はじめに

■コースの概要と目的

PostgreSQL を使用した開発や運用管理を行うには、PostgreSQL データベースの内部構造・特徴を正しく理解することが重要です。本セミナーでは、PostgreSQL のアーキテクチャ全般、データベースの運用管理のポイント、チューニング手法について解説します。

■受講対象者

・PostgreSQL の導入をご検討されている方。
・これから PostgreSQL を本格的に使用する管理者の方。
・現在ご利用中の PostgreSQL の運用管理でお困り/お悩みの方。

■前提条件

以下 2 点を満たしている方。

・RDBMS の基礎理解
「PostgreSQL 入門」コースを受講された方、もしくは、
データベースやリレーショナル・データベースの概要、SQL 操作やトランザクションを理解している方。

・Linux の基礎理解
Linux の基本操作として、ディレクトリ操作（ls、pwd、cd コマンド）やファイル操作（cat、vi コマンド）、Linux ユーザー操作（su、sudo コマンド）を理解している方。

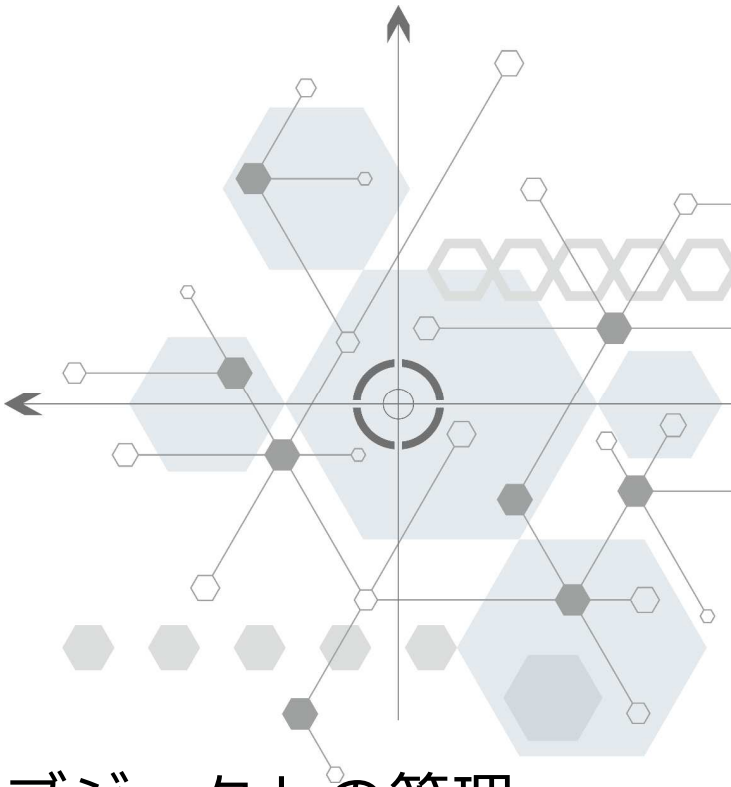
■テキスト内の記述について

▼構文

[ ]	省略可能
{ A | B }	A または B のどちらかを選択

▼マーク

	注意事項
	参考情報
	参照ページ



第 4 章

データベースオブジェクトの管理

この章では、PostgreSQL データベースオブジェクトの管理について解説します。

04 表の管理

変更や削除が頻繁に発生する表ではデータ構造が劣化します。格納効率が低下することで余分な領域を消費し、検索パフォーマンスの低下につながります。
PostgreSQL では特に追記型アーキテクチャに起因するデータ構造の劣化が発生しやすく、不要領域の回収や断片化の解消を行う必要があります。

（1）VACUUM による不要領域の回収

追記型アーキテクチャにより、更新・削除が繰り返される表では、不要領域が増大し、オブジェクトファイルのサイズが肥大します。そのため、不要領域を VACUUM で回収します。

1）VACUUM のモード

VACUUM には、通常の VACUUM と VACUMM FULL の 2 つのモードがあります。

モード	特徴
VACUUM	不要領域を再利用可能領域に変更します。 ・オブジェクトのサイズは変わりません。 ・対象表への SELECT、INSERT、UPDATE、DELETE 処理は可能です。
VACUUM FULL	不要領域を完全に取り除き、ファイルサイズを小さくします ・実行中は対象表全体を排他ロックするため、すべての SQL は実行できません。 ・処理負荷は高く、他の処理がかなり低速になる可能性があります。

☞「VACUUM による不要領域の回収」（1-17）

2）VACUUM の実行コマンド

対象の表を指定して VACUUM コマンドを実行します。
表名を省略した場合、接続中のデータベースに存在するすべてのオブジェクトが対象になります。

構文

```
VACUUM  [ FULL ] [ VERBOSE ] [ ANALYZE ] [表名];
```

VERBOSE 詳細な報告を出力します

ANALYZE　ANALYZE を同時に実行する場合に指定します。

☞「統計情報の取得」 (6-13)

■VACUUM 実行例

```
postgres=# VACUUM sample;
VACUUM

/*sample テーブルをバキュームした際の詳細な報告を出力*/
postgres=# VACUUM VERBOSE sample;
INFO:  vacuuming "postgres.public.sample"
INFO:  finished vacuuming "postgres.public.sample": index scans: 0
pages: 0 removed, 0 remain, 0 scanned (100.00% of total)
 :
index scan not needed: 0 pages from table (100.00% of total) had 0 dead item identifiers removed
avg read rate: 142.045 MB/s, avg write rate: 0.000 MB/s
buffer usage: 12 hits, 1 misses, 0 dirtied
WAL usage: 1 records, 0 full page images, 188 bytes
system usage: CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s
VACUUM
```

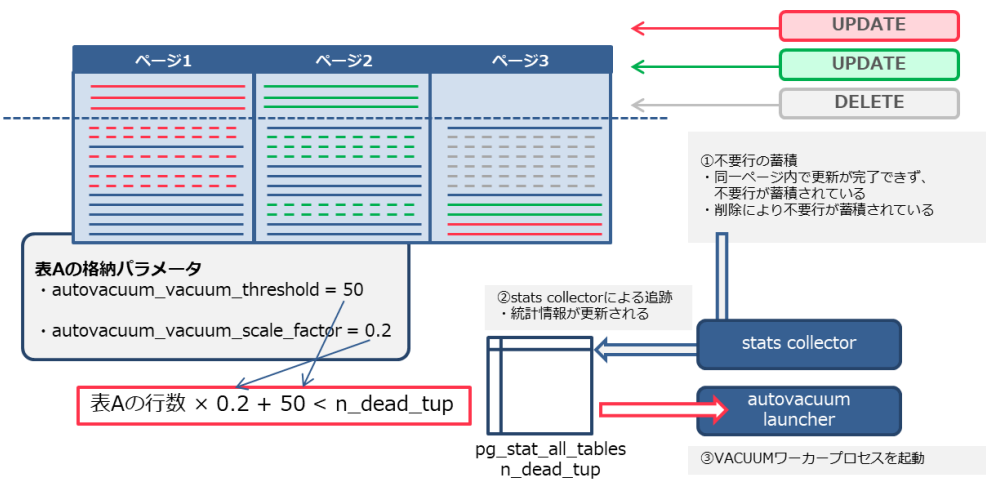
VACUUM FULL は不要領域をディスクに解放してファイルサイズを縮小できますが、将来その表がまた肥大化するのであれば意味がありません。また、多くのシステムで VACUUM FULL による排他ロックは許容されません。
可能な限り VACUUM FULL の使用を避け、適切な頻度で VACUUM を実行することが推奨されます。

（2）自動 VACUUM

自動 VACUUM は、表の変更量を定期的にチェックし、その量がしきい値を超えると（通常の）VACUUM と ANALYZE を自動実行する機能です。

1）自動 VACUUM の動作

自動 VACUUM を有効化（デフォルト）している場合、autovacuum launcher プロセスによって定期的に表の更新量がチェックされ、更新された行数がしきい値を上回っている場合にその表への VACUUM が行われます。



2）VACUUM 実行に関する推奨

PostgreSQL では VACUUM 処理は避けることはできませんが、以下のように VACUUM による負荷を抑えながら実行することができます。

・表単位に FILLFACTOR を設定し、不要行の生成を抑制します。

・一回あたりの VACUUM の負荷を低く抑えるため、自動 VACUUM 頻度を表単位に設定し、不要行が過度に蓄積されることを防ぎます。

・VACUUM の負荷がオンライン業務に影響することを防ぐため、自動 VACUUM を無効化し、業務の負荷が低い時間帯に手動で VACUUM します。

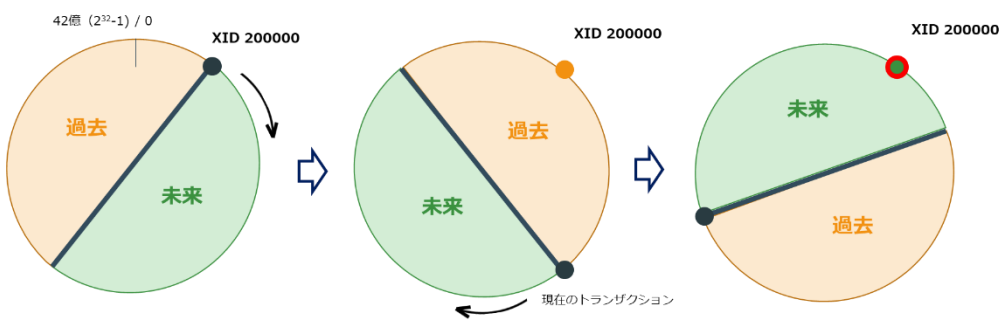
（3）トランザクション ID 周回問題を回避する VACUUM FREEZE

PostgreSQL ではトランザクションを 32bit(約 42 億)のトランザクション ID で管理しています。データベースを長期間運用しているとトランザクション ID が周回し、データベースが利用できない状態になります。これをトランザクション ID 周回問題と呼びます。トランザクション ID 周回問題を回避するためには、VACUUM FREZZE の定期な実行が必要です。VACUUM FREEZE はトランザクション ID が 1.5 億から 2 億進むごとに自動的に実行されます。しかし、トランザクションが閉じられないまま運用されていると、VACUUM FREEZE が阻害されることがあります。この場合、以下のような警告が発生するため、確認でき次第対処します。

■VACUUM FREEZE が実行されていない警告

```
WARNING:  oldest xmin is far in the past
HINT:  Close open transactions soon to avoid wraparound problems.
You might also need to commit or roll back old prepared transactions, or drop stale replication
slots.
```

■トランザクションの周回問題と VACUUM FREEZE



トランザクション ID が周回し、未来のトランザクション ID になると対象レコードが見えなくなる。

VACUUM FREEZE を実行することで対象レコードが見えるようになる。

（4）CLUSTER

表へのアクセスで主に索引スキャンが使われ、かつアクセス対象ブロックが集中しやすいようなワークロードにおいて、行データの物理的な並び順が検索パフォーマンスに影響する場合があります。
PostgreSQL では更新を繰り返すことで行データの物理的な並び順が変動するため、意図した順序でデータを並べ替える CLUSTER コマンドが提供されています。

1）CLUSTER の実行コマンド

対象の表および索引を指定して CLUSTER コマンドを実行します。

構文

```
CLUSTER 表名 [USING 索引名 ];
```

■CLUSTER 実行例

```
postgres=# ¥d sample
                        Table ”public.sample”
 Column |            Type             | Collation | Nullable | Default
--------+-----------------------------+-----------+----------+---------
 a      | integer                     |           | not null |
 b      | text                        |           |          |
 c      | timestamp without time zone |           |          |
Indexes:
    ”sample_pkey” PRIMARY KEY, btree (a)

postgres=# CLUSTER sample USING sample_pkey;
CLUSTER
```

2）CLUSTER の必要性

CLUSTER コマンドは VACUUM　FULL と同様に表全体の排他ロックを必要とします。
オンライン業務の性能を最優先とし、夜間にメンテナンス時間を設けられるような場合に限り
CLUSTER を実行し最適なパフォーマンスを得ることを検討します。