接続が確立される仕組み — ユーザーごとにプロセスを持つPostgreSQLのモデルと、postmasterを説明します。
- 接続と認証 — `max_connections` と、スーパーユーザーやほかのロール用の予約接続枠を説明します。
- データベース — Djangoのリファレンスで、永続接続、PostgreSQLのコネクションプール、トランザクション単位のプールを扱います。



用語集

実際の制約に合わせて選ぶ

代表的なデータセット、そこから答えを得たい問い、現在のボトルネックをお知らせください。品質、処理量、復旧の測り方を合意できます。

料金を見る お問い合わせ →

# 用語を実務に結び付ける

「コネクションプールとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/connection-pool/