FUJITSU

Fujitsu
Enterprise Postgres 15

運用ガイド

Linux

J2UL-2840-01PJZ0(00)
2023年4月

まえがき

本書の目的

Fujitsu Enterprise Postgresは、PostgreSQLの機能を拡張し、Linuxプラットフォームで動作するデータベースシステムです。

本書は、Fujitsu Enterprise Postgresの運用ガイドです。

本書の読者

本書は、Fujitsu Enterprise Postgresを導入、運用される方を対象としています。

なお、本書は、以下についての一般的な知識があることを前提に書かれています。

- PostgreSQL
- SQL
- Linux

本書の構成

本書の構成と内容は以下のとおりです。

第1章 Fujitsu Enterprise Postgresの運用

Fujitsu Enterprise Postgresの運用について説明しています。

第2章 インスタンスの起動とデータベースの作成

Fujitsu Enterprise Postgresのインスタンスの起動とデータベースの作成について説明しています。

第3章 データベースのバックアップ

データベースのバックアップについて説明しています。

第4章 Secure Sockets Layerによる安全な通信の構成

クライアントとサーバ間の通信データの暗号化について説明しています。

第5章 透過的データ暗号化による格納データの保護

データベースに格納するデータの暗号化について説明しています。

第6章 鍵管理システムをキーストアとして使用する場合の透過的データ暗号化の運用

鍵管理システムをキーストアとして使用する場合の透過的データ暗号化の運用について説明しています。

第7章 データ秘匿化

データ秘匿化機能について説明しています。

第8章 定期的な運用操作

Fujitsu Enterprise Postgresの定期的なデータベースの運用について説明しています。

第9章 WebAdminによるストリーミングレプリケーション

WebAdminによるストリーミングレプリケーションクラスタの作成方法について説明しています。

第10章 インメモリ機能の導入と運用

インメモリ機能の導入と運用について説明しています。

第11章 パラレルクエリ

パラレルクエリを実行する場合の考慮点について説明しています。

第12章 高速データロード機能

高速データロード機能の導入と利用方法について説明しています。

第13章 Global Meta Cache機能

Global Meta Cache機能の利用方法について説明しています。

第14章 Local Meta Cache制限機能

Local Meta Cache制限機能の利用方法について説明しています。

第15章 コピーコマンドを使用したバックアップ/リカバリ

ユーザーが作成したコピーコマンドによるバックアップおよびリカバリについて説明しています。

第16章 異常時の対処

ディスク障害やデータ破壊が発生した場合のリカバリについて説明しています。

付録A パラメータ

Fujitsu Enterprise Postgresのパラメータについて説明しています。

付録B システム管理関数

Fujitsu Enterprise Postgresのシステム管理関数について説明しています。

付録C システムビュー

Fujitsu Enterprise Postgresのシステムビューについて説明しています。

付録D 透過的データ暗号化機能が利用するテーブル

透過的データ暗号化機能が利用するテーブルについて説明しています。

付録E データ秘匿化機能が利用するテーブル

データ秘匿化機能が利用するテーブルについて説明しています。

付録F 高速データロード機能が利用するテーブル

高速データロード機能が利用するテーブルについて説明しています。

付録G WebAdminのWebサーバ機能の起動と停止

WebAdmin(Webサーバ機能)の起動と停止について説明しています。

付録H WebAdminウォレット

WebAdminウォレットの使い方について説明しています。

付録I WebAdminで使用できない文字

WebAdminで使用できない文字を説明しています。

付録J 障害調査情報の採取

初期調査のための情報の採取方法について説明しています。

輸出管理規制について

本ドキュメントを輸出または第三者へ提供する場合は、お客様が居住する国および米国輸出管理関連法規等の規制をご確認のうえ、必要な手続きをおとりください。

出版年月および版数

2023年　4月　初版

著作権

目　次

第1章 Fujitsu Enterprise Postgresの運用

Fujitsu Enterprise Postgresの運用について説明します。

1.1 運用方法の種類

Fujitsu Enterprise Postgresの運用管理には、以下の2つの方法があります。

- GUIツールを使用した運用管理
- コマンドを使用した運用管理

参照

データベース多重化機能を利用してデータベース多重化運用を行う場合は、“クラスタ運用ガイド(データベース多重化編)”を参照してください。

GUIツールを使用した運用管理

運用管理には、WebAdminを使用します。

- WebAdminによる管理

  従来、データベースを運用する際に必要不可欠であった、煩雑な環境設定やバックアップ／リカバリの複雑な運用設計を行う必要がありません。データベースの専門知識がなくても、簡単に、確実に、データベースの状態監視、ストリーミングレプリケーションクラスタの作成、データベースのバックアップ、およびリストアを運用することができます。

コマンドを使用した高度な運用管理

コマンドを使用してデータベースの詳細な設定や運用操作、運用管理ができます。

注意

- WebAdminとサーバコマンドを以下のように組み合わせて運用することはできません。
  - WebAdminで作成したインスタンスのコマンドによる運用
  - WebAdminでバックアップしたデータベースのコマンドによるリカバリ

  ただし、WebAdminで作成したインスタンスにおいてpgx_dmpallコマンドによりバックアップを取得することが可能です。また、pgx_dmpallコマンドで取得したバックアップを使用してWebAdminでリカバリすることが可能です。
- initdbコマンドで作成したインスタンスをWebAdminで運用するためには、WebAdminでインスタンスのインポートを行う必要があります。

各フェーズで利用する機能

GUIによる運用とコマンドによる運用のそれぞれについて、各フェーズで利用する機能を以下に示します。

運用		GUIによる運用	コマンドによる運用
セットアップ	インスタンスの作成	WebAdminを使用します。 サーバマシンの能力や、WebAdminによる運用に最適なパラメータが自動的に設定されます。	initdbコマンドを使用し、設定ファイルを直接編集します。
	スタンバイインスタンスの作成	WebAdminを使用します。 WebAdminでソースインスタンスの	pg_basebackupコマンドを使用し、スタンバイインスタンスを作成します。

運用		GUIによる運用	コマンドによる運用
		ベースバックアップを行い、スタンバイインスタンスを作成します。	
	設定ファイルの変更	WebAdminを使用します。	設定ファイルを直接編集します。
インスタンスの起動と停止		WebAdminを使用します。	pg_ctlコマンドを使用します。
データベースの作成		なし	psqlコマンドやアプリケーションからDDL文を指定して定義します。
データベースのバックアップ		WebAdmin、または、pgx_dmpallコマンドを使用します。	pgx_dmpallコマンドを使用することを推奨します。最新のデータベースへのリカバリを行うことができます。
データベースのリカバリ		WebAdminを使用します。	pgx_dmpallコマンドで取得したバックアップを用いる場合にはpgx_rcvallコマンドを使用します。
モニタリング	データベースの異常	WebAdminの画面で状態を確認できます。	データベースサーバのログに出力されるメッセージを監視します。
	ディスク容量	WebAdminの画面で状態を確認できます。空き容量が20%を下回ると警告を表示します。	OSのdfコマンドなどを使用して監視します。
	コネクション状況	なし	psqlやアプリケーションから標準統計情報ビューのpg_stat_activityを参照することで確認します。

1.2 WebAdminの起動方法

WebAdminの起動方法とログイン方法について説明します。

1.2.1 WebAdminへのログイン

WebAdminへのログイン方法を説明します。

WebAdminの利用環境

WebAdminを使用するには、以下のブラウザの使用を推奨します。

- Microsoft Edge (Build41以降)

WebAdminはFirefoxやChromeなどの他のブラウザでも使用できますが、外観は若干異なる場合があります。

WebAdminの起動URL

ブラウザのURLにWebAdmin画面の起動URLを、以下の形式で指定します。

```
http://ホスト名またはIPアドレス:ポート番号/
```

- ホスト名またはIPアドレス:WebAdminをインストールしたサーバのホスト名、またはIPアドレス
- ポート番号:WebAdminのポート番号。デフォルトのポート番号は27515です。

例

サーバのIPアドレスが“192.0.2.0”、ポート番号が“27515”の場合

```
http://192.0.2.0:27515/
```

起動画面が表示されます。この画面からWebAdminを起動、および製品マニュアルにアクセスすることができます。

注意

- WebAdminを利用するには、あらかじめWebAdminのWebサーバ機能を起動しておく必要があります。
- WebAdminのWebサーバ機能の起動方法の詳細については、“付録G WebAdminのWebサーバ機能の起動と停止”を参照してください。

WebAdminへのログイン

起動画面の“WebAdminを起動する”をクリックすると、WebAdminが起動し、ログイン画面が表示されます。ログイン画面から、WebAdminにログインできます。

ログイン時には、以下を指定します。

- 「ユーザー名」:インスタンス管理者のユーザー名(OSのユーザーアカウント)
- 「パスワード」:ユーザー名のパスワード

ポイント

インスタンス管理者のユーザー名はOSのユーザーアカウントを使用します。詳細は“導入ガイド(サーバ編)”の“インスタンス管理者ユーザーの作成”を参照してください。

1.3 コマンドによる運用操作

以下のコマンドを使用して、データベースを運用管理できます。

- サーバコマンド

  データベースクラスタの作成、データベースを制御するコマンドなどが含まれます。これらのコマンドは、データベースが稼働しているサーバ上で実行できます。

  コマンドを使用するには、環境変数の設定が必要です。

  #### 参照

  ー サーバコマンドの詳細は、“PostgreSQL Documentation”の“Reference”の“PostgreSQL Server Applications”、または“リファレンス”を参照してください。

  ー 環境変数の設定内容については、“導入ガイド(サーバ編)”の“initdbコマンドを使用する場合”において、インスタンスの作成手順の“環境変数の設定”を参照してください。

- クライアントコマンド

  psqlコマンド、データベースクラスタをスクリプトファイルへ抽出するコマンドなどが含まれます。これらのコマンドは、データベースに接続できるクライアント上、またはデータベースが稼動しているサーバ上で実行できます。

  コマンドを使用するには、環境変数の設定が必要です。

参照

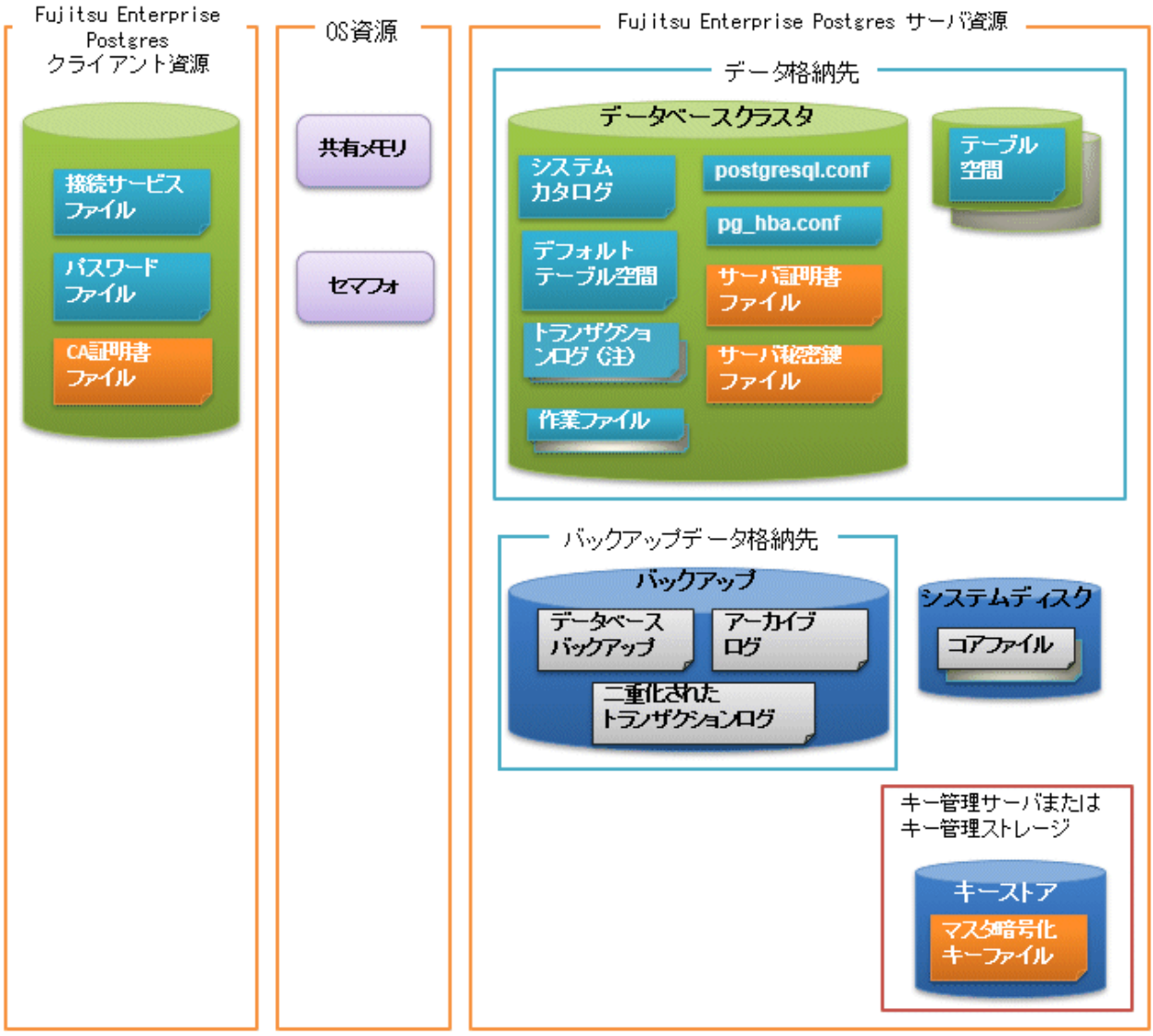
- クライアントコマンドの詳細は、“PostgreSQL Documentation”の“Reference”の“PostgreSQL Client Applications”または、“リファレンス”を参照してください。
- 環境変数の設定内容については、“導入ガイド(クライアント編)”の“環境変数の設定”を参照してください。

1.4 Fujitsu Enterprise Postgresの動作環境

Fujitsu Enterprise Postgresの動作環境、およびファイル構成について説明します。

1.4.1 動作環境

Fujitsu Enterprise Postgresの動作環境の構成を、以下の図に示します。また、OS資源、およびFujitsu Enterprise Postgres資源の役割を、以下の各表に示します。



注）I/O負荷分散したい場合は、トランザクションログをデータ格納先とは別のディスクに配置することもできます。

表1.1 OS資源

種類	役割
共有メモリ	データベースプロセスが外部との情報交換に使用します。
セマフォ	

表1.2 Fujitsu Enterprise Postgresクライアント資源

種類	役割
接続サービスファイル	Fujitsu Enterprise Postgresに接続するためのホスト名、ユーザーID、パスワードなどの情報を指定するファイルです。
パスワードファイル	Fujitsu Enterprise Postgresに接続するためのパスワードを安全に管理するためのファイルです。
CA証明書ファイル	通信データを暗号化する場合、サーバの正当性を検証するために使用するCA(認証局)証明書です。

表1.3 Fujitsu Enterprise Postgresサーバ資源

種類	役割
データベースクラスタ	データベース格納ディスク上にあるデータベース格納領域です。インスタンスで管理されるデータベースの集合体です。
システムカタログ	利用者が作成したデータベースの定義情報、運用情報を含め、システムが動作するうえで必要な情報を保持します。
デフォルトテーブル空間	デフォルトで格納されるテーブルファイル、インデックスファイルを保持します。
トランザクションログ	クラッシュリカバリ、ロールバックに備えたログ情報を保持します。 WAL(Write-Ahead Log)と同義です。
作業ファイル	アプリケーションの実行、コマンドの実行時に使用する作業ファイルです。
postgresql.conf	Fujitsu Enterprise Postgresの動作環境を規定する各種情報を保持します。
pg_hba.conf	Fujitsu Enterprise Postgresがクライアントをクライアントホスト単位で認証するために使用するファイルです。
サーバ証明書ファイル	通信データを暗号化し、サーバ認証を行う場合に、サーバ証明書情報を保持します。
サーバ秘密鍵ファイル	通信データを暗号化し、サーバ認証を行う場合に、サーバ秘密鍵情報を保持します。
テーブル空間	データベースクラスタとは別領域に、テーブルファイル、インデックスファイルを保持します。テーブル空間は、データベースクラスタ配下以外を指定します。
バックアップ	ディスク障害などの異常時に、データベースを復旧するために必要なデータが格納されています。
データベースバックアップ	データベースのバックアップデータを保持します。
アーカイブログ	リカバリに備えたログ情報を保持します。
二重化されたトランザクションログ(二重化WAL)	pgx_dmpallコマンドまたはWebAdminを使用したバックアップ・リカバリ運用を行う場合に、データベースクラスタとトランザクションログの両方が壊れた場合でも、データベースクラスタを異常発生の直前の状態に復旧することを可能とするために二重化されたトランザクションログです。
コアファイル	Fujitsu Enterprise Postgresのプロセスで異常が発生した場合に出力するFujitsu Enterprise Postgresプロセスのコアファイルです。
キー管理サーバまたはキー管理ストレージ	マスタ暗号化キーファイルを配置するサーバまたはストレージです。
マスタ暗号化キーファイル	格納データを暗号化する場合に、マスタ暗号化キーを保持します。マスタ暗号化キーファイルは、キー管理サーバまたはキー管理ストレージ上で管理します。

1.4.2 ファイル構成

Fujitsu Enterprise Postgresは、データベースを制御および格納するため、以下のファイルから構成されています。1つのインスタンスにおける、これらのファイルの数や配置位置などの関係を、以下に示します。

表1.4 1つのインスタンスにおけるファイル数と配置位置の指定方法

ファイル種別	必須	個数	配置位置の指定方法
プログラムファイル	○	複数	/opt/fsepv<*x*>server64 “<*x*>”は、製品のバージョンを示します。

ファイル種別	必須	個数	配置位置の指定方法
データベースクラスタ	○	1	WebAdminまたはサーバコマンドで指定します。
テーブル空間	○	複数	データベースクラスタ配下以外に、DDL文で指定します。
バックアップ	○	複数	WebAdminまたはサーバコマンドで指定します。
コアファイル	○	複数	サーバコマンドまたはpostgresql.confで指定します。
サーバ証明書ファイル(注)	―	1	postgresql.confで指定します。
サーバ秘密鍵ファイル(注)	―	1	postgresql.confで指定します。
マスタ暗号化キーファイル(注)	―	1	postgresql.confでキーストアとして作成先のディレクトリを指定します。
接続サービスファイル(注)	―	1	環境変数で指定します。
パスワードファイル(注)	―	1	環境変数で指定します。
CA証明書ファイル(注)	―	1	環境変数で指定します。

○: 必須、―: 選択

注) それぞれの該当機能を使用する場合に手動で設定します。

注意

- テーブル空間をネットワーク上のストレージデバイスに作成する場合を除いて、Fujitsu Enterprise Postgresで使用するファイルを、ネットワーク経由でマウントしたディレクトリに配置しないでください。
  例えば、NFS (Network File System) や CIFS(Common Internet File System) などが該当します。
  これは、ネットワークに異常が発生した場合にデータベースがハングする場合があるためです。
- ウィルス対策ソフトを使用している場合、Fujitsu Enterprise Postgresを構成するすべてのファイルがウィルススキャンの対象外となるようにディレクトリに対するスキャンの除外設定を行ってください。また、Fujitsu Enterprise Postgresを構成するファイルに対してウィルススキャンを行う場合は、Fujitsu Enterprise Postgresを停止し、Fujitsu Enterprise Postgresを利用した業務が動作していない状態で実行してください。

1.5 運用に利用するアプリケーションの互換に関する注意事項

Fujitsu Enterprise Postgresをバージョンアップする際、機能改善や機能拡張に伴ってアプリケーションに影響するような変更が発生する場合があります。

したがって、アプリケーションを開発する際には、今後新しいバージョンのFujitsu Enterprise Postgresにアップグレードする場合にも互換性を維持できるように注意して作成してください。

参照

詳細は、“アプリケーション開発ガイド”の“アプリケーションの互換に関する注意事項”を参照してください。

第2章 インスタンスの起動とデータベースの作成

本章では、インスタンスの起動からデータベースの作成まで、基本的な操作に関して説明します。

2.1 インスタンスの起動と停止

インスタンスの起動と停止について説明します。

ポイント

データベースサーバのOSの起動・停止に連動してインスタンスの自動起動・停止を行う設定を変更する場合は、“導入ガイド(サーバ編)”の“インスタンスの自動起動・停止の設定”を参照してください。

注意

インスタンスの停止にImmediateモードで停止した場合やインスタンスが異常終了した場合は、収集している統計情報が初期化されます。統計情報の初期化に備えて、SELECT文を使用して、定期的に統計情報を採取するなどの実施を検討してください。統計情報の詳細については“PostgreSQL Documentation”の“Server Administration”の“The Statistics Collector”を参照してください。

2.1.1 WebAdminを使用する場合

WebAdminを使用して、インスタンスの起動、停止、および稼働状態の確認を行うことができます。

インスタンスの起動

インスタンスの起動は、WebAdminの[インスタンス]タブで行います。

インスタンスが停止中の場合は画面に  が表示されます。

停止中のインスタンスを起動する場合は、  をクリックしてください。

インスタンスの停止

インスタンスの停止は、WebAdminの[インスタンス]タブで行います。

インスタンスが起動中の場合は画面に  が表示されます。

起動中のインスタンスを停止する場合は、  をクリックしてください。

停止モード

インスタンスを停止するモードを選択します。各モードの動作を以下に示します。

停止モード	接続されているクライアント	実行中のコマンドによるバックアップ
Smartモード(注)	接続されているすべてのクライアントの切断を待ちます。	実行中のコマンドによるバックアップの終了を待ちます。

停止モード	接続されているクライアント	実行中のコマンドによるバックアップ
Fastモード	すべての実行中のトランザクションをロールバックし、クライアントとの接続を強制的に切断します。	実行中のコマンドによるバックアップを終了させます。
Immediateモード	すべてのサーバプロセスを即座に中断します。次回起動時にクラッシュリカバリ処理が実行されます。	
強制終了モード	SIGKILLをプロセスに送信し、実行中のすべてのトランザクションを中止します。これにより、次回再起動時にクラッシュリカバリ処理が実行されます。	

注) Smartモードで停止処理中になった後、即座に停止したい場合は、以下の手順で停止させてください。

1. WebAdminのWebサーバ機能を再起動します。
2. [インスタンス]タブでをクリックします。
3. [インスタンス]タブのをクリックし、Immediateモードを選択してインスタンスを停止します。

インスタンスの稼働状態の確認

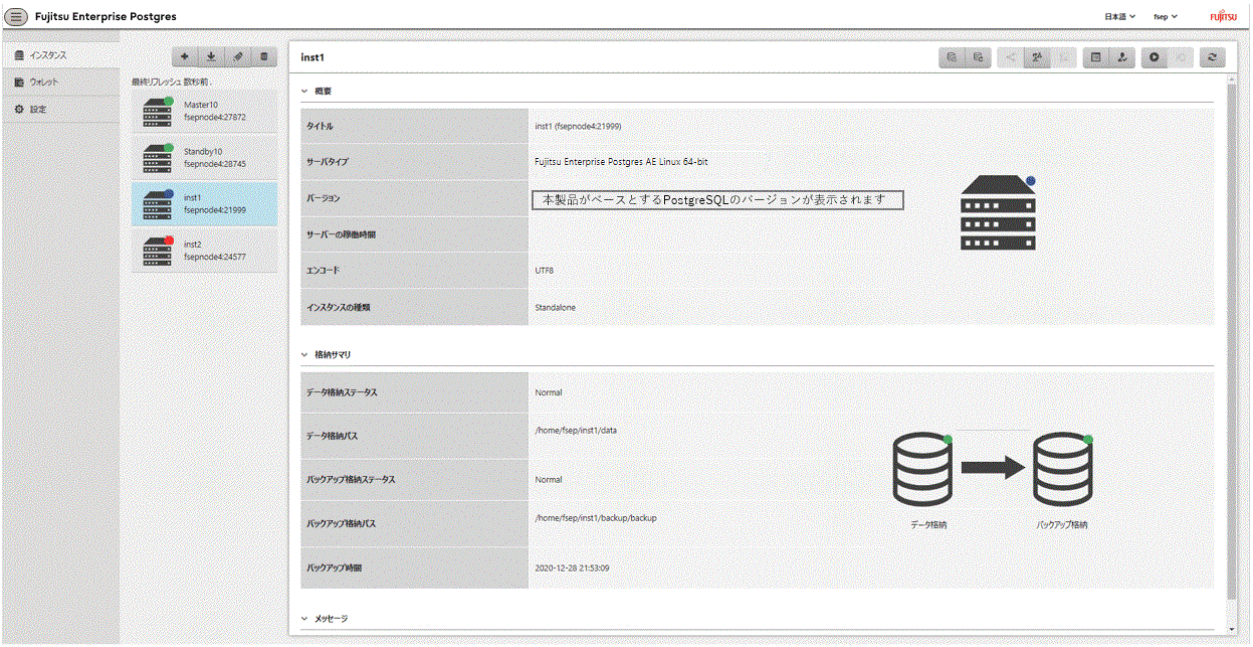
インスタンスの稼働状態は、[インスタンス]タブで確認できます。

リソースの状態が以下の状態マークで示されます。

状態マーク	意味
	リソースは正常に作動中
	リソースは停止中
	リソースにエラー発生
	リソースで作業が進行中、または状態の確認中
	リソースの操作状況が最適ではなく、介入が必要

インスタンスが異常終了した際には、停止となった原因を取り除き、WebAdminからインスタンスを起動してください。

図2.1 稼働状態の状態マークの例



注意

- WebAdminを操作しているとき、  をクリックするとステータスを変更できます。WebAdminが最新の操作状況やインスタンスリソースの情報をサーバから受け取り、反映します。
- サーバとの通信状態が異常となった場合、WebAdminが無応答になることがあります。その場合、ブラウザを閉じて、再度ログインしてください。解決できない場合、サーバのシステムログを確認し、通信に異常が発生していないか確認してください。
- インスタンスの起動中に出力される以下のメッセージは、起動処理の正常な動作によって出力されるメッセージのため、ユーザーが意識する必要はありません。

```
FATAL:  the database system is starting up
```

2.1.2 サーバコマンドを使用する場合

サーバコマンドを使用して、インスタンスの起動、停止、および稼働状態の確認を行うことができます。

サーバコマンドを使用するにあたっては、環境変数を設定してください。

参照

環境変数の設定内容については、“導入ガイド(サーバ編)”の“initdbコマンドを使用する場合”において、インスタンスの作成手順の“環境変数の設定”を参照してください。

インスタンスの起動

pg_ctlコマンドを使用して、インスタンスを起動します。

pg_ctlコマンドには、以下を指定します。

- モードは、startを指定します。
- -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

以下の場合に“FATAL: データベースシステムは起動処理中です(11189)”のメッセージが出力されることがあります。

- インスタンスの起動途中にアプリケーションやコマンド、またはプロセスがデータベースに接続した場合
- -Wオプションを指定せずにしてインスタンスを起動した場合

これは、pg_ctlコマンドがインスタンスのプロセスが起動を完了したかどうかを確認するために出力しています。
そのため、他にデータベースに接続するアプリケーションやコマンド、プロセスが存在していない場合は、このメッセージを無視してください。

例

```
> pg_ctl start -D /database/inst1
```

注意

-Wオプションを指定した場合、インスタンスの起動の完了を待たずにコマンドが復帰します。そのため、インスタンスの起動が正常に完了したのか、失敗したのかが分からない場合があります。

インスタンスの停止

pg_ctlコマンドを使用して、インスタンスを停止します。

pg_ctlコマンドには、以下を指定します。

- モードは、stopを指定します。
- -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

例

```
> pg_ctl stop -D /database/inst1
```

インスタンスの稼働状態の確認

pg_ctlコマンドを使用して、インスタンスの稼働状態を確認します。

pg_ctlコマンドには、以下を指定します。

- モードは、statusを指定します。
- -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

例

【起動状態の場合】

```
> pg_ctl status -D /database/inst1
pg_ctl: サーバが動作中です(PID: 1234)
```

【未起動状態の場合】

```
> pg_ctl status -D /database/inst1
pg_ctl: サーバが動作していません
```

参照

pg_ctlコマンドの詳細は、“PostgreSQL Documentation”の“Reference”の“pg_ctl”を参照してください。

2.2 データベースの作成

データベースの作成について説明します。

クライアントコマンドを使用したデータベースの定義について説明します。

以下にサーバ上からの操作例を示します。

1. psqlコマンドを使用して、データベースpostgresに接続します。
   “psql postgres”と実行します。

   ```
   > psql postgres
   psql (<x>) (注1)
   Type "help" for help.
   ```

   注1: <x>には、本製品がベースとするPostgreSQL のバージョンが表示されます。

2. データベースを作成します。
   データベースを作成するために、“CREATE DATABASE データベース名;”文を実行します。

```
postgres=# CREATE DATABASE db01;
CREATE DATABASE
```

3. データベースが作成されたことを確認します。
   “¥l+”を実行し、2で作成したデータベース名が表示されることを確認します。

```
postgres=# ¥l+
```

4. データベースpostgresと切断します。
   psqlコマンドを終了するために、“¥q”を実行します。

```
postgres=# ¥q
```

なお、createdbコマンドを使用してデータベースを作成することもできます。

参照

createdbコマンドを使用したデータベースの作成については、“PostgreSQL Documentation”の“Tutorial”の“Creating a Database”を参照してください。

第3章 データベースのバックアップ

本章では、データベースのバックアップに関して説明します。

バックアップの方法

以下のバックアップ方法でバックアップすることにより、ディスク装置の物理的破損やデータの論理的破損が発生する直前や、バックアップ時点にリカバリできます。

- WebAdminによるバックアップ

  GUIにより直感的な画面操作でバックアップできます。

  リカバリを行うときは、WebAdminでリカバリします。

- pgx_dmpallコマンドによるバックアップ

  スクリプトからpgx_dmpallコマンドを実行して、自動的にバックアップできます。

  自動的にバックアップするためには、OSの自動化ソフトウェアに登録する必要があります。各OSのドキュメントに従って実施してください。

  リカバリを行うときは、pgx_rcvallコマンドでリカバリします。

参考

pgx_dmpallコマンドとpgx_rcvallコマンドでは、ユーザーが作成したコピーコマンドを使用することで、データベースクラスタやテーブル空間に対する任意のバックアップ先へのバックアップおよび任意のバックアップ先からのリカバリを、任意のコピー方法で実施することができます。“第15章 コピーコマンドを使用したバックアップ/リカバリ”を参照してください。

バックアップの目安時間

WebAdminまたはpgx_dmpallコマンドを使用した場合の、バックアップ目安時間の導出式を示します。

バックアップ時間 = データ格納先の使用量 ÷ ディスク書込み性能 × 1.5

- データ格納先の使用量:データ格納先のディスク使用量
- ディスク書込み性能:運用を行うシステム環境における、1秒間あたりに書き込み可能な最大のデータ量(バイト/秒)の実測値
- 1.5:最も時間のかかるディスク書き込み以外の時間を見込んだ係数

pgx_dmpallコマンドでコピーコマンドを使用する場合、バックアップ時間はコピーコマンドの実装内容に依存します。

注意

- スタンバイモードのストリーミングレプリケーションクラスタの一部であるインスタンスは、バックアップできません。
- 選択したバックアップ方法を継続して使用してください。

  各バックアップ方法は、保存するデータ形式などに差異があります。そのため、以下の制約があります。

  - バックアップデータを使用して、異なる方法でリカバリすることはできません。
  - バックアップデータを異なる方法のバックアップデータに移行できません。
- 二重化したWALは、pgx_dmpallコマンドまたはWebAdminを使用するバックアップ・リカバリ運用以外では使用できません。
- データベースに格納するデータを暗号化する場合、キーストアのバックアップやバックアップにおける留意事項があります。詳細は、以下を参照してください。

  - 5.6.4 キーストアのバックアップとリカバリ
  - 5.7 データベースのバックアップとリストア/リカバリ

- テーブル空間を定義した場合、バックアップを行ってください。バックアップを行わなかった場合、リカバリ実行時にテーブル空間のディレクトリ作成が行われず、リカバリが失敗することがあります。リカバリに失敗した場合は、システムログを参照してテーブル空間のディレクトリ作成後、リカバリを再実行してください。

参考

以下の方式によるバックアップも可能です。これらの方法でバックアップをすることにより、バックアップを行った時点に復旧できます。

- SQLによるダンプを用いたバックアップ

  SQLを使用してデータをダンプします。データの移行にも使えるバックアップ方法です。

- ファイルシステムレベルのバックアップ

  インスタンスを停止し、OSコマンドなどを使用してデータベース資源をファイルとしてバックアップする方法です。

- 継続的アーカイブによるバックアップ

  PostgreSQLの標準的なバックアップ方法です。

これらのバックアップ方法の詳細は、“PostgreSQL Documentation”の“Server Administration”の“Backup and Restore”を参照してください。

3.1 定期的なバックアップ

バックアップを定期的に行うことを推奨します。

WebAdminまたはpgx_dmpallコマンドを使用して定期的にバックアップすることにより、以下の効果があります。

- 不要になったアーカイブログ（バックアップデータ格納先にコピーされたトランザクションログ）が削除されるため、ディスク使用率を抑えることができます。また、障害発生時にリカバリ時間を最小限にできます。

バックアップ周期

定期的にバックアップを行う間隔のことを、バックアップ周期と呼びます。たとえば、毎朝バックアップを行う場合、バックアップ周期は1日となります。
バックアップ周期は運用する業務内容によって異なりますが、Fujitsu Enterprise Postgresでは、1日単位で1回以上のバックアップ周期で運用することを推奨します。

3.2 バックアップ方法

データベースのバックアップ方法について説明します。

3.2.1 WebAdminを使用する場合

WebAdminを使用して、バックアップの実行およびバックアップ状態の確認を行うことができます。

注意

- インスタンスに対してバックアップが無効になっている場合、インスタンスはバックアップまたはリストアできません。詳細は“導入ガイド(サーバ編)”の“インスタンスの作成”の“[バックアップ]”を参照してください。
- データベースに格納するデータを暗号化している場合、キーストアの自動オープンを有効化してから行う必要があります。詳細は“5.6.3 キーストアの自動オープンの有効化”を参照してください。
- WebAdminでは「データ格納先」、「バックアップデータ格納先」、「トランザクションログ格納先」を示すために、それぞれ“データ格納パス”、“バックアップ格納パス”、“トランザクションログのパス”というラベルを使用しています。このマニュアルではこれらの用語は同じ意味で使われています。

バックアップの実行

以下の手順でデータベースをバックアップしてください。

1. バックアップするデータベースの選択

   [インスタンス]タブで、バックアップするインスタンスを選択して  をクリックします。

2. データベースのバックアップ実行

   [バックアップ]ダイアログが表示されます。バックアップを実行する場合は[はい]ボタンをクリックします。バックアップを実行すると、インスタンスが自動的に起動されます。

バックアップ状態

何らかの問題が発生してバックアップに失敗すると、[インスタンス]タブの[データ格納ステータス]または[バックアップ格納ステータス]の横に「Error」と表示されます。また、メッセージリストにエラーメッセージが表示されます。

この場合、バックアップデータは最適化されません。バックアップを実行した際には、必ずバックアップの実行結果を確認してください。バックアップに失敗した場合は、エラーメッセージの右側に[対処]ボタンが表示され、クリックするとエラーの原因を解決するための方法が表示されます。失敗の原因を取り除き、再度バックアップを実行してください。

3.2.2 サーバコマンドを使用する場合

pgx_dmpallコマンド、およびpgx_rcvallコマンドを使用して、バックアップの実行、およびバックアップ状態の確認を行います。

バックアップの準備

バックアップを実施する前に、バックアップの準備を実施する必要があります。

以下の手順で実施してください。

参照

バックアップに必要となるディレクトリの配置や注意事項については、“導入ガイド(サーバ編)”の“資源配置用のディレクトリの準備”を参照してください。

1. バックアップデータ格納ディスクの用意

   データ格納ディスクとは別のディスク装置をバックアップのために用意し、OSの機能を使用してマウントします。

2. バックアップデータ格納先のディレクトリの作成

   空のディレクトリを作成します。

   インスタンス管理者のみがアクセスできるように許可を設定します。

   例

```
# mkdir /backup/inst1
# chown fsepuser:fsepuser /backup/inst1
# chmod 700 /backup/inst1
```

3. バックアップに必要な設定

   インスタンスを停止し、postgresql.confファイルに以下のパラメータを設定してください。postgresql.confファイルを編集後は、インスタンスを起動してください。

パラメータ名	設定値	説明
backup_destination	バックアップデータ格納先のディレクトリ名	バックアップデータを格納するディレクトリ名を指定します。 指定するディレクトリは、インスタンス管理者のみがアクセスできるように権限を設定する必要があります。 なお、バックアップデータ格納先のディレクトリは、データ格納先のディレクトリ、テーブル空間ディレクトリ、およびトランザクションログ格納先のディレクトリの外に配置してください。
archive_mode	on	アーカイブログモードを指定します。 on(行う)を指定してください。
archive_command	'インストールディレクトリ/bin/pgx_walcopy.cmd "%p" "バックアップデータ格納先ディレクトリ/archived_wal/%f"'	トランザクションログを保存するコマンドと格納先パス名を指定します。
archive_library	"(デフォルト)	アーカイブに使用するライブラリを指定します。 "(デフォルト)を指定してください。

   パラメータの詳細については、“付録A パラメータ”、および“PostgreSQL Documentation”の“Server Administration”の“Write Ahead Log”を参照してください。

バックアップの実行(ファイルバックアップ)

pgx_dmpallコマンドを使用して、ファイルバックアップを実行します。また、pgx_dmpallコマンドをOSの自動化ソフトウェアに組み込んでバックアップを実行することもできます。

バックアップデータは、postgresql.confのbackup_destinationパラメータに指定したディレクトリに格納されます。

-Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

例

```
> pgx_dmpall -D /database/inst1
```

注意

バックアップを実行した際には、実行時に取得したデータと、前回取得したデータのバックアップデータが保管されます。

データベースに格納するデータを暗号化している場合、以下を参照してキーストアのバックアップを実施してください。

- 5.6.4 キーストアのバックアップとリカバリ

バックアップ状態

pgx_rcvallコマンドを使用して、バックアップの状況を確認します。

pgx_rcvallコマンドには、以下を指定します。

- -lオプションは、バックアップデータの情報を表示します。
- -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

```
> pgx_rcvall -l -D /database/inst1
Date                    Status          Dir
2022-03-01 13:30:40     COMPLETE        /backup/inst1/2022-03-01_13-30-40
```

何らかの問題が発生してバックアップに失敗していると、システムログにメッセージが出力されます。

この場合、バックアップデータは最適化されません。バックアップを実行した際には、必ずバックアップの実行結果を確認してください。バックアップに失敗した場合は、失敗の原因を取り除き、再度バックアップを実行してください。

参照

pgx_dmpallコマンド、およびpgx_rcvallコマンドの詳細は、“リファレンス”の“pgx_dmpall”、および“pgx_rcvall”を参照してください。

リストアポイントの設定

データベースを、ある決めた時点にリカバリしたい場合に備え、リカバリによってデータを戻したい時点に名前を付けることができます。この時点をリストアポイントと呼び、psqlコマンドにより設定することができます。

アプリケーションの実行前などに設定しておくと、データの内容がいつの時点に戻るかがわかりやすくなり便利です。

リストアポイントは、バックアップ実行後の任意の時点に設定することができます。一方、バックアップの実行前にリストアポイントを設定した場合は、その時点にリカバリすることはできません。リストアポイントはアーカイブログに記録され、アーカイブログはバックアップの実行により破棄されるためです。

例

以下は、psqlコマンドでデータベースに接続し、SQL文を実行してリストアポイントを設定する場合の例です。

ただし、アプリケーションの互換性を維持することを考慮し、SQL文中の関数を直接使用しないようにしてください。詳細は、“アプリケーション開発ガイド”の“アプリケーションの互換に関する注意事項”を参照してください。

```
postgres=# SELECT pg_create_restore_point('batch_20220303_1');
LOG:  restore point "batch_20220303_1" created at 0/20000E8
STATEMENT:  select pg_create_restore_point('batch_20220303_1');
 pg_create_restore_point
-------------------------
 0/20000E8
(1 row)
```

リストアポイントを利用してデータベースをリカバリする場合、“16.3.2 pgx_rcvallコマンドを使用する場合”を参照してください。

注意

- リストアポイントは、データベース内で一意となるように命名してください。以下の例に示すようにリストアポイントを設定する年月日や時刻を付加し、他のリストアポイントと混同しないようにしてください。

  ー YYMMDD_HHMMSS

  - YYMMDD : 年月日を表します。
  - HHMMSS : 時刻を表します。

- 設定したリストアポイントを確認する方法はありません。任意のファイルなどに記録しておいてください。

参照

pg_create_restore_pointの詳細は、“PostgreSQL Documentation”の“Functions and Operators”の“System Administration Functions”を参照してください。

第4章 Secure Sockets Layerによる安全な通信の構成

クライアントとサーバ間の通信データに機密情報を含む場合、通信データを暗号化することで、ネットワーク上の盗聴による脅威から通信データを保護することができます。

4.1 通信データの暗号化のセットアップ

クライアントとサーバ間の通信データを暗号化する場合、以下のセットアップを行ってください。

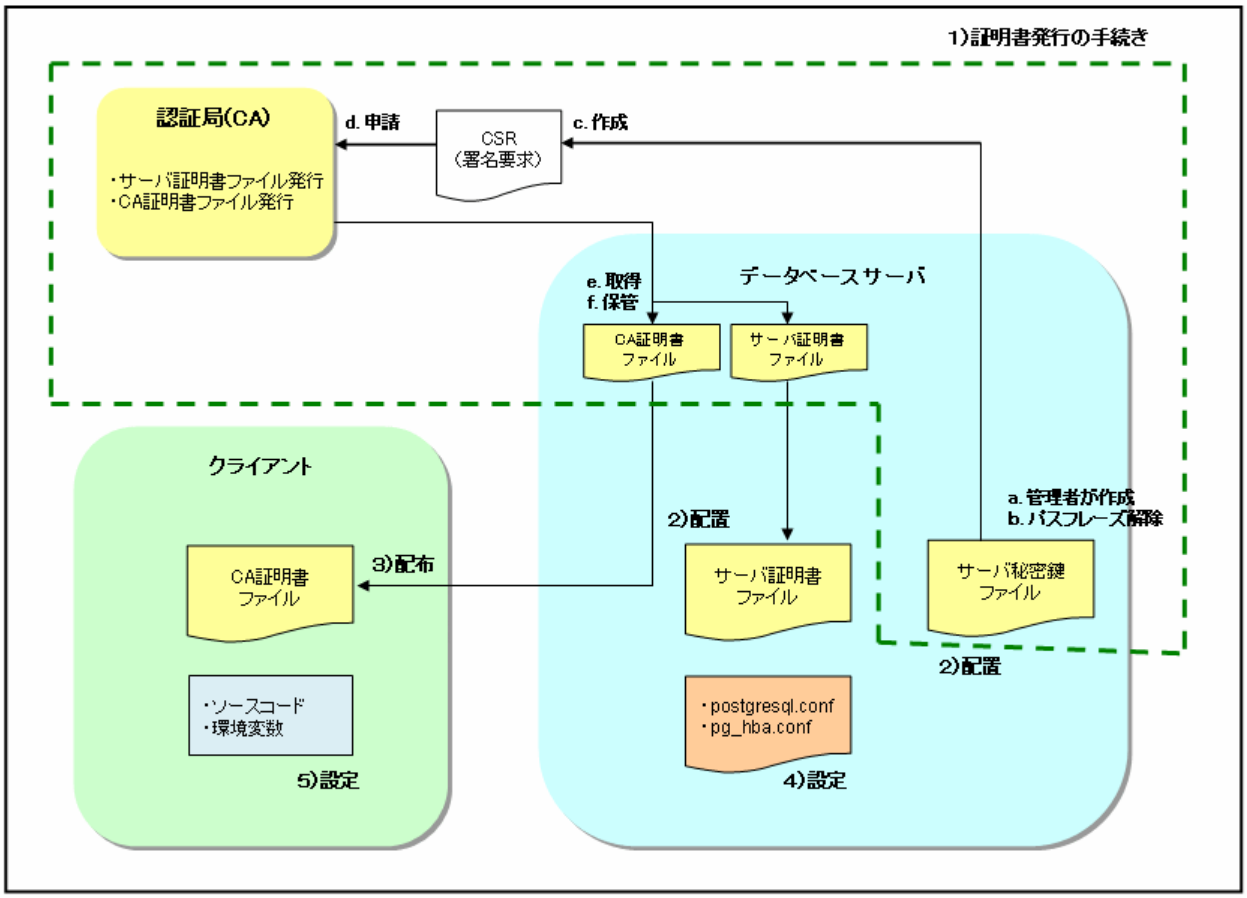
通信データの暗号化を行うと、通信内容を保護することに加え、中間者攻撃（例えばサーバのなりすましによりデータやパスワードを横奪するなど）を防止することができます。

表4.1 セットアップ手順

セットアップ手順
1) 証明書発行の手続き
2) サーバ証明書ファイル、サーバ秘密鍵ファイルの配置
3) CA証明書ファイルのクライアントへの配布
4) データベースサーバの動作環境の設定
5) クライアントの動作環境の設定

通信データの暗号化の環境を以下に示します。

図4.1 通信データの暗号化の環境



4.1.1 証明書発行の手続き

サーバ認証を行う場合は、認証局(CA)に証明書を発行してもらう手続きが必要です。
Fujitsu Enterprise Postgresでは、X.509の規格のPEM形式のファイルをサポートしています。
認証局からDER形式で発行された場合は、opensslコマンドなどのツールを使用してDER形式をPEM形式に変換してください。

以下に手順の概要を示します。詳細は、証明書ファイルの取得先である、公的または独自の認証局(CA)より公開されている手順を参照してください。

a. サーバ秘密鍵ファイルの作成

b. サーバ秘密鍵ファイルのパスフレーズ解除

c. サーバ秘密鍵ファイルからCSR(サーバ証明書を取得するための署名要求)を作成

d. 認証局(CA)へサーバ証明書を申請

e. 認証局(CA)から、サーバ証明書ファイルおよびCA証明書ファイルを取得

f. サーバ証明書ファイルおよびCA証明書ファイルを保管
注) 紛失や破損した場合は、再発行が必要になります。

上記の手順により、以下のファイルが準備できます。

- サーバ秘密鍵ファイル
- サーバ証明書ファイル
- CA証明書ファイル

4.1.2 サーバ証明書ファイル、サーバ秘密鍵ファイルの配置

データベースサーバのローカルディスクにディレクトリを作成し、サーバ証明書ファイル、サーバ秘密鍵ファイルを配置します。
サーバ証明書ファイル、サーバ秘密鍵ファイルのアクセス権は、OSの機能を利用して、データベース管理者にのみ読込み権を設定してください。
また、サーバ証明書ファイルおよびサーバ秘密鍵ファイルは破損に備え、バックアップをして、厳重に管理してください。

4.1.3 CA証明書ファイルのクライアントへの配布

クライアントのローカルディスクにディレクトリを作成し、配布されたCA証明書ファイルを配置します。
CA証明書ファイルを誤って削除などしないように、OSの機能を利用して、読込み権を設定してください。

4.1.4 データベースサーバの動作環境の設定

参照

詳細については、“PostgreSQL Documentation”の“Server Administration”の“Secure TCP/IP Connections with SSL”を参照してください。

4.1.5 クライアントの動作環境の設定

参照

詳細については、アプリケーションの開発環境に応じて、“アプリケーション開発ガイド”の以下の項を参照してください。

- “JDBCドライバ”の“セットアップ”の“通信データを暗号化する場合の設定”
- “C言語用ライブラリ(libpq)”の“セットアップ”の“通信データを暗号化する場合の設定”
- “C言語による埋め込みSQL”の“セットアップ”の“通信データを暗号化する場合の設定”

4.1.6 データベース多重化運用を行う場合

データベース多重化機能を使用して、かつSSLのサーバ証明書を利用した通信を行う場合には、以下のいずれかを行ってください。

- サーバ証明書を1つ作成し、複製して各サーバに配置する
  sslmodeがverify-fullに設定されている場合、すべてのドメイン名をsubjectAltNameに追加する。
- サーバ証明書を各サーバごとに作成する。

参照

クライアント側でのアプリケーションの指定方法については、“アプリケーション開発ガイド”の“アプリケーションの接続先切り替え機能を利用する”を参照してください。

第5章 透過的データ暗号化による格納データの保護

データベースに格納するデータの暗号化について説明します。

参照

暗号化キーの保管場所として外部の鍵管理システムを利用する場合には、“第6章 鍵管理システムをキーストアとして使用する場合の透過的データ暗号化の運用”を参照してください。

5.1 暗号化によるデータの保護

PostgreSQLでは、認証とアクセス制御によって、データベース内のデータは認可されていないデータベースユーザーによるアクセスから保護されます。しかし、データベースサーバの認証とアクセス制御を迂回する攻撃者に対しては、OSファイルは保護されません。

Fujitsu Enterprise PostgresではOSのファイル内のデータが暗号化されているため、たとえそのファイルやディスクが盗まれても、貴重な情報は保護されます。
データベースに格納するデータは、データファイルに書き出されるときに暗号化され、読み出されるときに復号されます。これはインスタンスによって自動的に行われるため、ユーザーやアプリケーションが意識することなく、キーの管理や暗号化/復号の処理を実行できます。これを透過的データ暗号化（TDE: Transparent Data Encryption）と呼びます。

TDEには次の特長があります。

暗号化の仕組み

2層の暗号化キーとキーストア

各テーブル空間には、その中のすべてのデータを暗号化/復号するテーブル空間暗号化キーがあります。テーブル空間暗号化キーは、マスタ暗号化キーで暗号化されて保存されます。
マスタ暗号化キーは、データベースクラスタに1つだけ存在します。利用者が指定するパスフレーズに基づいて暗号化され、キーストアに保存されます。Fujitsu Enterprise Postgresは、ファイルベースのキーストアを提供します。パスフレーズを知らない攻撃者は、キーストアからマスタ暗号化キーを読み出すことはできません。

強力な暗号化アルゴリズムを利用

暗号化アルゴリズムとしてAES（Advanced Encryption Standard）を使用します。AESは2002年に米国連邦政府の標準として採用され、世界中で広く使われています。

記憶領域のゼロ・オーバヘッド

テーブルやインデックス、WALに格納されるデータの大きさは、暗号化しても変わりません。そのため、追加の見積りやディスクは不要です。

暗号化の範囲

指定したテーブル空間内のすべてのユーザーデータ

暗号化を指定する単位はテーブル空間です。暗号化テーブル空間内に作成されるテーブルとインデックス、一時テーブルと一時インデックスの全体が暗号化されます。利用者はどのテーブルや列を暗号化するかを考える必要はありません。

テーブル空間の暗号化の詳細については、“5.4 テーブル空間の暗号化”を参照してください。

バックアップデータ

pgx_dmpallコマンド、およびpg_basebackupコマンドは、OSファイルをコピーすることによりバックアップデータを作成します。そのため、暗号化されたデータのバックアップは暗号化されたままです。バックアップ・メディアが盗まれても、情報は漏えいから保護されます。

WALと一時ファイル

暗号化されたテーブルとインデックスの更新で生成されるWALは、更新対象と同じセキュリティ強度で暗号化されます。大きな結合やソートを実行するときには、暗号化データは一時ファイルにも暗号化された形で書き出されます。

ストリーミングレプリケーションのサポート

ストリーミングレプリケーションと透過的データ暗号化を組み合わせて使うことができます。プライマリサーバで暗号化されたデータとWALは、暗号化されたままスタンバイサーバに転送され、格納されます。

注意

以下については暗号化されません。

- pg_dumpおよびpg_dumpallコマンドの出力ファイル
- COPYコマンドの出力ファイル
- LISTEN/NOTIFYコマンドでやりとりする通知イベントのペイロード
- 暗号化テーブル空間に対するchecksum妥当性検証は、バックアップ処理中およびpg_checksumsユーティリティの使用中は行われません。

5.2 マスタ暗号化キーの設定

透過的データ暗号化を使用するには、キーストアを作成し、マスタ暗号化キーを設定する必要があります。

1. postgresql.confのkeystore_locationパラメータに、キーストアを格納するディレクトリを設定します。

   データベースクラスタごとに異なる場所を指定してください。

   ```
   keystore_location = '/key/store/location'
   ```

   postgresql.confについては、“付録A パラメータ”を参照してください。

   postgresql.confファイルを編集後は、インスタンスを起動、または再起動してください。

   ー WebAdminを使用する場合

   “2.1.1 WebAdminを使用する場合”を参照してインスタンスを再起動します。

   ー pg_ctlコマンドを使用する場合

   pg_ctlコマンドには以下を指定します。

   - モードは、restartを指定します。
   - -Dオプションはデータ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。
   - -wオプションを指定することを推奨します。これにより、インスタンスの起動の完了を待ってからコマンドが復帰します。-wオプションを指定しない場合、インスタンスの起動が正常に完了したのか、失敗したのかが分からない場合があります。

例

```
> pg_ctl restart -w -D /database/inst1
```

2. 以下のようなSQL関数を実行してマスタ暗号化キーを設定します。

   データベースのスーパーユーザーで実行してください。

```
SELECT pgx_set_master_key('passphrase');
```

   passphraseは、今後キーストアをオープンするために使われるパスフレーズです。マスタ暗号化キーはこのパスフレーズによって保護されるため、短く単純で推測されやすい文字列を指定しないよう注意してください。

   pgx_set_master_key関数の詳細は、“B.2 透過的データ暗号化制御関数”を参照してください。

注意

ここで指定したパスフレーズを忘れてしまうと、暗号化されたデータにはアクセスできなくなります。忘れたパスフレーズを取り戻したり、データを復号する方法はありません。決してパスフレーズを忘れないようにしてください。

pgx_set_master_key関数は、キーストア格納ディレクトリにkeystore.ksというファイルを作成します。そして、ランダムなビット列からなるマスタ暗号化キーを生成し、それを指定されたパスフレーズで暗号化してkeystore.ksに格納します。キーストアは、オープンされた状態になります。

5.3 キーストアのオープン

暗号化テーブル空間を作成したり暗号化データにアクセスするには、キーストアをオープンしておく必要があります。キーストアをオープンすると、マスタ暗号化キーがデータベースサーバのメモリにロードされ、暗号化と復号に利用できるようになります。

インスタンスを起動するたびにキーストアをオープンしてください。キーストアをオープンするには、データベースのスーパーユーザーが次のようにSQL関数を実行します。

```
SELECT pgx_open_keystore('passphrase');
```

passphraseはキーストアの作成時に指定したパスフレーズです。

pgx_open_keystore関数の詳細は、“B.2 透過的データ暗号化制御関数”を参照してください。

ただし、以下の場合には、リカバリのために暗号化されたWALを復号する必要があるため、インスタンスを起動する時にパスフレーズの入力が必要になります。この場合は、上述のpgx_open_keystore関数を実行することができません。

- インスタンス起動時にクラッシュリカバリが行われる場合
- 継続的アーカイブによるリカバリを行う場合

上記の場合には、pg_ctlコマンドに--keystore-passphraseオプションを指定して起動してください。パスフレーズ入力を促すプロンプトが以下のように表示されます。

```
> pg_ctl --keystore-passphrase start
パスフレーズを入力してください:
サーバは起動中です
>
```

ポイント

自動オープン・キーストアを使用すると、パスフレーズを入力することなく、インスタンスの起動時に自動的にキーストアをオープンできます。詳細は“5.6.3 キーストアの自動オープンの有効化”を参照してください。

5.4 テーブル空間の暗号化

暗号化テーブル空間を作成するには、事前にキーストアがオープンされている必要があります。

暗号化するテーブル空間を作成するときに、実行時パラメータに暗号化アルゴリズムを設定します。たとえば、暗号化アルゴリズムとしてキー長が256ビットのAESを用い、secure_tablespaceという名前のテーブル空間を作成するには、次のようにします。

```
-- 以降に作成するテーブル空間の暗号化アルゴリズムを指定
SET tablespace_encryption_algorithm = 'AES256';
CREATE TABLESPACE secure_tablespace LOCATION '/My/Data/Dir';
-- 以降に作成するテーブル空間は暗号化しないように指定
SET tablespace_encryption_algorithm = 'none';
```

または

```
CREATE TABLESPACE secure_tablespace LOCATION '/My/Data/Dir' WITH (tablespace_encryption_algorithm = 'AES256' );
```

テーブル空間が空の場合、以下のコマンドを使用して暗号化アルゴリズムを変更できます。

```
ALTER TABLESPACE secure_tablespace SET (tablespace_encryption_algorithm=AES256);
```

空でないテーブル空間に暗号化アルゴリズムを設定しようとすると、エラーが発生します。

暗号化アルゴリズムとして、キー長が128または256ビットのAESを使用できます。256ビットのAESを推奨します。実行時パラメータの指定方法については“付録A パラメータ”を参照してください。

テーブル空間を作成するときにGUCとコマンドラインオプションの両方を提供した場合、コマンドラインオプションが選択されます。

テーブル空間pg_defaultおよびpg_globalを暗号化することはできません。

そして、作成した暗号化テーブル空間にテーブルやインデックスを作成します。暗号化テーブル空間に作成されたリレーションは自動的に暗号化されます。

例

例1: 作成時に暗号化テーブル空間を指定

```
CREATE TABLE my_table (...)
    TABLESPACE secure_tablespace;
```

例2: 作成時にはテーブル空間を明示せず、デフォルト・テーブル空間を使用

```
SET default_tablespace = 'secure_tablespace';
CREATE TABLE my_table (...);
```

一時テーブルと一時インデックスを暗号化する場合も同様です。つまり、TABLESPACE句を明示的に指定するか、またはtemp_tablespacesパラメータに暗号化テーブル空間を列挙し、CREATE TEMPORARY TABLEやCREATE INDEXを実行します。

ポイント

データベースを作成するときに、CREATE DATABASE文のTABLESPACE句に暗号化テーブル空間を指定すると、明示的にテーブル空間を指定しないでそのデータベースに作成されたリレーションは暗号化されます。さらに、システムカタログも暗号化されるので、ユーザー定義関数のソースコードも保護されます。

例:データベース定義文にテーブルスペースを指定

```
CREATE DATABASE DB01 TABLESPACE=SP01 ・・・・・ ;
```

また、システムカタログにはデータの一部も格納されることから、これらのデータも含めて暗号化するために、上記により暗号化テーブル空間を指定してデータベースを作成してください。

5.5 暗号化されているテーブル空間の確認

システムビューpgx_tablespacesは、各テーブル空間が暗号化されているかどうか、および暗号化アルゴリズムについての情報を示します。列の詳細については“C.1 pgx_tablespaces”を参照してください。

次のようなSQL文を実行することで、どのテーブル空間が暗号化されているかを知ることができます。

ただし、アプリケーションの互換性を維持することを考慮し、下記のSQL文中のシステムカタログ(pg_tablespace)を直接参照しないようにしてください。

```
SELECT spcname, spcencalgo
FROM pg_tablespace ts, pgx_tablespaces tsx
WHERE ts.oid = tsx.spctablespace;
```

例

```
postgres=# SELECT spcname, spcencalgo FROM pg_tablespace ts, pgx_tablespaces tsx WHERE ts.oid = tsx.spctablespace;
       spcname       | spcencalgo
---------------------+------------
 pg_default          | none
 pg_global           | none
 secure_tablespace | AES256
(3 rows)
```

参照

アプリケーションの互換性の維持に関する詳細は、“アプリケーション開発ガイド”の“アプリケーションの互換に関する注意事項”を参照してください。

5.6 キーストアの管理

盗難による脅威に備えるための、キーストアやマスタ暗号化キーの管理方法について説明します。

5.6.1 マスタ暗号化キーの変更

同じ暗号化キーを長期間使い続けることは、攻撃者に対して暗号データの解読の機会を与えてしまいます。一定期間ごと、またはキーが危険にさらされたときには、キーを変更することを薦めます。

どのくらいの期間でキーを変更すればよいかについては、暗号化アルゴリズムとキー管理についての業界のベストプラクティスに従ってください。たとえば、米国のNISTが発行している‘NIST Special Publication 800-57’があります。PCI DSSもこれに言及しています。その中では、マスタ暗号化キーは1年ごとに一度変更することが推奨されています。

マスタ暗号化キーを変更するには、最初に設定したときと同じくpgx_set_master_key関数を実行します。詳細は“5.2 マスタ暗号化キーの設定”を参照してください。

マスタ暗号化キーを変更したら、直ちにキーストアをバックアップしてください。

5.6.2 キーストアのパスフレーズの変更

組織のセキュリティ・ポリシーでは通常、パスフレーズを知るセキュリティ管理者が異動や退職で職務から外れるときには、パスフレーズを変更することが求められます。また、万一ソーシャル・エンジニアリングのような手口でパスフレーズが危険にさらされたときには、パスフレーズを変更することをお薦めします。

キーストアのパスフレーズを変更するには、データベースのスーパーユーザーで次のようにSQL関数を実行します。

```
SELECT pgx_set_keystore_passphrase('old_passphrase', 'new_passphrase');
```

パスフレーズを変更したら、直ちにキーストアをバックアップしてください。

pgx_set_keystore_passphrase関数の詳細は、“B.2 透過的データ暗号化制御関数”を参照してください。

5.6.3 キーストアの自動オープンの有効化

自動オープン・キーストアを使用すると、パスフレーズを入力することなく、インスタンスの起動時に自動的にキーストアをオープンできます。キーストアの自動オープンを有効にするには、pgx_keystoreコマンドを実行します。

```
> pgx_keystore --enable-auto-open /key/store/location/keystore.ks
パスフレーズを入力してください:
キーストアの自動オープンが有効になりました
>
```

参照

pgx_keystoreコマンドの詳細については、“リファレンス”の“pgx_keystore”を参照してください。

自動オープンを有効にすると、元のキーストアと同じディレクトリに自動オープン・キーストアが作成されます。自動オープン・キーストアのファイル名はkeystore.aksです。keystore.aksは、keystore.ksの復号された内容を難読化したコピーです。このファイルが存在すると、インスタンスの起動時に、キーストアをオープンするためのパスフレーズを入力する必要はありません。

元のキーストアのファイルkeystore.ksを削除しないでください。マスタ暗号化キーやパスフレーズを変更するために必要です。これらを変更すると、元のキーストアのファイルkeystore.ksからkeystore.aksが再作成されます。

キーストアを格納したディレクトリ、およびkeystore.ksとkeystore.aksは、インスタンスを起動するユーザーのみがアクセスできるように保護してください。

なお、これらのファイルを作成するSQL関数およびコマンドは、インスタンスを起動するユーザーのみがアクセスできるようにファイルの許可モードを設定します。そのため、ファイルをリストアしたときなどに手動で同様の許可モードを設定してください。

例

```
# chown -R fsepuser:fsepuser /key/store/location
# chmod 700 /key/store/location
# chmod 600 /key/store/location/keystore.ks
# chmod 600 /key/store/location/keystore.aks
```

自動オープン・キーストアは、それが作成されたコンピュータでのみオープンします。

キーストアの自動オープンを無効化するには、keystore.aksを削除してください。

注意

- WebAdminを使用してリカバリを行う際には、キーストアの自動オープンを有効にしてから行ってください。
- 暗号化を有効にしたあと、または設定を変更したあとは、“5.7 データベースのバックアップとリストア/リカバリ”を参照して、データベースをバックアップしてください。
- キーストアの格納先には、以下のディレクトリとは別のディレクトリを指定してください。
  - データ格納先
  - テーブル空間格納先
  - トランザクションログ格納先
  - バックアップデータ格納先

5.6.4 キーストアのバックアップとリカバリ

キーストアの破損や消失に備え、次のときにはキーストアをバックアップしてください。ただし、データベースとキーストアは別々の記憶媒体に保管してください。両方を同じ記憶媒体に格納した場合、その記憶媒体が盗まれると、暗号化データを解読されるおそれがあります。自動オープン・キーストアのオープンにはパスフレーズが必要ないため、特に安全な場所に保管してください。

- 最初にマスタ暗号化キーを設定したとき
- マスタ暗号化キーを変更したとき
- データベースをバックアップするとき
- キーストアのパスフレーズを変更したとき

ポイント

キーストアをバックアップするときには、古いキーストアを上書きしないようにしてください。なぜなら、データベースをリカバリするときには、データベースのバックアップを取得した時点のキーストアをリストアする必要があるためです。データベースのバックアップデータが不要になったら、それに対応するキーストアを削除してください。

例

- 2022年3月1日にデータベースとキーストアをバックアップします。

```
> pgx_dmpall -D /database/inst1
> cp -p /key/store/location/keystore.ks /keybackup/keystore_20220301.ks
```

pgx_dmpallコマンドには、以下を指定します。

ー -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

- 2022年3月5日にマスタ暗号化キーを変更し、キーストアをバックアップします。

```
> psql -c "SELECT pgx_set_master_key('passphrase')" postgres
> cp -p /key/store/location/keystore.ks /keybackup/keystore_20220305.ks
```

psqlコマンドには、以下を指定します。

ー -cオプションは、マスタ暗号化キーを設定するSQL関数を指定します。

ー 引数は、接続するデータベース名を指定します。

キーストアが破損または消失した場合には、最新のマスタ暗号化キーが含まれるキーストアをリストアしてください。もし最新のマスタ暗号化キーを含むキーストアがない場合は、データベースのバックアップを取得した時点のキーストアをリストアし、そのデータベースのバックアップからデータベースをリカバリしてください。これにより、キーストアも最新状態にリカバリされます。

例

- 最新のマスタ暗号化キーが含まれる、2022年3月5日時点のキーストアをリストアします。

```
> cp -p /keybackup/keystore_20220305.ks /key/store/location/keystore.ks
```

- 最新のマスタ暗号化キーが含まれるキーストアのバックアップがない場合、2022年3月1日にデータベースとともにバックアップしたキーストアをリストアしてリカバリします。

```
> cp -p /keybackup/keystore_20220301.ks /key/store/location/keystore.ks
> pgx_rcvall -B /backup/inst1 -D /database/inst1 --keystore-passphrase
```

pgx_rcvallコマンドには、以下を指定します。

ー -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

ー -Bオプションは、バックアップデータ格納先のディレクトリを指定します。

ー --keystore-passphraseオプションは、キーストアをオープンするためのパスフレーズの入力を促します。

キーストアをリストアした際には、再度キーストアの自動オープンを有効にしてください。これにより、自動オープン・キーストア(keystore.aks)の内容が、リストアしたキーストアの内容と一致します。

自動オープン・キーストアのファイルkeystore.aksをバックアップしないことをお薦めします。万一、自動オープン・キーストアが格納されたバックアップ媒体とデータベースのバックアップ媒体の両方が盗まれてしまった場合、攻撃者はパスフレーズを知らなくてもデータを読み取れてしまうためです。

自動オープン・キーストアが破損または消失した場合には、再び自動オープンを有効にしてください。これにより、keystore.ksからkeystore.aksが再作成されます。

参照

pgx_rcvallコマンド、およびpgx_dmpallコマンドの詳細は、“リファレンス”の“pgx_rcvall”および“pgx_dmpall”を参照してください。

psqlコマンドの詳細は、“PostgreSQL Documentation”の“Reference”の“psql”を参照してください。

pgx_set_master_key関数の詳細は、“B.2 透過的データ暗号化制御関数”を参照してください。

キーストアの自動オープンの有効化については、“5.6.3 キーストアの自動オープンの有効化”を参照してください。

5.7 データベースのバックアップとリストア/リカバリ

Fujitsu Enterprise Postgresでは、以下に示す5つのバックアップ/リカバリの方法が利用できます。いずれの方法を用いる場合も、キーストアを同時にバックアップしてください。

ただし、データベースとキーストアは別々の記憶媒体に保管してください。両方を同じ記憶媒体に格納した場合、その記憶媒体が盗まれると、暗号化データを解読されるおそれがあります。

WebAdminによるバックアップ・リカバリ

- バックアップ

  WebAdminは、暗号化されたデータをバックアップします。

  データベースのバックアップ後、キーストアをバックアップしてください。

- リカバリ

  データベースのバックアップを取得した時点のキーストアをリストアしてください。詳細は、“5.6.4 キーストアのバックアップとリカバリ”を参照してください。

  “5.6.3 キーストアの自動オープンの有効化”の手順に従い、キーストアの自動オープンを有効にしてください。その後、WebAdminでリカバリを実行してください。

pgx_dmpall、pgx_rcvallコマンドによるバックアップ・リカバリ

- バックアップ

  pgx_dmpallコマンドは、暗号化されたデータをバックアップします。

  データベースのバックアップ後、キーストアをバックアップしてください。

- リカバリ

  データベースのバックアップを取得した時点のキーストアをリストアしてください。

  キーストアの自動オープンは、必要に応じて設定してください。

キーストアの自動オープンを有効にしない場合、--keystore-passphraseオプションを指定して、pgx_rcvallコマンドを実行してください。パスフレーズの入力を促すプロンプトが表示されます。

例

- 2022年3月1日にデータベースとキーストアをバックアップします。

```
> pgx_dmpall -D /database/inst1
> cp -p /key/store/location/keystore.ks /keybackup/keystore_20220301.ks
```

pgx_dmpallコマンドには、以下を指定します。

ー -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

- 2022年3月1日に取得したバックアップから、データベースとキーストアをリカバリします。

```
> cp -p /keybackup/keystore_20220301.ks /key/store/location/keystore.ks
> pgx_keystore --enable-auto-open /key/store/location/keystore.ks （自動オープンを有効にする場合のみ実行）
> pgx_rcvall -B /backup/inst1 -D /database/inst1 --keystore-passphrase
```

pgx_rcvallコマンドには、以下を指定します。

ー -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

ー -Bオプションは、バックアップデータ格納先のディレクトリを指定します。

ー --keystore-passphraseオプションは、キーストアをオープンするためのパスフレーズの入力を促します。

SQLによるダンプとリストア

- バックアップ

pg_dumpコマンド、およびpg_dumpallコマンドが出力するデータは暗号化されません。そのため、“5.8 データベースのインポートとエクスポート”を参照して、OpenSSLなどのコマンドを利用してデータを暗号化してください。

データベースのバックアップ後、キーストアをバックアップしてください。

- リストア

OpenSSLなどのコマンドを利用して、バックアップデータを暗号化した場合、そのデータを復号してください。

pg_dumpallコマンドが出力するデータは、デフォルトではテーブル空間の暗号化の指定を含みます。そのため、psqlコマンドは、テーブル空間を暗号化してリストアします。

ファイルシステムレベルのバックアップとリストア

- バックアップ

インスタンスを停止して、OSのファイルコピーコマンドでデータ格納先のディレクトリやテーブル空間ディレクトリをバックアップします。暗号化テーブル空間のファイルは、暗号化された状態でバックアップされます。

バックアップ実行後、キーストアをバックアップしてください。

- リストア

データベースのバックアップを取得した時点のキーストアをリストアしてください。

インスタンスを停止して、OSのファイルコピーコマンドでデータ格納先のディレクトリやテーブル空間ディレクトリをリストアします。

継続的アーカイブによるバックアップとポイントインタイムリカバリ

- バックアップ

  pg_basebackupコマンドは、暗号化されたデータをバックアップします。

  バックアップ実行後、キーストアをバックアップしてください。

- リカバリ

  データベースのバックアップを取得した時点のキーストアをリストアしてください。

  キーストアの自動オープンは、必要に応じて設定してください。

  キーストアの自動オープンを有効にしない場合、--keystore-passphraseオプションを指定して、pg_ctlコマンドでインスタンスを起動してください。パスフレーズの入力を促すプロンプトが表示されます。

参照

pg_ctlコマンドの詳細は、“PostgreSQL Documentation”の“Reference”の“pg_ctl”を参照してください。

以下のコマンドの詳細は、“PostgreSQL Documentation”の“Reference”のそれぞれの項を参照してください。

- psql
- pg_dump
- pg_basebackup

以下のコマンドの詳細は、“リファレンス”を参照してください。

- pgx_rcvall
- pgx_dmpall
- pg_dumpall

キーストアをリストアした際には、再度キーストアの自動オープンを有効にしてください。これにより、自動オープン・キーストア(keystore.aks)の内容が、リストアしたキーストアの内容と一致します。

キーストアの自動オープンの有効化については、“5.6.3 キーストアの自動オープンの有効化”を参照してください。

5.8 データベースのインポートとエクスポート

COPY TOコマンドが出力するファイルは暗号化されません。そのため、他のシステムにファイルを転送する場合は、OpenSSLのコマンドなどでファイルを暗号化したり、scpやsftpなどで通信中のデータを暗号化してください。

平文のファイルは、不要になったら安全な方法で削除してください。

ファイルを安全に削除するには、以下の方法が利用できます。

- shredコマンド

例

```
# テーブルmy_tableの内容をCSV形式のファイルにエクスポートする。
> psql -c "COPY my_table TO '/tmp/my_table.csv' (FORMAT CSV)" postgres

# エクスポートしたファイルを暗号化する。
> openssl enc -e -aes256 -in my_table.csv -out my_table.csv.enc
（ここで暗号化に使うパスフレーズの入力が求められる）

# 平文のファイルを安全に削除する。
> shred -u -x my_table.csv
（暗号化したファイルを他のシステムに転送する）

# 他のシステムで、暗号化ファイルを復号する。
> openssl enc -d -aes256 -in my_table.csv.enc -out my_table.csv
（ここで復号に使うパスフレーズの入力が求められる）
```

COPY FROMコマンドのインポート先となるテーブルやインデックスが暗号化テーブル空間にある場合、インポートされたデータは自動的に暗号化された形で格納されます。

5.9 既存データの暗号化

既存の暗号化されていないテーブル空間を暗号化することはできません。また、暗号化されたテーブル空間を暗号化しないように変更することもできません。

代替手段として、テーブルやインデックスを別のテーブル空間に移動してください。このためには、次のSQLコマンドが利用できます。

```
ALTER TABLE table_name SET TABLESPACE new_tablespace;
ALTER INDEX index_name SET TABLESPACE new_tablespace;
ALTER DATABASE database_name SET TABLESPACE new_tablespace;
```

参照

SQLコマンドについては、“PostgreSQL Documentation”の“Reference”の“SQL Commands”を参照してください。

5.10 クラスタシステムにおける運用

高可用性システムやストリーミングレプリケーション、データベース多重化機能を使用したクラスタシステム上で、透過的データ暗号化を使用する場合について説明します。

5.10.1 データベース多重化運用を用いないHAクラスタ

データベース多重化運用を使用しないHAクラスタ環境で透過的データ暗号化を使用する場合、以下の点を留意してください。

キーストア・ファイルの配置と自動オープン

キーストア・ファイルの配置方法には以下の2通りあります。

- 1つのキーストア・ファイルを共有する方法

- キーストア・ファイルのコピーを配置する方法

1つのキーストア・ファイルを共有する方法

プライマリサーバとスタンバイサーバで、同じキーストア・ファイルを使用する方法です。

プライマリサーバが稼働している間はスタンバイサーバは動作しないため、同時にこのファイルをアクセスすることはなく、共有することができます。

キーストア・ファイルをさらに安全に管理するためには、安全な場所に隔離されたキー管理サーバまたはキー管理ストレージ上に配置してください。

キーストアの自動オープンの有効化は、プライマリサーバとスタンバイサーバの両方で行ってください。

キーストア・ファイルのコピーを配置する方法

プライマリサーバのキーストア・ファイルのコピーをスタンバイサーバ側に配置する方法です。

プライマリ、スタンバイの両方のサーバからアクセスできる共用のサーバやディスク装置が準備できない場合に、この方法を採用してください。

ただし、プライマリサーバでマスタ暗号化キーやパスフレーズを変更した場合、スタンバイサーバにキーストア・ファイルを再度コピーする必要があります。

キーストア・ファイルをさらに安全に管理するためには、プライマリサーバ、スタンバイサーバそれぞれに対して、安全な場所に隔離されたキー管理サーバまたはキー管理ストレージを準備し、キーストア・ファイルを配置してください。

キーストアの自動オープンの有効化は、プライマリサーバとスタンバイサーバの両方で行ってください。自動オープン・キーストアファイル（keystore.aks）をスタンバイサーバにコピーしても、キーストアの自動オープンは有効になりません。

5.10.2 データベース多重化運用

ストリーミングレプリケーションや、ストリーミングレプリケーションを利用したデータベース多重化機能を使用する環境で透過的データ暗号化を使用する場合、以下の点を留意してください。

キーストア・ファイルの配置

プライマリサーバのキーストア・ファイルのコピーをスタンバイサーバ側に配置してください。

これは、両方のサーバが同時にキーストア・ファイルをアクセスすることがあり、共有できないためです。

ポイント

キーストア・ファイルをさらに安全に管理するためには、安全な場所に隔離されたキー管理サーバまたはキー管理ストレージ上に配置してください。プライマリ、スタンバイの両方のサーバが使うキーストアは、同一のキー管理サーバまたはキー管理ストレージで管理することができます。

ただし、プライマリサーバとスタンバイサーバが使うキーストアは異なるディレクトリを作成して、プライマリサーバのキーストアをスタンバイサーバが使うディレクトリ上にコピーして使用してください。

キーストアの自動オープン

キーストアの自動オープンを必ず有効化してください。

このとき、キーストアの自動オープンの有効化は、データベース多重化運用を構成するすべてのサーバで行ってください。キーストアの自動オープンの設定はサーバ固有の情報を含むので、ファイルをコピーしただけでは、有効化されません。

パスフレーズの変更

パスフレーズの変更は、データベース多重化運用を構成するすべてのサーバに反映されるので、特別な作業は必要ありません。

スタンバイサーバの構築と起動

pg_basebackupコマンドまたはpgx_rcvallコマンドでスタンバイサーバを構築する前に、プライマリサーバからスタンバイサーバにキーストア・ファイルをコピーしておいてください。自動オープン・キーストアを使用する場合は、コピーしたキーストア・ファイルを用いてスタンバイサーバで自動オープンを有効にします。

スタンバイサーバを起動するときにキーストアをオープンしてください。これは、プライマリサーバから受信した暗号化されたWALを復号し、再生するために必要です。キーストアをオープンするには、pg_ctlコマンドまたはpgx_rcvallコマンドに、--keystore-passphraseを指定してパスフレーズを入力するか、または自動オープン・キーストアを使用します。

マスタ暗号化キーとパスフレーズの変更

マスタ暗号化キーとパスフレーズはプライマリサーバで変更します。そのとき、プライマリサーバからスタンバイサーバにキーストアをコピーする必要はありません。スタンバイサーバを再起動したり、キーストアを再度オープンする必要もありません。マスタ暗号化キーとパスフレーズの変更は、スタンバイサーバのキーストアにも反映されます。

参照

pgx_rcvallコマンドの詳細は、“リファレンス”の“pgx_rcvall”を参照してください。

pg_ctlコマンドの詳細は、“PostgreSQL Documentation”の“Reference”の“pg_ctl”を参照してください。

pg_basebackupコマンドの詳細は、“PostgreSQL Documentation”の“Reference”の“pg_basebackup”を参照してください。

ストリーミングレプリケーションを構築する手順については、“PostgreSQL Documentation”の“Server Administration”の“High Availability, Load Balancing, and Replication”を参照してください。

5.11 セキュリティに関する注意事項

- データベースサーバのメモリ(共有バッファ)内では、復号されたデータがキャッシュされます。その結果、プロセスのメモリダンプであるcoreファイルには、暗号化されていないデータが格納されます。したがって、安全な方法でメモリダンプを削除してください。安全にファイルを削除するために、以下のコマンドが利用できます。

  ー shredコマンド

- OSのスワップ領域には、データベースサーバのメモリから暗号化されていないデータが書き出されることがあります。スワップ領域からの情報漏えいを防ぐには、スワップ領域の使用を無効にするか、またはフルディスク暗号化製品を使ってスワップ領域を暗号化することを検討してください。

- サーバログファイルの内容は暗号化されません。そのため、SQL文に定数を指定していると、その値がサーバログファイルに出力されることがあります。これを防ぐには、log_min_error_statementなどのパラメータの設定を検討してください。

- キーストアをオープンしたりマスタ暗号化キーを変更するSQL関数を実行するときには、パスフレーズを含むそのSQL文がサーバログファイルに出力されないように注意する必要があります。そのためには、log_min_error_statementなどのパラメータの設定を検討してください。もしデータベースサーバとは別のコンピュータからこれらのSQL関数を実行する場合は、クライアントとデータベースサーバとの間の通信をSSLで暗号化してください。

- Fujitsu Enterprise Postgresでは、論理レプリケーションを利用することができます。これにより、バックアップされていないクラスタを、透過的データ暗号化が有効になっているデータベースにサブスクライブすることができます。論理レプリケーションでは、パブリッシャー

とサブスクライバー間で同じ暗号化方式を使用する必要はありません。サブスクライブされたデータのコピーを暗号化したい場合、ユーザーがサブスクライブしたデータベースに対して暗号化ポリシーを作成する必要があります。デフォルトでは、公開されている暗号化テーブル空間データはサブスクライバー側では暗号化されません。

5.12 構築済みアプリケーションの導入のヒント

透過的データ暗号化では、アプリケーションを変更することなく、アプリケーション全体のデータを容易に暗号化できます。

データベース管理者は、次のように構築済みアプリケーションを導入します。ただし、本手順を実行するとデフォルトテーブル空間にデータを格納することになりますので、元々の設計と異なる場合は必要に応じて対処してください。

1. (通常の手順) 構築済みアプリケーションのための所有者ユーザーとデータベースを作成します。

```
CREATE USER crm_admin ...;
CREATE DATABASE crm_db ...;
```

2. (暗号化のための手順) 構築済みアプリケーションのデータを格納する暗号化テーブル空間を作成します。

```
SET tablespace_encryption_algorithm = 'AES256';
CREATE TABLESPACE crm_tablespace LOCATION '/crm/data';
```

3. (暗号化のための手順) 構築済みアプリケーションの所有者ユーザーのデフォルト・テーブル空間として、暗号化テーブル空間を設定します。

```
ALTER USER crm_admin SET default_tablespace = 'crm_tablespace';
ALTER USER crm_admin SET temp_tablespaces = 'crm_tablespace';
```

4. (通常の手順) 構築済みアプリケーションのインストールを実行します。アプリケーションのインストーラは、データベースサーバのホスト名とポート番号、ユーザー名とデータベース名の入力を求めます。インストーラは入力された情報を使ってデータベースサーバに接続し、SQLスクリプトを実行します。インストーラがないアプリケーションでは、データベース管理者がSQLスクリプトを手動で実行します。

通常、アプリケーションのSQLスクリプトは、ER図から変換されたCREATE TABLEやCREATE INDEX、GRANT/REVOKEなどの論理定義のSQL文を含んでいます。データベースやユーザー、テーブル空間を作成するSQL文は含みません。そのSQLスクリプトを実行するユーザーのデフォルトのテーブル空間を設定しておくことで、SQLスクリプトが生成するオブジェクトはそのテーブル空間に配置されます。

第6章 鍵管理システムをキーストアとして使用する場合の透過的データ暗号化の運用

鍵管理システムをキーストアとして使用する場合の透過的データ暗号化の運用について説明します。

参照

Fujitsu Enterprise Postgresで利用可能な鍵管理システムの要件は、“導入ガイド(サーバ編)”の“鍵管理システムの要件”を参照してください。

6.1 暗号化によるデータの保護

“5.1 暗号化によるデータの保護”を参照してください。以下では、“5.1 暗号化によるデータの保護”で説明しているファイルベースのキーストアにおける透過的データ暗号化の運用との相違点を説明します。

暗号化の仕組み

2層の暗号化キーとキーストア

各テーブル空間には、その中のすべてのデータを暗号化/復号するテーブル空間暗号化キーがあります。テーブル空間暗号化キーは、マスタ暗号化キーで暗号化されて保存されます。
鍵管理システムに保存されている暗号化キーを、データベースクラスタで共通のマスタ暗号化キーとして使用します。Fujitsu Enterprise Postgresは鍵管理システムをマスタ暗号化キーのキーストアとして参照します。

鍵管理システムのタイプ

以下の2つのタイプの鍵管理システムを使用できます。

- kmip

  OASIS(Organization for the Advancement of Structured Information Standards)によって標準化されているKMIP(Key Management Interoperability Protocol)というプロトコルを使用して利用できる鍵管理システムです。

- custom

  KMIPプロトコルを採用せずに、要求の形式を変換するアダプタを用いて連携する鍵管理システムです。

参照

Fujitsu Enterprise Postgresで利用可能な鍵管理システムの要件は、“導入ガイド(サーバ編)”の“鍵管理システムの要件”を参照してください。

テーブル空間暗号化キーの共有

アダプタを利用して鍵管理システムと連携する場合、マスタ暗号化キーを使ったテーブル空間暗号化キーの暗号化や復号は鍵管理システム側で行われます。
テーブル空間暗号化キーを使用しようとするたびに鍵管理システムへアクセスする必要がないように、テーブル空間暗号化キーをデータベースクラスタ内で共有できます。

マスタ暗号化キーを利用した暗号化/復号処理のコストが問題になるのは以下の場合です。

- データベースへの複数の接続が暗号化テーブル空間にアクセスする
- 暗号化テーブル空間にアクセスする接続が繰り返され、かつコネクションプーリングが無効である

暗号化キーの識別子

鍵管理システム上に保管されている暗号化キーを特定するための識別子として、鍵IDを利用します。

鍵ID

鍵管理システム上の暗号化キーを特定するための情報で、鍵管理システム内で一意の情報です。暗号化キーの実体(バイト列)と鍵IDの対応は暗号化キーのライフサイクルを通じて変わりません。

識別子の呼び方は各鍵管理システムによって異なりますが、本機能ではこのような情報を鍵IDと呼びます。

鍵管理システムの変更

透過的データ暗号化機能の運用開始後に、使用する鍵管理システムを別の鍵管理システムに変更できます。

6.2 マスタ暗号化キーの設定

透過的データ暗号化を使用するには、キーストアを作成し、マスタ暗号化キーを設定する必要があります。

1. postgresql.confのshared_preload_librariesパラメータに、ライブラリ名“tde_kms”を指定してロードします。

```
shared_preload_libraries = 'tde_kms'
```

2. アダプタを利用する場合、アダプタをプラグインとして登録します。プラグインの格納場所のディレクトリをpostgresql.confのtde_kms.plugin_pathパラメータに指定します。使用するプラグインはこのディレクトリに格納します。

```
tde_kms.plugin_path = '/home/fsepuser/plugin/'
```

3. テーブル空間暗号化キーを共有する場合は、postgresql.confのtde_kms.enable_shared_dekパラメータに“on”を設定します。

```
tde_kms.enable_shared_dek = on
```

4. postgresql.confのtde_kms.kms_conninfo_fileパラメータに、鍵管理システムの接続情報を記述したファイルを設定します。パラメータの詳細については、“付録A パラメータ”を参照してください。

   鍵管理システム接続情報ファイルkms_conninfo.confの場合の例

```
tde_kms.kms_conninfo_file = 'kms_conninfo.conf'
```

   鍵管理システム接続情報ファイルの記述例

   タイプkmipの場合

```
kmip   mykmipsvr  mykmipsvr.example.com   5696   cert   sslcert=postgres.crt   sslkey=postgres.key
sslrootcert=root.crt
```

   タイプcustomの場合

```
custom   mykms   mykms   arg=--profile arg=user1
```

5. CREATE EXTENSION文を実行し、拡張モジュールをインストールします。

```
CREATE EXTENSION tde_kms;
```

6. 透過的データ暗号化を有効化するために、pgx_declare_external_master_key関数を呼び出して、マスタ暗号化キーとして使用する暗号化キーを宣言します。暗号化キーを特定するための識別子として、鍵IDを指定します。pgx_declare_external_master_key関数については、“B.2.3 pgx_declare_external_master_key”を参照してください。

```
SELECT pgx_declare_external_master_key( kms_name => 'mykmipsvr', key_id => 'a0eebc99-9c0b-0000-0000-000000000000',
sslpassphrase => 'mykmippassphrase' );
```

6.3 キーストアのオープン

暗号化テーブル空間を作成したり暗号化データにアクセスするには、キーストアをオープンしておく必要があります。キーストアをオープンすると、マスタ暗号化キーを暗号化と復号に利用できるようになります。

インスタンスを起動するたびにキーストアをオープンしてください。キーストアをオープンするには、データベースのスーパーユーザーが次のようにSQL関数を実行します。

```
SELECT pgx_open_keystore( sslpassphrase => 'passphrase');
```

passphraseはクライアント証明書用秘密鍵ファイルのパスフレーズです。

pgx_open_keystore関数の詳細は、“B.2 透過的データ暗号化制御関数”を参照してください。

ただし、以下の場合には、リカバリのために暗号化されたWALを復号する必要があるため、インスタンスを起動する時にパスフレーズの入力が必要になります。この場合は、上述のpgx_open_keystore関数を実行することができません。

- インスタンス起動時にクラッシュリカバリが行われる場合
- 継続的アーカイブによるリカバリを行う場合

上記の場合には、pg_ctlコマンドに--kms-secretオプションを指定して起動してください。パスフレーズ入力を促すプロンプトが以下のように表示されます。

```
> pg_ctl --kms-secret start
Enter secret:
```

ポイント

キーストアの自動オープンを有効にすると、パスフレーズを入力することなく、インスタンスの起動時に自動的にキーストアをオープンできます。詳細は“6.6.2 キーストアの自動オープンの有効化”を参照してください。

6.4 テーブル空間の暗号化

“5.4 テーブル空間の暗号化”を参照してください。

6.5 暗号化されているテーブル空間の確認

“5.5 暗号化されているテーブル空間の確認”を参照してください。

6.6 キーストアの管理

鍵管理システムを利用している場合のマスタ暗号化キーの管理方法について説明します。

6.6.1 マスタ暗号化キーの変更

マスタ暗号化キーを変更するには、最初に設定したときと同じくpgx_declare_external_master_key関数を実行します。詳細は“6.2 マスタ暗号化キーの設定”を参照してください。また、導入時に指定した鍵管理システム名と異なる鍵管理システム名を指定することで、利用する鍵管理システムを変更できます。詳細は、“6.6.5 鍵管理システムの変更”を参照してください。

6.6.2 キーストアの自動オープンの有効化

秘密にすべき情報を含めすべての資格情報を鍵管理システム接続情報ファイルに指定することで、パスフレーズを入力することなく、インスタンスの起動時に自動的にキーストアをオープンできます。キーストアの自動オープンを有効にするには、pgx_keystoreコマンドを実行します。

ファイルsslkeypassphrase.kscに難読化した資格情報を格納する場合の例

```
> pgx_keystore -s -o sslkeypassphrase.ksc
Enter secret:
```

鍵管理システム接続情報ファイルに難読化した資格情報を指定します。

```
kmip    mykmipsvr    kmip.example.com    5696    cert    sslcert=postgres.crt    sslkey=postgres.key sslrootcert=root.crt
sslkeypassphrase-obf=sslkeypassphrase.ksc
```

鍵管理システム接続情報ファイルは、それが作成されたコンピュータでのみ有効です。

キーストアの自動オープンを無効化するには、sslkeypassphrase-obfに指定した秘密鍵の難読化した資格情報を格納したファイルを削除し、鍵管理システム接続情報ファイルのsslkeypassphrase-obfオプションの記述を削除してください。

参照

pgx_keystoreコマンドの詳細については、“リファレンス”の“pgx_keystore”を参照してください。

鍵管理システム接続情報ファイルの詳細は“付録A パラメータ”を参照してください。

6.6.3 鍵管理システムに対する資格情報の変更

鍵管理サービスに対する資格情報が変更された場合、Fujitsu Enterprise Postgresによる鍵管理システムへの接続の際に使用する資格情報も変更する必要があります。

pgx_open_keystore関数を使用して、Fujitsu Enterprise Postgresで使用する資格情報の変更ができます。変更後の鍵管理システムへの接続には、新しい資格情報が使用されます。

ストリーミングレプリケーション構成を組んでいる場合、すべてのレプリカで資格情報を変更する必要があります。

pgx_open_keystore関数の詳細は、“B.2.1 pgx_open_keystore”を参照してください。

6.6.4 マスタ暗号化キーの確認

ビューpgx_tde_master_keyは、使用しているマスタ暗号化キーに関する情報を示します。列の詳細については“D.1 pgx_tde_master_key”を参照してください。

6.6.5 鍵管理システムの変更

鍵管理システムを使用する透過的データ暗号化機能の導入後に、連携する鍵管理システムを変更する必要が生じた場合、以下の手順で変更できます。

この手順の後、Fujitsu Enterprise Postgresのデータは新しい鍵管理システム上の暗号化キーを使用して暗号化されます。

新しい鍵管理システムの定義

新しい鍵管理システムが、鍵管理システムの接続情報ファイルに書かれていない場合は、その設定ファイルに定義を追加します。古い鍵管理システムと、新しい鍵管理システムには別の鍵管理システム名を付与します。

変更を有効にするために、設定ファイルのリロードを行ってください。

使用するマスタ暗号化キーの宣言

pgx_dexlare_external_master_key関数を使用して、新しく使用する暗号化キーを宣言します。鍵管理システム名として新しい鍵管理システムに付けた名前を指定します。その他の引数として、新しい鍵管理システム上の使用する鍵IDおよび資格情報が必要になります。正常に終了すると、新しい鍵管理システム上の暗号化キーを使用して暗号化されます。

注意

鍵管理システムを変更する前のバックアップデータを暗号化しているマスタ暗号化キーは、古い鍵管理システム上にしか存在しません。そのため、変更以前のバックアップデータをリストアする可能性がある期間は、古い鍵管理システムとそこに存在する暗号化キーが必要です。古い鍵管理システム上の暗号化キーの削除、古い鍵管理システムの破棄、鍵管理のサービスの解約などを行うとバックアップデータが復号できないため、そのデータを利用できなくなります。

6.7 データベースのバックアップとリストア/リカバリ

Fujitsu Enterprise Postgresでは、以下に示す5つのバックアップ/リカバリの方法が利用できます。

WebAdminによるバックアップ・リカバリ

- バックアップ

  WebAdminは、暗号化されたデータをバックアップします。

- リカバリ

  古いマスタ暗号化キーを使用していた時点にリカバリする場合、リカバリ後ただちに最新のマスタ暗号化キーに変更してください。

  “6.6.2 キーストアの自動オープンの有効化”の手順に従い、キーストアの自動オープンを有効にしてください。その後、WebAdminでリカバリを実行してください。

pgx_dmpall、pgx_rcvallコマンドによるバックアップ・リカバリ

- バックアップ

  pgx_dmpallコマンドは、暗号化されたデータをバックアップします。

- リカバリ

  古いマスタ暗号化キーを使用していた時点にリカバリする場合、リカバリ後ただちに最新のマスタ暗号化キーに変更してください。

  キーストアの自動オープンは、必要に応じて設定してください。

  キーストアの自動オープンを有効にしない場合、--kms-secretオプションを指定して、pgx_rcvallコマンドを実行してください。パスフレーズの入力を促すプロンプトが表示されます。

SQLによるダンプとリストア

- バックアップ

  pg_dumpコマンド、およびpg_dumpallコマンドが出力するデータは暗号化されません。そのため、“5.8 データベースのインポートとエクスポート”を参照して、OpenSSLなどのコマンドを利用してデータを暗号化してください。

- リストア

  OpenSSLなどのコマンドを利用して、バックアップデータを暗号化した場合、そのデータを復号してください。

  pg_dumpallコマンドが出力するデータは、デフォルトではテーブル空間の暗号化の指定を含みます。そのため、psqlコマンドは、テーブル空間を暗号化してリストアします。

ファイルシステムレベルのバックアップとリストア

- バックアップ

  インスタンスを停止して、OSのファイルコピーコマンドでデータ格納先のディレクトリやテーブル空間ディレクトリをバックアップします。暗号化テーブル空間のファイルは、暗号化された状態でバックアップされます。

- リストア

  インスタンスを停止して、OSのファイルコピーコマンドでデータ格納先のディレクトリやテーブル空間ディレクトリをリストアします。

  古いマスタ暗号化キーを使用していた時点にリカバリする場合、リカバリ後ただちに最新のマスタ暗号化キーに変更してください。

継続的アーカイブによるバックアップとポイントインタイムリカバリ

- バックアップ

  pg_basebackupコマンドは、暗号化されたデータをバックアップします。

- リカバリ

  古いマスタ暗号化キーを使用していた時点にリカバリする場合、リカバリ後ただちに最新のマスタ暗号化キーに変更してください。

  キーストアの自動オープンは、必要に応じて設定してください。

  キーストアの自動オープンを有効にしない場合、--kms-secretオプションを指定して、pg_ctlコマンドでインスタンスを起動してください。パスフレーズの入力を促すプロンプトが表示されます。

参照

pg_ctlコマンドの詳細は、“PostgreSQL Documentation”の“Reference”の“pg_ctl”を参照してください。

以下のコマンドの詳細は、“PostgreSQL Documentation”の“Reference”のそれぞれの項を参照してください。

- psql
- pg_dump
- pg_basebackup

以下のコマンドの詳細は、“リファレンス”を参照してください。

- pgx_rcvall
- pgx_dmpall
- pg_dumpall

6.8 データベースのインポートとエクスポート

“5.8 データベースのインポートとエクスポート”を参照してください。

6.9 既存データの暗号化

“5.9 既存データの暗号化”を参照してください。

6.10 クラスタシステムにおける運用

高可用性システムやストリーミングレプリケーション、データベース多重化機能を使用したクラスタシステム上で、透過的データ暗号化を使用する場合について説明します。

6.10.1 データベース多重化運用を用いないHAクラスタ

データベース多重化運用を使用しないHAクラスタ環境で、鍵管理システムをキーストアとして透過的データ暗号化を使用する場合、以下の点を留意してください。

鍵管理システムの接続情報ファイルの配置と自動オープン

postgresql.confファイルのtde_kms.kms_conninfo_fileパラメータに指定する鍵管理システムの接続情報を記述するファイル、およびそのファイルから参照される証明書などのファイルはプライマリサーバとスタンバイサーバで共有することができます。ただし、キーストアの自動オープンの有効化のために用いる難読化された資格情報ファイルは、自動オープンの有効化の手順に従いそれぞれのサーバで作成し、それぞれのサーバに配置する必要があります。

共有しない場合、スタンバイサーバから利用する鍵管理システムにプライマリサーバで設定した鍵管理システム名と同じ鍵管理システム名で接続できる必要があります。接続情報ファイルおよび接続情報ファイルから参照される証明書などのファイルをスタンバイサーバに配置してください。キーストアの自動オープンの有効化のために用いる難読化された資格情報ファイルは、自動オープンの有効化の手順に従いそれぞれのサーバで作成し、それぞれのサーバに配置する必要があります。

鍵管理システムに対する資格情報の変更

鍵管理システムに対する資格情報が変更された場合、プライマリサーバではpgx_open_keystore関数を使用して資格情報を変更してください。スタンバイサーバではキーストアの自動オープンの有効化を再度実施してください。

6.10.2 データベース多重化運用

ストリーミングレプリケーションや、ストリーミングレプリケーションを利用したデータベース多重化機能を使用する環境で、鍵管理システムをキーストアとして透過的データ暗号化を使用する場合、以下の点を留意してください。

鍵管理システムの接続情報ファイルの配置と自動オープン

postgresql.confファイルのtde_kms.kms_conninfo_fileパラメータに指定する鍵管理システムの接続情報を記述するファイル、およびそのファイルから参照される証明書などのファイルはプライマリサーバとスタンバイサーバで共有することができます。ただし、キーストアの自動オープンの有効化のために用いる難読化された資格情報ファイルは、自動オープンの有効化の手順に従いそれぞれのサーバで作成し、それぞれのサーバに配置する必要があります。

共有しない場合、利用する鍵管理システムにプライマリサーバで設定した鍵管理システム名と同じ鍵管理システム名で接続できる必要があります。接続情報ファイルおよび接続情報ファイルから参照される証明書などのファイルをデータベース多重化運用を構成するすべてのサーバに配置してください。キーストアの自動オープンの有効化のために用いる難読化された資格情報ファイルは、自動オープンの有効化の手順に従いそれぞれのサーバで作成し、それぞれのサーバに配置する必要があります。

鍵管理システムに対する資格情報の変更

鍵管理システムに対する資格情報が変更された場合、データベース多重化運用を構成するすべてのサーバでpgx_open_keystore関数を使用して資格情報を変更してください。

スタンバイサーバの起動

スタンバイサーバを起動するときにキーストアをオープンしてください。これは、プライマリサーバから受信した暗号化されたWALを復号し、再生するために必要です。キーストアをオープンするには、pg_ctlコマンドまたはpgx_rcvallコマンドに、--kms-secretを指定して資格情報を入力するか、またはキーストアの自動オープンを有効化します。

マスタ暗号化キーの変更

マスタ暗号化キーはプライマリサーバで変更します。スタンバイサーバを再起動したり、キーストアを再度オープンする必要はありません。マスタ暗号化キーの変更は、スタンバイサーバにも反映されます。

参照

pgx_rcvallコマンドの詳細は、“リファレンス”の“pgx_rcvall”を参照してください。

pg_ctlコマンドの詳細は、“PostgreSQL Documentation”の“Reference”の“pg_ctl”を参照してください。

ストリーミングレプリケーションを構築する手順については、“PostgreSQL Documentation”の“Server Administration”の“High Availability,Load Balancing, and Replication”を参照してください。

6.11 セキュリティに関する注意事項

“5.11 セキュリティに関する注意事項”を参照してください。

6.12 構築済みアプリケーションの導入のヒント

“5.12 構築済みアプリケーションの導入のヒント”を参照してください。

第7章 データ秘匿化

データ秘匿化とは、アプリケーションによって発行された問合せに対して、一部のデータを改訂して利用者に参照させる機能です。
たとえば、従業員データの問合せに対して、8桁の従業員番号の最後の4桁以外を“*”で改訂して参照させる場合などに利用できます。

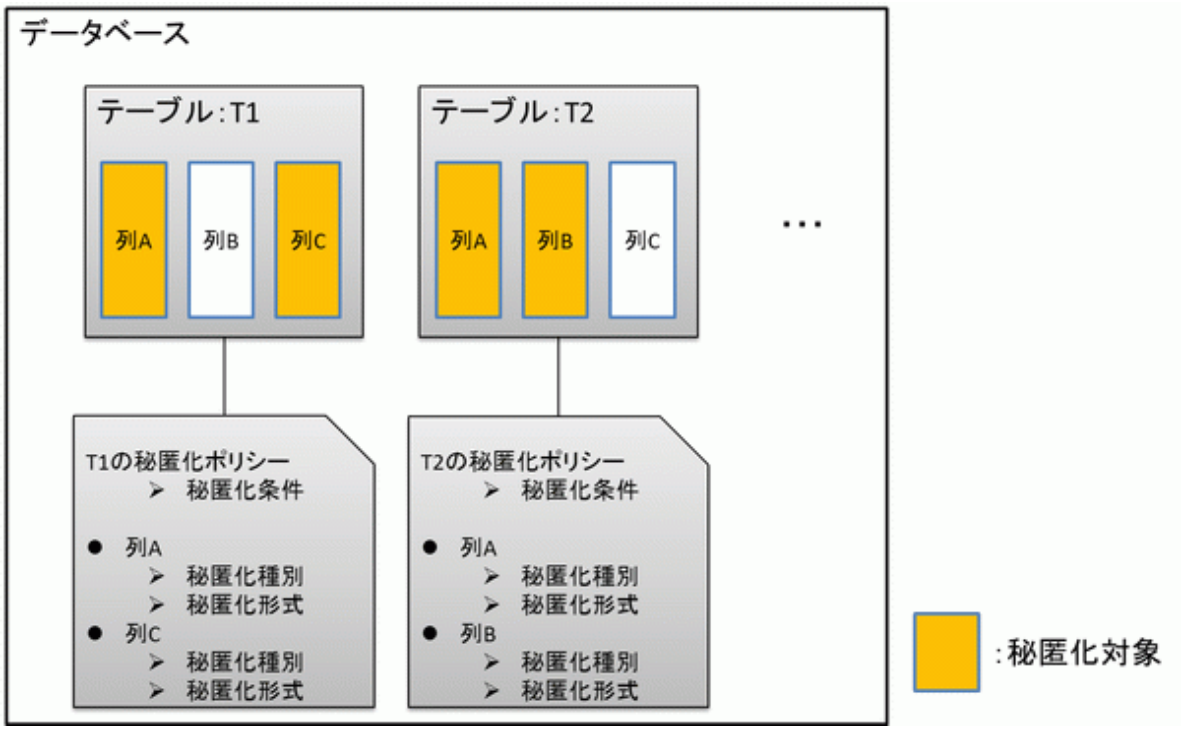
注意

本機能を利用する場合は、改訂したデータを別の媒体に移行して、利用者に参照させる利用方法を推奨します。これは、利用者がデータベースに直接アクセスして秘匿化データを抽出すると、設定した秘匿化ポリシーや秘匿化の対象列への問合せ結果を分析することで、改訂前のデータを推定できる可能性があるためです。

7.1 秘匿化ポリシー

秘匿化ポリシーとは、アプリケーションからのアクセス時に特定の条件下でデータを改訂する方法を規定したものです。一つのテーブルに対し一つの秘匿化ポリシーが作成できます。秘匿化ポリシーには、秘匿化対象、秘匿化種別、秘匿化条件、秘匿化形式などが設定できます。

図7.1 秘匿化ポリシー



注意

秘匿化ポリシーを定義することにより、対応するテーブルの検索性能が劣化する場合があります。

7.1.1 秘匿化対象

秘匿化対象とは、秘匿化ポリシーを適用する列です。秘匿化対象または秘匿化対象を含む関数を参照した場合、該当する実行結果は改訂されて取得されます。

実行結果を改訂する対象のコマンドは以下です。

- SELECT
- COPY

- pg_dump
- pg_dumpall

注意

- 秘匿化対象をINSERT～SELECT対象列に指定した場合は、改訂前のデータで処理を実施します。
- 秘匿化対象をSELECT対象列以外に指定した場合は、改訂前のデータで処理を実施します。
- 秘匿化対象をデータ型が変換される関数に指定した場合はエラーになります。

7.1.2 秘匿化種別

秘匿化種別とは、列データを改訂する方法です。秘匿化種別はfunction_typeパラメータに指定します。秘匿化種別には、以下の3つが指定でき、秘匿化対象のデータの特性によって選択することが可能です。

全秘匿化

列データの内容をすべて改訂します。問合せを行ったアプリケーションに返される改訂された値は列のデータ型によって異なります。たとえば、数値型の列は0で改訂され、文字型の列は空白で改訂されます。

部分秘匿化

列データの一部を改訂します。
たとえば、従業員番号の最後の4桁以外を“*”で改訂できます。

正規表現秘匿化

正規表現を用いた検索を利用して列データを改訂します。
たとえば、文字の長さが変化する可能性のある電子メールアドレスなどの文字列に対し、正規表現を用いることで“@”以前を“*”で改訂できます。正規表現秘匿化は文字型データのみで使用可能です。

注意

- 複数の有効な秘匿化対象を1つの関数に指定した場合は、最も左にある秘匿化対象の秘匿化種別が適用されます。
  たとえば、数値型の秘匿化対象c1、c2について“SELECT GREATEST(c1, c2) FROM t1”を実行した場合、c1の秘匿化種別が適用されます。
- マルチバイト文字を含むデータを秘匿化する場合、秘匿化種別に部分秘匿化を指定しないでください。期待した結果が得られない場合があります。

7.1.3 秘匿化条件

秘匿化条件とは、秘匿化が実行されるための条件です。秘匿化条件はexpressionパラメータに指定します。秘匿化条件を定義することで、異なるユーザーに改訂されたデータまたは実際のデータのいずれかを表示させることが可能です。秘匿化条件には、boolean型の結果を返す式を指定する必要があり、TRUEと評価された場合にのみ秘匿化が実行されます。指定できる式の詳細は、“PostgreSQL Documentation”の“Value Expressions”を参照してください。ただし、列を含む式は指定できません。
たとえば、“postgres”ユーザーのみにデータを秘匿化する場合、秘匿化条件に'current_user = "postgres"'を指定してください。

参考

秘匿化が常に実行されるためには、秘匿化条件が常にTRUEと評価されるよう'1=1'のように指定します。

7.1.4 秘匿化形式

秘匿化形式とは、秘匿化条件が成立した時の改訂する方法と表示される文字の組み合わせです。秘匿化形式は秘匿化種別ごとに異なっています。秘匿化形式については以下に示します。

全秘匿化

全秘匿化では、すべての文字をデータベースで決められた値に改訂します。改訂後の文字はpgx_confidential_valuesテーブルを参照することで確認できます。また、改訂後の文字はpgx_update_confidential_valuesシステム管理関数で変更可能です。

参照

秘匿化が可能なデータ型は、“7.3 データ秘匿化が可能なデータ型”を参照してください。

部分秘匿化

部分秘匿化では、データをfunction_parametersの内容に従って改訂します。function_parametersはデータ型によって指定方法が異なります。

カテゴリ	function_parametersの指定方法
数値型	'改訂文字, 開始位置, 終了位置' ・ 改訂文字:表示する数字を指定します。指定できる範囲は0-9です。 ・ 開始位置:秘匿化を開始する数字位置を指定します。指定できる範囲は正の整数です。 ・ 終了位置:秘匿化を終了する数字位置を指定します。指定できる範囲は開始位置よりも大きい正の整数です。 **例** 1桁目から5桁目までを9に秘匿化する場合は以下を指定します。 function_parameters := '9, 1, 5' この場合、元データが“123456789”の場合は、“999996789”に改訂されます。
文字型	'入力形式, 出力形式, 改訂文字, 開始位置, 終了位置' ・ 入力形式:データが現在どのようにフォーマットされているかを指定します。秘匿化される可能性がある文字にV、空白やハイフンなど秘匿化されない文字はFを指定します。 ・ 出力形式:表示されるデータを書式化する方法を定義します。秘匿化される可能性がある文字にVを指定します。入力形式の各文字Fについては出力させる任意の文字を指定してください。単一引用符を出力したい場合は2つ続けて記述してください。 ・ 改訂文字:任意の単一文字を指定します。単一引用符を出力したい場合は2つ続けて記述してください。 ・ 開始位置:秘匿化を開始するVの数字位置を指定します。指定できる範囲は正の整数です。 ・ 終了位置:秘匿化を終了するVの数字位置を指定します。終了位置の値を決める時はFの位置は含めないでください。指定できる範囲は開始位置よりも大きい正の整数です。 **例** 電話番号の始めの3桁を除いて*に秘匿化する場合は以下を指定します。 function_parameters := 'VVVFVVVVFVVVV, VVV-VVVV-VVVV, *, 4, 11' この場合、元データが“012-3456-7890”の場合は、“012-****-****”に改訂されます。

カテゴリ	function_parametersの指定方法
日付／タイムスタンプ型	'MDYHMS' • M:月を秘匿化します。月を秘匿化する場合は1から12を小文字のmの後ろに追記します。秘匿化しない場合は大文字のMを指定します。 • D:日を秘匿化します。日を秘匿化する場合は1から31を小文字のdに追記します。月の日よりも大きい数値を入力すると、その月の最終日が表示されます。秘匿化しない場合は大文字のDを指定します。 • Y:年を秘匿化します。年を秘匿化する場合は1から9999を小文字のyに追記します。秘匿化をしない場合は大文字のYを指定します。 • H:時間を秘匿化します。時間を秘匿化する場合は0から23を小文字のhに追記します。秘匿化をしない場合は大文字のHを指定します。 • M:分を秘匿化します。分を秘匿化する場合は0から59を小文字のmに追記します。秘匿化をしない場合は大文字のMを指定します。 • S:秒を秘匿化します。秒を秘匿化する場合は0から59を小文字のsに追記します。秘匿化をしない場合は大文字のSを指定します。 **例** 時間、分、秒を0時0分0秒に秘匿化する場合は以下を指定します。 function_parameters := 'MDYh0m0s0' この場合、元データが“2022-02-10 10:10:10”の場合は、“2022-02-10 00:00:00”に改訂されます。

参照

- function_parametersについては、“B.3.2 pgx_create_confidential_policy”を参照してください。
- 秘匿化が可能なデータ型は、“7.3 データ秘匿化が可能なデータ型”を参照してください。

正規表現秘匿化

正規表現秘匿化では、データをregexp_pattern、regexp_replacement、regexp_flagsの内容に従って改訂します。regexp_patternには、正規表現による改訂方法を指定します。regexp_replacementには、正規表現の改訂後の文字を指定します。regexp_flagsには、正規表現のフラグを指定します。

例

bから始まる3文字をすべてXに改訂する場合は以下を指定します。

regexp_pattern := 'b..'

regexp_replacement:= 'X'

regexp_flags := 'g'

この場合、元データが“foobarbaz”の場合は、“fooXX”に改訂されます。

参照

- regexp_pattern、regexp_replacement、regexp_flagsに指定できる値は、“PostgreSQL Documentation”の“POSIX Regular Expressions”のpattern、replacement、flagsを参照してください。
- 秘匿化が可能なデータ型は、“7.3 データ秘匿化が可能なデータ型”を参照してください。

注意

- 列のデータ型がcharacter(n)、char(n)の場合、改訂後の文字列長が列の長さ(n)を超えると、列の長さまで切り詰められて表示されます。
- 列のデータ型がcharacter varying(n)、varchar(n)の場合、改訂後の文字列長が改訂前の文字列長の長さを超えると、改訂前の文字列長まで切り詰められて表示されます。

7.2 利用方法

準備

本機能を使用するためには、以下の準備が必要となります。

1. postgresql.confファイルのパラメータを設定します。

   パラメータ“shared_preload_libraries”の先頭に『pgx_datamasking』を追加します。

2. インスタンスを再起動します。

3. 本機能を利用するデータベースに対して、CREATE EXTENSION を実施します。

   ここでは、対象のデータベースを“postgres”として説明します。

   psqlコマンドを利用して、データベース“postgres”に接続します。

   例

   ```
   postgres=# CREATE EXTENSION pgx_datamasking;
   CREATE EXTENSION
   ```

   注意

   『pgx_datamasking』は、必ずパラメータ“shared_preload_libraries”の先頭に追加してください。

   参考

   ― 本機能を利用しないときは、“pgx_datamasking.enable”を“false”に設定します。秘匿化ポリシーを設定していても、データは秘匿されません。“pgx_datamasking.enable”を“true”に設定することで、再度本機能を利用することができます。この設定は、SET文による指定とpostgresql.confファイルのパラメータ指定で可能です。
   例

   ```
   postgres=# SET pgx_datamasking.enable=false;
   ```

   ― 今後、新しいデータベースを作成する場合に、デフォルトで本機能が利用できるように、データベース“template1”に対しても同様に本準備作業を実施することを推奨します。

利用

秘匿化を利用するためには、秘匿化ポリシーの設定が必要です。秘匿化ポリシーは運用時に作成、変更、確認、有効化／無効化、削除が可能です。
秘匿化ポリシーの利用方法について、例を用いて以下の手順で説明します。

1. 作成方法
2. 変更方法
3. 確認方法

4. 有効化/無効化方法
5. 削除方法

注意

秘匿化ポリシーの設定はデータベースのスーパーユーザーのみ実行可能です。

7.2.1 秘匿化ポリシーの作成

以下にサーバ上からの実行例を示します。

1. 秘匿化ポリシーの作成
   秘匿化ポリシーを作成するために、pgx_create_confidential_policyシステム管理関数を実行します。
   設定する値は以下とします。
   ○ 秘匿化対象:数値型のc1
   ○ 秘匿化種別:FULL
   ○ 秘匿化条件:'1=1'

```
postgres=# select pgx_create_confidential_policy(table_name := 't1', policy_name := 'p1', expression := '1=1',
column_name := 'c1', function_type := 'FULL');
 pgx_create_confidential_policy
--------------------------------
 t
(1 row)
```

2. 表示されるデータの確認
   秘匿化対象のデータ(列c1)が正しく改訂されていることを確認します。

```
postgres=# select * from t1;
 c1 |      c2
----+--------------
  0 | 012-3456-7890
  0 | 012-3456-7891
  0 | 012-3456-7892
(3 row)
```

参照

- pgx_create_confidential_policyシステム管理関数の詳細は、“B.3.2 pgx_create_confidential_policy”を参照してください。

注意

- 秘匿化ポリシーはテーブルに対し1つのみ作成可能です。
- 作成した秘匿化ポリシーは、すべてのユーザーが閲覧可能であるため、改訂されたデータを参照する利用者には本機能を設定したデータベースへのログイン権限を付与しないでください。秘匿化ポリシーは“pgx_confidential_columns”テーブル、“pgx_confidential_policies”テーブル、“pgx_confidential_values”テーブルに定義されます。

7.2.2 秘匿化ポリシーの変更

1. 以下にサーバ上からの実行例を示します。
2. 秘匿化ポリシーの変更
   秘匿化ポリシーを変更するために、pgx_alter_confidential_policyシステム管理関数を実行します。
   変更内容は以下とします。
   ○ 変更内容:秘匿化対象の追加

○ 秘匿化対象:文字型のc2
○ 秘匿化種別:PARTIAL
○ 秘匿化条件:'VVVFVVVVFVVVV, VVV-VVVV-VVVV, *, 4, 11'

```
postgres=# select pgx_alter_confidential_policy(table_name := 't1', policy_name := 'p1', action := 'ADD_COLUMN',
column_name := 'c2', function_type := 'PARTIAL', function_parameters := 'VVVFVVVVFVVVV, VVV-VVVV-VVVV, *, 4, 11');
 pgx_alter_confidential_policy
-------------------------------
 t
(1 row)
```

3. 表示されるデータの確認
秘匿化対象のデータが正しく改訂されていることを確認します。

```
postgres=# select * from t1;
 c1 |      c2
----+---------------
  0 | 012-****-****
  0 | 012-****-****
  0 | 012-****-****
(3 row)
```

参照

・ pgx_alter_confidential_policyシステム管理関数の詳細は、“B.3.1 pgx_alter_confidential_policy”を参照してください。

7.2.3 秘匿化ポリシーの確認

以下にサーバ上からの実行例を示します。

1. 秘匿化ポリシーが設定されている秘匿化対象に関する情報の確認
秘匿化ポリシーが設定されている秘匿化対象を確認するために、pgx_confidential_columnsテーブルを参照します。

```
postgres=# select * from pgx_confidential_columns;
 schema_name | table_name | policy_name | column_name | function_type |          function_parameters          |
regexp_pattern | regexp_replacement | regexp_flags | column_description
-------------+------------+-------------+-------------+---------------+--------------------------------------
+----------------+--------------------+--------------+--------------------
 public      | t1         | p1          | c1          | FULL          |                                      |
|                    |              |
 public      | t1         | p1          | c2          | PARTIAL       | VVVFVVVVFVVVV, VVV-VVVV-VVVV, *, 4, 11 |
|                    |              |
(2 row)
```

2. 秘匿化ポリシーの内容に関する情報の確認
秘匿化ポリシーの内容を確認するために、pgx_confidential_policiesを参照します。

```
postgres=# select * from pgx_confidential_policies;
 schema_name | table_name | policy_name | expression | enable | policy_description
-------------+------------+-------------+------------+--------+--------------------
 public      | t1         | p1          | 1=1        | t      |
(1 row)
```

参照

・ pgx_confidential_columnsテーブルの詳細は、“E.1 pgx_confidential_columns”を参照してください。

・ pgx_confidential_policiesテーブルの詳細は、“E.2 pgx_confidential_policies”を参照してください。

7.2.4 秘匿化ポリシーの有効化／無効化

以下にサーバ上からの実行例を示します。

1. 秘匿化ポリシーの無効化
   秘匿化ポリシーを無効にするためには、pgx_enable_confidential_policyシステム管理関数を実行します。

```
postgres=# select pgx_enable_confidential_policy(table_name := 't1', policy_name := 'p1', enable := 'f');
 pgx_enable_confidential_policy
--------------------------------
 t
(1 row)
```

2. 表示されるデータの確認
   秘匿化ポリシーの無効化により元データが表示されていることを確認します。

```
postgres=# select * from t1;
 c1 |      c2
----+---------------
  1 | 012-3456-7890
  2 | 012-3456-7891
  3 | 012-3456-7892
(3 row)
```

3. 秘匿化ポリシーの有効化
   秘匿化ポリシーを有効にするためには、pgx_enable_confidential_policyシステム管理関数を実行します。

```
postgres=# select pgx_enable_confidential_policy(table_name := 't1', policy_name := 'p1', enable := 't');
 pgx_enable_confidential_policy
--------------------------------
 t
(1 row)
```

4. 表示されるデータの確認
   秘匿化ポリシーの有効化により秘匿化対象のデータが正しく改訂されていることを確認します。

```
postgres=# select * from t1;
 c1 |      c2
----+---------------
  0 | 012-****-****
  0 | 012-****-****
  0 | 012-****-****
(3 row)
```

参照

- pgx_enable_confidential_policyシステム管理関数の詳細は、“B.3.4 pgx_enable_confidential_policy”を参照してください。

7.2.5 秘匿化ポリシーの削除

以下にサーバ上からの実行例を示します。

1. 秘匿化ポリシーの削除
   秘匿化ポリシーを削除するためには、pgx_drop_confidential_policyシステム管理関数を実行します。

```
postgres=# select pgx_drop_confidential_policy(table_name := 't1', policy_name := 'p1');
 pgx_drop_confidential_policy
------------------------------
 t
(1 row)
```

2. 表示されるデータの確認
   秘匿化ポリシーの削除により元データが表示されていることを確認します。

```
postgres=# select * from t1;
 c1 |      c2
----+--------------
  1 | 012-3456-7890
  2 | 012-3456-7891
  3 | 012-3456-7892
(3 row)
```

参照

- pgx_drop_confidential_policyシステム管理関数の詳細は、“B.3.3 pgx_drop_confidential_policy”を参照してください。

7.3 データ秘匿化が可能なデータ型

データ秘匿化機能を利用できるデータ型は以下のとおりです。

カテゴリ	データ型	利用可能な秘匿化種別		
		全秘匿化	部分秘匿化	正規表現秘匿化
数値型	smallint	○	○	×
	integer	○	○	×
	bigint	○	○	×
	decimal	○	○	×
	numeric	○	○	×
	float	○	○	×
	real	○	○	×
	double precision	○	○	×
文字型	character varying(n)	○	○	○
	varchar(n)	○	○	○
	character(n)	○	○	○
	char(n)	○	○	○
日付／タイムスタンプ型	date	○	○	×
	timestamp	○	○	×

注意

データ秘匿化が可能なデータ型であっても、データが特殊な値(NaN 、Infinity、-Infinity)の場合は秘匿化されません。

7.4 セキュリティに関する注意事項

- 論理レプリケーションを利用すると、バックアップされていないクラスタは、データ秘匿化ポリシーが有効になっているデータベースにサブスクライブすることができます。また、論理レプリケーションを使用すると、パブリッシャーとサブスクライバーのデータベースは、それぞれ独自のデータ秘匿化ポリシーまたは同じデータ秘匿化ポリシーを持つことができます。サブスクリプションが作成されるたびにパブリッシャーデータベースのデータ秘匿化を無効にする必要があります。これにより、サブスクライバーは秘匿化されたバージョンではなく元のデータ(初期コピー)を取得できるようになります。また、ユーザーはサブスクライブしている各データベースに秘匿化ポリシーを設定する必要があります。

- ユーザーがデータベースのすべてのテーブルをパブリッシュしていて、すべてのデータベースに対して同じデータ秘匿化ポリシーをサブスクライブしたデータベースに適用しない限り、データ秘匿化の機密テーブル(pgx_confidential_policies、pgx_confidential_columnsなど)をパブリッシュする際には十分注意してください。これらの機密テーブルにはデータベースのすべてのテーブルの秘匿化ポリシーが含まれているため、パブリッシュされていないテーブルの機密ポリシーが誤って公開される可能性があります。また、サブスクライバーデータベースに異なるデータ秘匿化ポリシーを適用することはできません。

第8章 定期的な運用操作

本章では、日常のデータベース業務を行うにあたって、定期的に行う必要のある運用操作について説明します。

8.1 ログの設定と監視

Fujitsu Enterprise Postgresでは、データベースのエラーや警告などをログファイルに出力できます。

エラー発生有無や、エラー原因を特定するために有益な情報となります。

デフォルトでは、システムログに出力されます。また、Fujitsu Enterprise Postgresを運用する前に、Fujitsu Enterprise Postgresのログファイル(log_destinationなど)のログを取得するように設定することを推奨します。

上記で取得したログファイルを定期的に監視し、エラーが発生していないか監視してください。

参照

- ログの詳細は、“PostgreSQL Documentation”の“Server Administration”の“Error Reporting and Logging”を参照してください。
- WebAdminで運用する場合のログの設定は、“導入ガイド(サーバ編)”の“設定パラメータ”を参照してください。

8.2 ディスクの使用量の監視と空き領域の確保

データベースを長期間利用すると、ディスク装置の空き領域を圧迫し、ディスク領域が不足してしまうことがあります。この状態になると、データベース業務が停止して業務が続行できなくなる可能性があります。

そのため、定期的にディスク容量の使用状況を監視し、同じディスク装置内にある不要なファイルを削除します。

ディスクの使用量は、下記のディレクトリを配置しているディスクに対して監視してください。

- データ格納先のディレクトリ
- トランザクションログ格納先のディレクトリ(トランザクションログをデータ格納先のディレクトリとは別に格納している場合)
- バックアップデータ格納先のディレクトリ
- テーブル空間格納先のディレクトリ

8.2.1 ディスクの使用量の監視

ディスクの使用量を確認するには、OSが提供する下記のコマンドを使用して確認してください。

- dfコマンド

また、SQL文でテーブルやインデックスの単位で確認することもできます。

この方法は、“PostgreSQL Documentation”の“Server Administration”の“Determining Disk Usage”を確認してください。

参考

WebAdminによる運用を行っている場合は、ディスクの使用量が全体の80%に達すると警告が表示されます。

8.2.2 ディスクの空き領域の確保

ディスク容量の空き領域は、OSが提供する以下のコマンドを使用して、同じディスク装置内にあるデータベース以外の不要なファイルを削除することにより、確保してください。

- rmコマンド

また、以下を定期的に実行することにより、ディスク容量を確保できます。

- データ格納ディスクの容量を確保する場合

  REINDEX文を実行します。詳細は"8.5 インデックスの再編成"を参照してください。

- バックアップデータ格納ディスクの容量を確保する場合

  WebAdmin、またはpgx_dmpallコマンドによりバックアップを実行します。

8.3 コネクションの自動切断

何らかの原因でアプリケーションがダウンして、アプリケーションが異常終了した場合、データベースサーバ上にアプリケーションとのコネクションが残る可能性があります。この状態が長時間続いた場合、新しいアプリケーションがデータベースサーバに接続してもエラーになったり、テーブルなどが使用中である旨のエラーになることがあります。

そのため、一定時間、アイドル中のコネクションを自動的に切断することを推奨します。

コネクションが自動切断されるまでの時間は、postgresql.confファイルに以下のパラメータを設定してください。

パラメータ名	設定値	説明
tcp_keepalives_idle	keepaliveを送信するまでの時間(秒数) 0の場合はシステムのデフォルト値を使用します。	指定された秒数間、アイドル中のコネクションに対して、keepaliveを送信します。 30秒を指定することを推奨します。
tcp_keepalives_interval	keepaliveの送信間隔時間(秒数) 0の場合はシステムのデフォルト値を使用します。	指定された間隔でkeepaliveを送信します。 10秒を指定することを推奨します。
tcp_user_timeout	サーバからの応答を待機する時間(ミリ秒) 0の場合はシステムのデフォルト値を使用します。 未設定の場合、0が指定された場合と同じ動作になります。	接続の確立後、クライアントからサーバへの送信にてTCP再送処理が動作した場合に、切断とみなすまでの時間を指定します。 本パラメータに0以外を指定した場合、自動切断までの時間は本パラメータに指定した待機時間で判断します。実際の待機時間は、本パラメータで指定した時間を経過した後の、最初のkeepalive再送のタイミングまでとなります。

注意

tcp_user_timeoutパラメータに0以外を指定した場合、tcp_keepalives_idleパラメータおよびtcp_keepalives_intervalパラメータによる待機時間は無効となり、tcp_user_timeoutパラメータの指定値による待機時間となります。

参照

パラメータの詳細については、“PostgreSQL Documentation”の“Server Administration”の“Connection Settings”を参照してください。

8.4 アプリケーションの接続状態の監視

Fujitsu Enterprise Postgresは、更新、削除されたデータを即座に削除しません。バキュームにより、参照するトランザクションが存在しないと判断した場合、これらの不要データを回収します。

しかし、長時間接続したコネクションや、資源を占有したコネクションが存在している場合、不要データが回収されず、データベースが膨張して、データベースの性能劣化となる可能性があります。

参照

バキュームの詳細は、“PostgreSQL Documentation”の“Server Administration”の“Routine Vacuuming”を参照してください。

このような場合、原因となるコネクションを監視することで、データベースの性能劣化を抑止できます。

長時間、待ち状態となっているコネクションを監視するには、以下の方法があります。

- 8.4.1 ビュー(pg_stat_activity)を使用する方法

8.4.1 ビュー(pg_stat_activity)を使用する方法

ビュー(pg_stat_activity)を利用して、長時間、待ち状態のクライアントのコネクションを監視します。

例

以下の例は、クライアントの待ち状態が、60分以上の場合、該当コネクションを表示します。

ただし、アプリケーションの互換性を維持することを考慮し、下記のSQL文中のシステムカタログを直接参照しないようにしてください。

```
postgres=# select * from pg_stat_activity where backend_type = 'client backend' and state='idle in transaction' and
current_timestamp > cast(query_start + interval '60 minutes' as timestamp);
-[ RECORD 1 ]----+------------------------------
datid            | 13003
datname          | db01
pid              | 4638
leader_pid       |
usesysid         | 10
usename          | fsepuser
application_name | apl01
client_addr      | 192.33.44.15
client_hostname  |
client_port      | 27500
backend_start    | 2022-02-24 09:09:21.730641+09
xact_start       | 2022-02-24 09:09:23.858727+09
query_start      | 2022-02-24 09:09:23.858727+09
state_change     | 2022-02-24 09:09:23.858834+09
wait_event_type  | Client
wait_event       | ClientRead
state            | idle in transaction
backend_xid      |
backend_xmin     |
```

```
query_id        |
query           | begin;
backend_type    | client backend
```

参照

- アプリケーションの互換性の維持に関する詳細は、“アプリケーション開発ガイド”の“アプリケーションの互換に関する注意事項”を参照してください。
- pg_stat_activityの詳細は、“PostgreSQL Documentation”の“Server Administration”の“The Statistics Collector”を参照してください。

8.5 インデックスの再編成

データベースは通常、テーブルにインデックスを定義しますが、データを頻繁に更新すると、インデックスがディスク装置の空き領域を効率的に利用できなくなってきます。また、これに伴い、データベースへのアクセス性能が少しずつ低下する可能性があります。

ディスク装置の使用領域を整理し、データベースへのアクセス性能の低下を防ぐために、REINDEXコマンドを定期的に実行してインデックスを再編成することを推奨します。

ディスクの使用量は、“8.2 ディスクの使用量の監視と空き領域の確保”に説明している方法で、データ格納先のディレクトリに対して確認してください。

注意

REINDEXコマンドは、処理中のインデックスに対する排他ロックを取得し、インデックスの元となるテーブルの書き込みをロックするため、これらにアクセスする他の処理はロック待ちで停止する場合があります。

そのため業務終了後に実施するなどの考慮が必要です。

参照

REINDEXコマンドの定期的な実行によるインデックスの再編成について、詳細な内容は“PostgreSQL Documentation”の“Server Administration”の“Routine Reindexing”を参照してください。

ポイント

通常は月に1回の頻度で、データベースの保守などを行う時期を見計らって実行してください。インデックスの使用量をSQL文で調べ、日々増加の傾向が見られる場合には、ディスクの空き容量と比較してインデックス再作成の実行頻度を調整します。

SQL文および実行結果の例を以下に示します。

ただし、アプリケーションの互換性を維持することを考慮し、下記のSQL文中のシステムカタログや関数を直接参照したり使用しないようにしてください。

[SQL文]

```
SELECT
  nspname AS schema_name,
  relname AS index_name,
  round(100 * pg_relation_size(indexrelid) / pg_relation_size(indrelid)) / 100 AS index_ratio,
  pg_size_pretty(pg_relation_size(indexrelid)) AS index_size,
  pg_size_pretty(pg_relation_size(indrelid)) AS table_size
FROM pg_index I
  LEFT JOIN pg_class C ON (C.oid = I.indexrelid)
  LEFT JOIN pg_namespace N ON (N.oid = C.relnamespace)
```

```
WHERE
  C.relkind = 'i' AND
  pg_relation_size(indrelid) > 0
ORDER BY pg_relation_size(indexrelid) DESC, index_ratio DESC;
```

[実行結果]

```
 schema_name |            index_name            | index_ratio | index_size | table_size
-------------+----------------------------------+-------------+------------+------------
 public      | pgbench_accounts_pkey            |        0.16 | 2208 kB    | 13 MB
 pg_catalog  | pg_depend_depender_index         |         0.6 | 224 kB     | 368 kB
 pg_catalog  | pg_depend_reference_index        |        0.58 | 216 kB     | 368 kB
...
```

参照

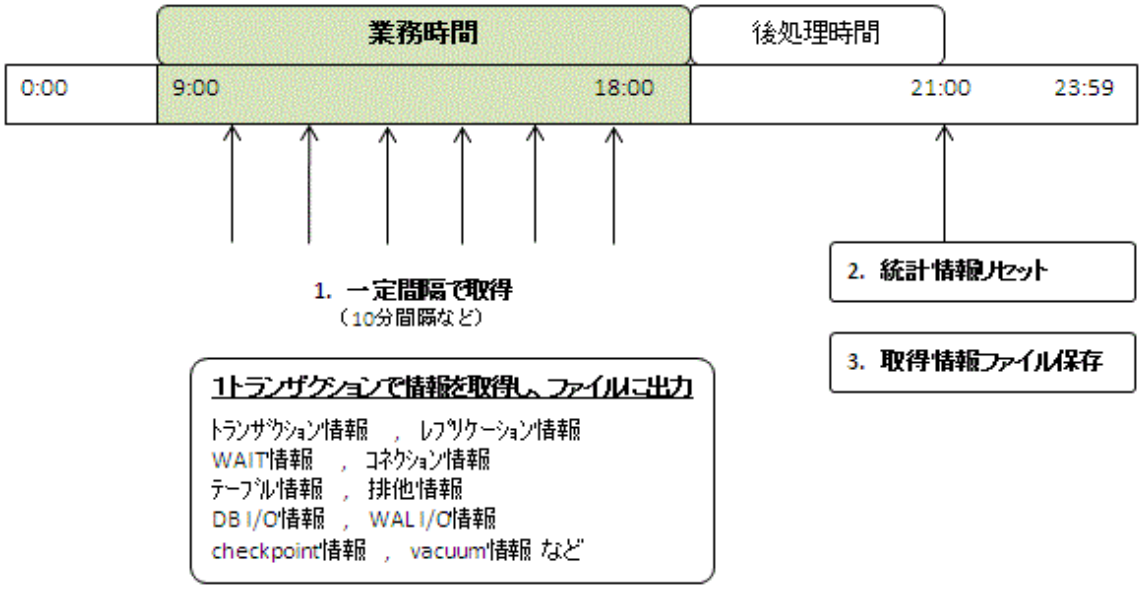
アプリケーションの互換性の維持に関する詳細は、“アプリケーション開発ガイド”の“アプリケーションの互換に関する注意事項”を参照してください。

8.6 データベース活動状況の監視

Fujitsu Enterprise Postgresでは、データベースの活動状況に関する情報を収集できます。これらを監視することで、データベース状況の変化を確認できます。

これらの情報は、内部ロックなどの待機情報を含め、性能面でのボトルネック解消に役立つ情報となります。また、当社技術員(サポート)への調査依頼時にも有益な情報となるため、取得することを推奨します。

図8.1 情報収集のイメージ



1. 業務時間帯に一定間隔で統計情報を取得します。

   取得した情報をファイルに蓄積します。

   可能な限り1つのトランザクション内で、各種の統計情報ビューのデータを取得してください。ある瞬間でのシステムの性能のスナップショットを取ることができるからです。

   取得対象のシステムビューについては、“8.6.1 取得できる情報”を参照してください。

2. 業務終了後の後処理時間帯に、統計情報をリセットします。

   リセット方法については、“8.6.3 情報のリセット方法”を参照してください。

3. 蓄積した情報ファイルを保存します。

   日々の性能の変化を確認したり、サポートに問い合わせるまで情報が削除されないようにするために、蓄積した情報ファイルは最低でも2日間は保持してください。

24時間業務の場合には、夜間などの業務負荷の低い時間帯に、統計情報のリセットおよび蓄積ファイルの保存を実施するようにしてください。

注意

統計情報は日々のデータベース状況を加算していくため、統計情報をリセットしないと、カウンタの値が上限を超えて正確な値を得られません。

ここでは、以下について説明します。

- 取得できる情報
- 取得するための設定
- 情報のリセット方法

8.6.1 取得できる情報

取得できる情報は、以下の2つに分類されます。

- PostgreSQL共通の情報
- Fujitsu Enterprise Postgresが追加した情報

PostgreSQL共通の情報

参照

PostgreSQL共通の情報については、“PostgreSQL Documentation”の“Server Administration”の“Monitoring Database Activity”を参照してください。

Fujitsu Enterprise Postgresが追加した情報

Fujitsu Enterprise Postgresが追加した情報として、以下を取得することができます。

表8.1 Fujitsu Enterprise Postgresが追加した情報

ビュー名	説明
pgx_stat_lwlock	軽量ロックの内容ごとに1行の形で、発生に関する統計情報を示します。これらの情報はボトルネックの解消につながる情報となります。 詳細については“C.2 pgx_stat_lwlock”を参照してください。
pgx_stat_latch	Fujitsu Enterprise Postgres内部の待機情報について、内容ごとに1行の形で、発生に関する統計情報を示します。これらの情報はボトルネックの解消につながる情報となります。 詳細については“C.3 pgx_stat_latch”を参照してください。
pgx_stat_walwriter	WALの書込みに関する統計情報を1行のみで表示します。詳細については“C.4 pgx_stat_walwriter”を参照してください。
pgx_stat_sql	SQL文の種類ごとに1行の形で、SQL文の発生回数に関する情報を示します。詳細については“C.5 pgx_stat_sql”を参照してください。
pgx_stat_gmc	Global Meta Cache機能のキャッシュヒット率と使用メモリサイズに関する情報を示します。詳細については“C.6 pgx_stat_gmc”を参照してください。また、Global Meta Cache機能については、“第13章 Global Meta Cache機能”を参照してください。

8.6.2 取得するための設定

情報内容に応じて、取得するための設定方法が異なります。

- PostgreSQL共通の情報
- Fujitsu Enterprise Postgresが追加した情報

PostgreSQL共通の情報

参照

PostgreSQL共通の情報については、“PostgreSQL Documentation”の“Server Administration”の“Monitoring Database Activity”の“The Statistics Collector”を参照してください。

Fujitsu Enterprise Postgresが追加した情報

Fujitsu Enterprise Postgresが追加した情報については、デフォルトで情報を収集します。

情報収集の有効/無効を変更する場合は、postgresql.conf内の設定パラメータを変更してください。以下に、情報収集の有効/無効を変更可能なビューと、設定パラメータを示します。

ビュー名	設定パラメータ
pgx_stat_lwlock pgx_stat_latch	track_waits(注)
pgx_stat_sql	track_sql
pgx_stat_gmc	track_gmc

備考) pgx_stat_walwriterは変更できません。

注) SQL文を“EXPLAIN ANALYZE”で実行する場合は、本情報の収集により通常より処理時間が増加する場合があります。“EXPLAIN ANALYZE”を実行して処理時間を確認する場合は、本パラメータを“off”にすることを推奨します。

パラメータの詳細については、“付録A パラメータ”を参照してください。

8.6.3 情報のリセット方法

情報のリセット方法を説明します。

Fujitsu Enterprise Postgresが追加した情報

Fujitsu Enterprise Postgresが追加した情報については、PostgreSQL共通の情報と同様に“pg_stat_reset_shared関数”でリセットすることができます。

pg_stat_reset_shared関数に、以下のパラメータを設定します。

関数	戻り値の型	説明
pg_stat_reset_shared(text)	void	引数に応じて、クラスタ全体の統計情報カウンタの一部をゼロに戻します(スーパーユーザー権限が必要です)。 pg_stat_reset_shared('lwlock')を呼び出すと、pgx_stat_lwlockで示される値すべてがゼロになります。 以下の場合も同様に、それぞれの統計情報カウンタの値すべてがゼロになります。 ・ pg_stat_reset_shared('latch')を呼び出した場合:

関数	戻り値の型	説明
		pgx_stat_latchで示される値すべて ・ pg_stat_reset_shared('walwriter')を呼び出した場合: pgx_stat_walwriterで示される値すべて ・ pg_stat_reset_shared('sql')を呼び出した場合: pgx_stat_sqlで示される値すべて ・ pg_stat_reset_shared('gmc')を呼び出した場合: pgx_stat_gmcのsize列以外の値すべて

参照

pg_stat_reset_shared関数のその他のパラメータについては、“PostgreSQL Documentation”の“Server Administration”の“Monitoring Database Activity”の“Statistics Functions”を参照してください。

第9章 WebAdminによるストリーミングレプリケーション

WebAdminを使用してストリーミングレプリケーションクラスタを作成する方法を説明します。

ストリーミングレプリケーションによって1つ以上のスタンバイインスタンスの作成が可能になります。スタンバイインスタンスはマスタインスタンスに接続し、WALレコードを使用してデータを複製します。スタンバイインスタンスは読み取り専用です。

WebAdminを使用してストリーミングレプリケーションクラスタを作成することができます。WebAdminによるクラスタの作成は以下の構成で可能になります。

- マスタスタンバイ構成:マスタとスタンバイインスタンスを同時に作成する構成です。
- スタンバイオンリー構成:既存のインスタンスからスタンバイインスタンスを作成する構成です。

ポイント

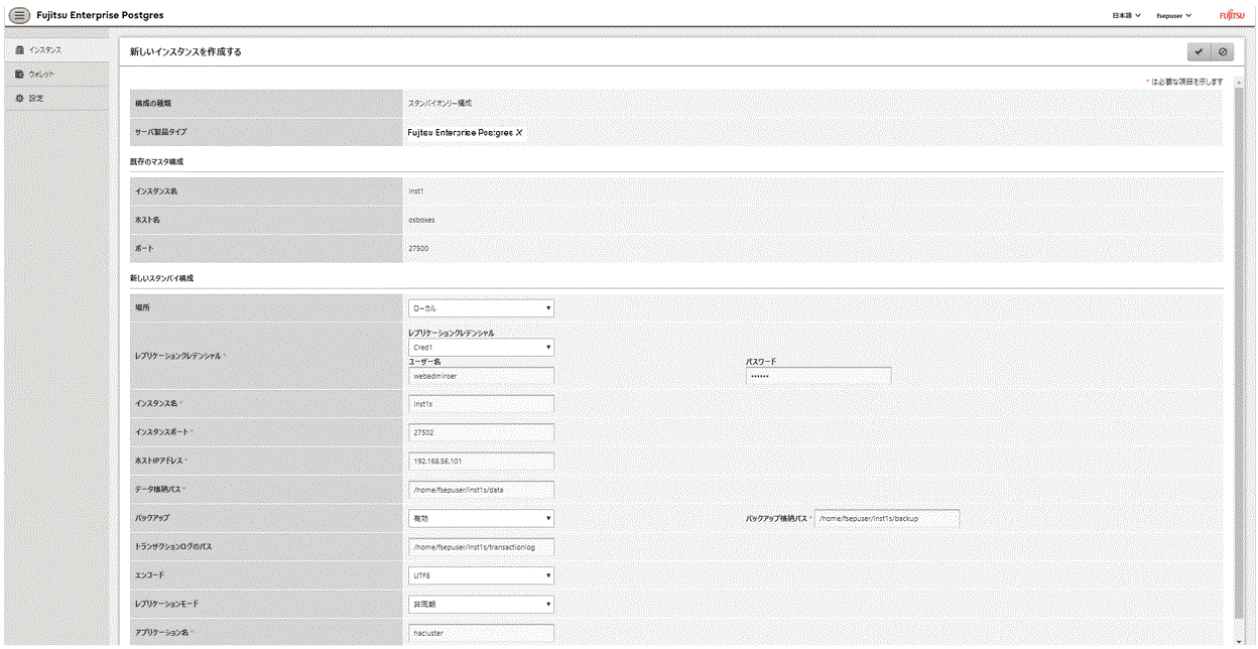
- スタンバイインスタンスは、スタンドアロンインスタンス、マスタインスタンスまたは別のスタンバイインスタンスからも作成できます。
- WebAdminを使用してストリーミングレプリケーションクラスタを作成すると、[ホスト名]に指定したホスト名(IPアドレス)が割り当てられたネットワークを、WebAdmin間で使用するネットワークとログ転送用ネットワークに使用します。
- 業務用ネットワークと異なるネットワークをログ転送用ネットワークとして使用する場合には、[ホスト名]に業務ネットワークと異なるログ転送用ネットワークが利用するホスト名を指定してください。

9.1 スタンバイインスタンスの作成

以下の手順に従ってスタンバイインスタンスを作成してください。

1. [インスタンス]タブで、スタンバイインスタンスの作成元インスタンスを選択します。
2.  をクリックします。
3. 作成するスタンバイインスタンスの情報を入力します。以下は、インスタンス"inst1"からスタンバイインスタンスを作る例です。

   選択されたインスタンスの名前、ホストアドレスおよびポートは、簡単に参照できるようにすでに表示されています。



   以下の項目を入力します。

   ー [場所]:現ユーザーがログインしているサーバにインスタンスを作成するか、リモートサーバに作成するかを設定します。初期設定は"ローカル"です。"ローカル"は、WebAdminが現在起動しているサーバマシンにインスタンスを作成します。

— [レプリケーションクレデンシャル]:マスタインスタンスに接続するために、スタンバイインスタンスのユーザー名とパスワードが必要です。ユーザー名とパスワードは入力するか、ウォレットから選択することができます。ウォレットエントリー作成の詳細は、“付録H WebAdminウォレット”を参照してください。

— [インスタンス名]:作成するスタンバイデータベースインスタンス名

以下の条件に従って設定してください。

- 最大16文字
- 先頭文字はASCII英字
- 他の文字はASCII英数字

— [インスタンスポート]:スタンバイデータベースインスタンスのポート番号

— [ホストIPアドレス]: スタンバイインスタンスが作成されるサーバマシンのIPアドレス。スタンバイインスタンスがマスタインスタンスに接続するよう設定をするために必要な情報です。

— [データ格納パス]:データベースのデータ格納先ディレクトリ

— [バックアップ格納パス]:データベースのバックアップ格納先ディレクトリ

— [トランザクションログのパス]:トランザクションログの格納先ディレクトリ

— [エンコード]:データベースの符号化方式

— [レプリケーションモード]:作成されるスタンバイインスタンスの複製モード("非同期"または"同期")

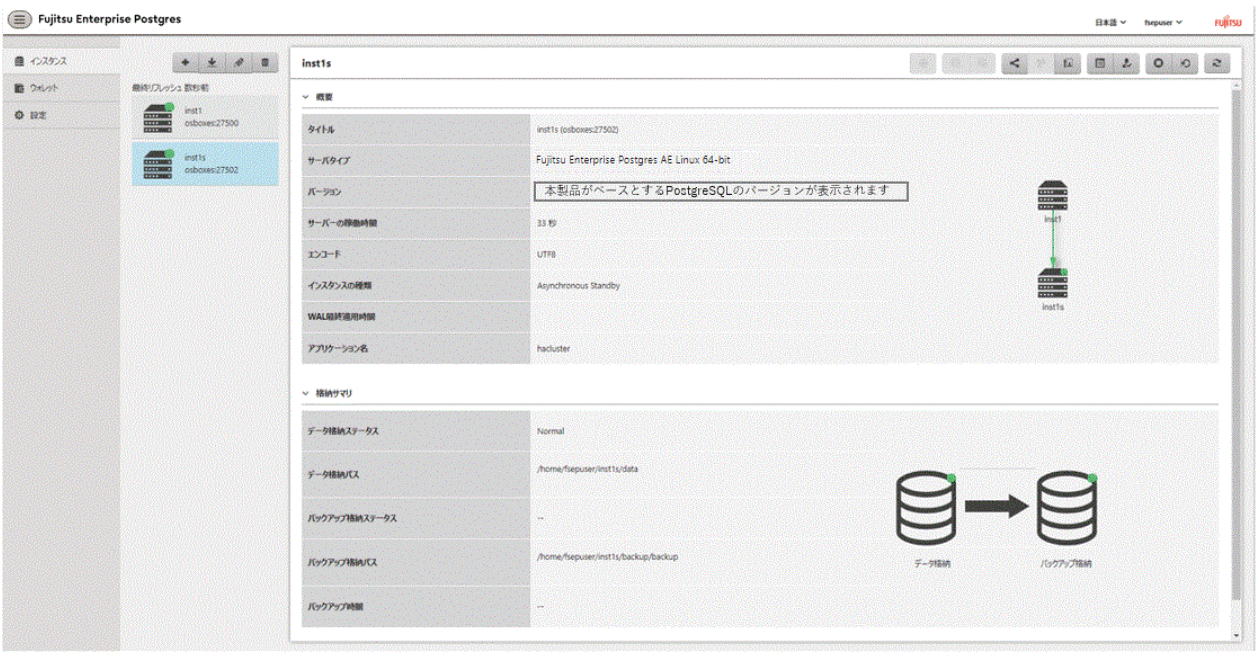
— [アプリケーション名]:マスタインスタンスに対するスタンバイインスタンスの特定に使用する参照名。

以下の条件に従って設定してください。

- 最大16文字
- 先頭文字はASCII英字
- 他の文字はASCII英数字

4. ✔をクリックしてスタンバイインスタンスを作成します。

5. スタンバイインスタンスが作成できたら、[インスタンス]タブで"inst1s"を選択します。以下のページが表示されます。



注意

- WebAdminでは、スタンバイインスタンスはバックアップができません。そのため、  と  アイコンは無効になり、[バックアップ格納ステータス]と[バックアップ時間]には数値が表示されません。
- WebAdminを使ってMirroring Controllerを管理する場合は、スタンバイインスタンスで下記メッセージがサーバログおよびシステムログに出力される場合があります。本メッセージが出力された場合でも、インスタンスは正常に運用できており、対処は不要です。

```
ERROR:  pgx_rcvall failed
ERROR:  pgx_rcvall: backup of the database has not yet been performed, or an incorrect backup storage directory was specified
```

- レプリケーションクレデンシャル(インスタンス管理者のユーザー名とパスワード)に指定できない文字については、“付録I WebAdminで使用できない文字”を参照してください。

9.2 スタンバイインスタンスの昇格

WebAdminを使用して、マスタインスタンスとスタンバイインスタンス間のストリーミングレプリケーションを中断することができます。

スタンバイインスタンスをスタンドアロンインスタンスに昇格するには、以下の手順でストリーミングレプリケーションを中断させてください。

1. [インスタンス]タブで、昇格させるスタンバイインスタンスを選択します。
2.  をクリックします。
3. 確認ダイアログボックスで[はい]をクリックします。

スタンバイインスタンスが昇格してスタンドアロンインスタンスになり、ストリーミングレプリケーションクラスタの一部ではなくなります。

スタンバイインスタンスが昇格してスタンドアロンインスタンスになると、バックアップ保存状態がエラーになります。これは、インスタンスが新規にスタンドアロンに昇格した際は、バックアップが全く無いからです。[対処]または  をクリックして新規にバックアップをすると、エラーの状態がリセットされます。

9.3 非同期レプリケーションから同期レプリケーションへの変換

マスタインスタンスとスタンバイインスタンス間のストリーミングレプリケーションを非同期または同期モードに設定することができます。このモード設定は、スタンバイインスタンスの作成が完了した後でも変更できます。

非同期スタンバイインスタンスを同期に変換するには、以下の手順に従ってください。

1. [インスタンス]タブで、関連するクラスタのマスタインスタンスを選択します。
2.  をクリックします。
3. [ストリーミングレプリケーション]セクションで、[同期スタンバイ名]の値を編集します。
   - ー 同期モードにしたいスタンバイインスタンスのアプリケーション名を追加します。
4.  をクリックします。
5. マスタインスタンスを選択して、  をクリックします。
6. スタンバイインスタンスを選択します。[インスタンスの種類]に更新された状態が表示されます。

注意

- 非同期スタンバイインスタンスを同期に変換すると、スタンバイインスタンスの準備ができるまで、マスタインスタンスは受信トランザクションをキューに追加します。このため、この操作は予定しているメンテナンス期間に行うことをお薦めします。
- 同期スタンバイインスタンスを追加する際、Fujitsu Enterprise Postgresは[同期スタンバイ名]に最初に入力されたインスタンスのみ同期状態とします。

- 同期および非同期のスタンバイモードの違いとその特徴に関しては、“PostgreSQL Documentation”の“High Availability, Load Balancing, and Replication”の“Streaming Replication”を参照してください。

9.4 同期レプリケーションから非同期レプリケーションへの変換

マスタインスタンスとスタンバイインスタンス間のストリーミングレプリケーションを非同期または同期モードに設定することができます。このモード設定は、スタンバイインスタンスの作成が完了した後でも変更できます。

同期スタンバイインスタンスを非同期に変換するには、以下の手順に従ってください。

1. [インスタンス]タブで、関連するクラスタのマスタインスタンスを選択します。
2. をクリックします。
3. [ストリーミングレプリケーション]セクションで、[同期スタンバイ名]の値を編集します。
   - 非同期モードにしたいスタンバイインスタンスのアプリケーション名を削除します。
4. をクリックします。
5. マスタインスタンスを選択して、をクリックします。
6. スタンバイインスタンスを選択します。[インスタンスの種類]に更新された状態が表示されます。

注意

同期および非同期のスタンバイモードの違いとその特徴に関しては、“PostgreSQL Documentation”の“High Availability, Load Balancing, and Replication”の“Streaming Replication”を参照してください。

9.5 レプリケーションクラスタの結合

WebAdminでは、古いマスタインスタンスを待機ノードとして簡単に結合することができます。

1. [インスタンス]タブで、新しいクラスタノードがWALエントリをストリームする場所からリモートインスタンスを選択し、をクリックします。
2. 新しいノードからストリーミング要求を受け入れるようにノードを設定します。
3. [インスタンス]タブで、クラスタに接続する必要がある新しいスタンバイインスタンスを選択し、をクリックします。
4. [レプリケーションホスト名]をリモートインスタンスに設定します。
5. [レプリケーションクレデンシャル]を入力します。

   リモートインスタンスに接続するために、スタンバイインスタンスのユーザー名とパスワードを指定する必要があります。ユーザー名とパスワードは入力するか、ウォレットから選択することができます。ウォレットエントリ作成の詳細は、“付録H WebAdminウォレット”を参照してください。レプリケーションクレデンシャル(インスタンス管理者のユーザー名とパスワード)に指定できない文字については、“付録I WebAdminで使用できない文字”を参照してください。
6. [ホストIPアドレス]を入力します。

   スタンバイインスタンスが作成されるノードのIPアドレスを指定します。
7. をクリックして、[レプリケーションクラスタの結合]ダイアログボックスを開きます。

   [はい]をクリックしてスタンバイインスタンスをセットアップします。
8. 正常に完了すると、確認ダイアログボックスが表示されます。
9. [閉じる]ボタンをクリックして、インスタンスの詳細画面に戻ります。

インスタンスがスタンバイインスタンスになり、ストリーミングレプリケーションクラスタの一部になります。レプリケーション図には、スタンバイインスタンスとリモートインスタンスの関係が表示されます。[設定]画面では、リモートインスタンスのレプリケーション関係を非同期から同期、または同期から非同期に変更できます。

第10章　インメモリ機能の導入と運用

インメモリ機能は、カラムナ型インデックス(Vertical Clustered Index; VCI)とメモリレジデント機能により、集計処理を高速に行う機能です。

VCIは、集計処理に適したデータ構造です。さらに並列検索、およびディスク圧縮機能を備えるため、ディスクI/Oを削減した集計処理の高速化が可能です。

メモリレジデント機能は、集計処理で発生するディスクI/Oを削減する機能です。VCIのデータを事前にメモリに読み出す事前ロード機能、またメモリからのVCIデータの追い出しを抑えるステーブルバッファ機能から構成されます。このステーブルバッファ機能では、共有メモリのうちパラメータで指定した割合をVCI用のメモリとして確保します。

本章では、インメモリ機能の導入と運用について説明します。

10.1 Vertical Clustered Index(VCI)の導入

VCIの導入について説明します。

VCIの導入は、以下の流れで実施します。

導入の流れ

1. 導入の検討
2. リソースの見積り
3. セットアップ

10.1.1 導入の検討

VCIは、サーバ内の空きリソースを活用して、検索処理の高速化を実現します。

様々な場面で高速化が望めますが、特に以下の条件を多く満たす場合に、高い効果を得ることができます。

- 単一表の処理である
- テーブルの多くの行を扱う処理である
- テーブルの一部の列を扱う処理である
- sumやaverageなど複数の集約処理を同時に実行するような、非常に重い集計を行う処理である

以下の場合は、VCIが利用されないため、事前に有効性の判断が必要です。

- 対象のテーブルや列のデータ型が、VCIの制限事項を含む場合
- 対象のSQL文が、VCIの動作条件を満たさない場合
- コスト計算によりVCIが遅いと判断された場合

注意

VCIを利用した運用を行う場合には、postgresql.confのfull_page_writesパラメータの設定が有効(on)でなければなりません。そのため、本パラメータが無効(off)となっている場合には、VCIに対する一連の操作がエラーとなります。また、一時的にfull_page_writesパラメータの設定を無効(off)にしてVCIを作成しないテーブルについて操作を行う場合には、その間、VCIの作成やVCIを作成したテーブルに対する操作を行わないでください。

参照

- VCIの制限事項については、“10.1.4 VCIを利用できるデータ”を参照してください。
- VCIの動作条件については、“アプリケーション開発ガイド”の“Vertical Clustered Index(VCI)を利用した検索”の“動作条件”を参照してください。

10.1.2 リソースの見積り

セットアップの事前に、リソースの見積りを行います。

高速化したい集計処理を決定し、必要とする列データを洗い出してください。列数に応じて、以下の資源が追加で必要です。

- メモリ

  VCIの作成対象となる列のディスク容量分のメモリを追加で用意してください。

- ディスク

  VCIでは、既存のテーブルデータに加え、列データをディスク上に保持するため、VCI作成対象の列のディスク容量をもとに、追加でディスクを用意してください。この際、I/Oによる他業務との影響を避けるため、従来のディスクとは別にディスクを用意し、テーブルスペースとして指定することを推奨します。

参考

VCIの使用メモリ量が設定値を超えた場合には、ディスク上のデータを使って動作します。

参照

メモリとディスクの見積り式については、“導入ガイド(サーバ編)”の“メモリの見積り”、および“データベースのディスク容量の見積り”を参照してください。

10.1.3 セットアップ

VCIのセットアップについて説明します。

VCIのセットアップは、以下の流れで実施します。

セットアップの流れ

1. パラメータの設定
2. 拡張のインストール
3. VCIの作成
4. VCIが作成されていることの確認

10.1.3.1 パラメータの設定

postgresql.confファイルを編集し、VCIに必要なパラメータを設定します。postgresql.confファイルの編集後は、インスタンスを起動、または再起動してください。

事前に設定が必要、または推奨するパラメータについて示します。

パラメータ名	設定値	説明	パラメータの省略
shared_preload_libraries	vci, pg_prewarm	VCI、および事前ロード機能の共用ライブラリ名を指定します。	省略不可
session_preload_libraries	vci, pg_prewarm	VCI、および事前ロード機能の共用ライブラリ名を指定します。	省略不可
reserve_buffer_ratio	ステーブルバッファテーブルに使用する共有メモリの割合(%)	共有メモリのうち、ステーブルバッファテーブルに使用する割合を、パーセンテージの値で指定します。	省略可

パラメータ名	設定値	説明	パラメータの省略
vci.control_max_workers	VCIを管理するバックグラウンドワーカーの数	VCIを管理するバックグラウンドワーカー数を指定します。 この値を max_worker_processesに加える必要があります。	省略可
vci.max_parallel_degree	並列検索で使用するバックグラウンドワーカーの最大数	並列検索で使用するバックグラウンドワーカーの最大数を指定します。 この値を max_worker_processesに加える必要があります。	省略可

例

```
shared_preload_libraries = 'vci, pg_prewarm'
session_preload_libraries = 'vci, pg_prewarm'
reserve_buffer_ratio = 20
vci.control_max_workers = 8
vci.max_parallel_degree = 4
max_worker_processes = 18 # 例. 元が6であった場合、6 + 8 + 4 = 18
```

注意

VCIを使用する場合、procfsがマウントされていない状態でインスタンスの起動を行うと、エラーが発生します。インスタンスの起動時は、事前にprocfsのマウント状態を確認してください。

参照

VCI用の全パラメータについては、“付録A パラメータ”を参照してください。また、各パラメータのデフォルト値、および指定範囲などの詳細についても同様に参照してください。ただし、“shared_preload_libraries”、“session_preload_libraries”、“max_worker_processes”の詳細は、“PostgreSQL Documentation”の“Server Administration”の“Server Configuration”を参照してください。

10.1.3.2 拡張のインストール

CREATE EXTENSION文を実行し、VCIとpg_prewarmの拡張をインストールします。拡張のインストールはデータベース単位に必要です。

- VCIのインストール

```
db01=# CREATE EXTENSION vci;
```

- pg_prewarmのインストール

```
db01=# CREATE EXTENSION pg_prewarm;
```

注意

- VCIの拡張インストールは、スーパーユーザーのみ実行可能です。
- VCIの拡張インストールは、publicスキーマに対してのみ実行可能です。

- VCIの拡張に対して実行できない操作があります。詳細は、“10.2.1 VCIに対して利用できないコマンド”を参照してください。

10.1.3.3 VCIの作成

CREATE INDEXのUSING句に“vci”を指定して実行し、対象となる列にVCIを作成します。また、WITH句のstable_bufferパラメータに“true”を指定し、ステーブルバッファ機能を有効にします。

VCI用にディスクを分ける場合は、TABLESPACE句に適切なテーブルスペースを指定してください。

```
db01=# CREATE INDEX idx_vci ON table01 USING vci (col01, col02) WITH (stable_buffer=true);
```

注意

- CREATE INDEXのON句に指定できないテーブル種別があります。詳細は、“10.1.4.1 リレーション種別”を参照してください。
- CREATE INDEXの列指定に指定できないデータ型があります。詳細は、“10.1.4.2 データ型”を参照してください。
- VCIに対して、実行できない操作があります。詳細は、“10.2.1 VCIに対して利用できないコマンド”を参照してください。
- CREATE INDEXの列指定に、同じ列を複数指定することはできません。
- テンプレートデータベースに属するテーブルの列に対して、VCIを作成することはできません。
- CREATE INDEXを実行すると、VCIの他に、“vci_{10桁のリレーションOID}_{5桁のリレーション属性}_{リレーション種別を示す1桁の数値、または文字}”という名前で、複数のビューが作成されます。これらのビューを、VCI内部リレーションといいます。VCI内部リレーションはVCIの集計処理で使用する資源のため、更新、削除は行わないでください。
- VCIの作成では、指定した列のすべてのデータを列形式に置き換えるため、データが挿入された既存テーブルに対してCREATE INDEXを実行する場合は、一般的なインデックス(B-tree)に比べ時間を要します。ただし、CREATE INDEX中の業務の継続は可能です。
- CREATE INDEX USING VCIがパーティションテーブルに実行されたときのデフォルトの振る舞いは、全パーティションが一致するインデックスを持つようにする全パーティションへの再帰的な実行です。各パーティションは最初に同等のインデックスがすでに存在するかの判断のために検査され、存在するならば、作成するインデックスに対するパーティションインデックスとしてアタッチされます。新たに作成するインデックスが既存インデックスの親インデックスとなります。一致するインデックスが存在しない場合、新たなインデックスが作られて、自動的にアタッチされます。各パーティションの新たなインデックス名は、コマンドでインデックスが指定されなかった場合と同様に決定されます。ONLYオプションが指定された場合、再帰処理は行われず、そのインデックスは無効とマークされます(ALTER INDEX ... ATTACH PARTITIONは、全パーティションが一致するインデックスを取得すると、インデックスを有効にマークします)。しかしながら、ONLYが指定されたとしても、将来的には、CREATE TABLE ... PARTITION OFを使って作成されるあらゆるパーティションは自動的に一致するインデックスを持つことに注意してください。
- 並列インデックスの構築は、VCIインデックスではサポートされていません。

10.1.3.4 VCIが作成されていることの確認

SELECT文の実行によりpg_indexesカタログを参照し、対象となる列にVCIが作成されていることを確認します。

例

```
db01=# SELECT indexdef FROM pg_indexes WHERE indexdef LIKE '%vci%';
                          indexdef
-------------------------------------------------------
CREATE INDEX idx_vci ON table01 USING vci (col01, col02)
(1 row)
```

10.1.4 VCIを利用できるデータ

各リレーション種別やデータ型に対する、VCI作成可否の詳細を説明します。

10.1.4.1 リレーション種別

リレーション種別によっては、VCIを作成することができません。

VCI作成対象のリレーション種別、および対象外のリレーション種別について以下に示します。

作成対象外のリレーション種別は、“10.1.3.3 VCIの作成”で示したCREATE INDEXのON句に指定することができません。

- VCI作成対象のリレーション
  - 通常テーブル
  - UNLOGGED TABLE
- VCI作成対象外のリレーション
  - マテリアライズド・ビュー
  - 一時テーブル
  - ビュー
  - 一時ビュー
  - 外部テーブル

10.1.4.2 データ型

列のデータ型によっては、VCIを作成することができません。

VCI作成対象の列のデータ型、および作成対象外の列のデータ型について、以下の表に示します。

作成対象外のデータ型の列は、“10.1.3.3 VCIの作成”のCREATE INDEXの列指定に指定することができません。

データ型の分類	型名	作成対象/対象外
数値データ型	smallint	○
	integer	○
	bigint	○
	decimal	○
	numeric	○
	real	○
	double precision	○
	serial	○
	bigserial	○
通貨型	money	○
文字列型	varchar(n)	○
	char(n)	○
	nchar	○
	nvarchar(n)	○
	text	○
バイナリ列データ型	bytea	○
日付/時刻データ型	timestamp	○
	timestamp with time zone	○
	date	○
	time	○

データ型の分類	型名	作成対象/対象外
	time with time zone	○
	interval	○
論理値データ型	boolean	○
幾何データ型	point	×
	line	×
	lseg	×
	box	×
	path	×
	polygon	×
	circle	×
ネットワークアドレス型	cidr	×
	inet	×
	macaddr	×
	macaddr8	×
ビット列データ型	bit(n)	○
	bit varying(n)	○
テキスト検索に関する型	tsvector	×
	tsquery	×
UUID型	uuid	○
XML型	xml	×
JSON型	json	×
	jsonb	×
範囲型	int4range	×
	int8range	×
	numrange	×
	tsrange	×
	tstzrange	×
	daterange	×
オブジェクト識別子データ型	oid	×
	regproc	×
	regprocedure	×
	regoper	×
	regoperator	×
	regclass	×
	regtype	×
	regconfig	×
	regdictionary	×
pg_lsn型	pg_lsn	×
配列型	-	×

データ型の分類	型名	作成対象/対象外
ユーザ定義型 (基本型、列挙型、複合型、範囲型)	-	×

○:VCI作成対象

×:VCI作成対象外

10.2 VCIの運用

VCIの運用方法について説明します。

10.2.1 VCIに対して利用できないコマンド

VCIの拡張、およびVCIそのものに対して、実行できない操作があります。

VCIの拡張とVCIに対して実行できないSQLコマンド、およびクライアントアプリケーションコマンドについて、以下に示します。

SQLコマンド

・ VCIの拡張に対して実行できない操作

コマンド	サブコマンド	説明
ALTER EXTENSION	UPDATE	VCIの拡張を指定できません。 VCIでは不要な操作であるためです。
	SET SCHEMA	
	ADD	
	DROP	
CREATE EXTENSION	SCHEMA	VCIの拡張指定時は、左記のサブコマンドを指定できません。 VCIでは不要な操作であるためです。

・ VCIに対して実行できない操作

コマンド	サブコマンド	説明
ALTER INDEX	SET	VCI指定時は、左記のサブコマンドを指定できません。 操作が必要な場合は、いったんVCIをDROP INDEXで削除し、操作後に再度CREATE INDEXで作成してください。
	SET TABLESPACE	
	ALL IN TABLESPACE	
ALTER OPERATOR CLASS	RENAME TO	VCI指定時は、左記のサブコマンドを指定できません。 VCIではサポートしていない操作です。
	OWNER TO	
	SET SCHEMA	
ALTER OPERATOR FAMILY	ADD	
	DROP	
	RENAME TO	
	OWNER TO	
	SET SCHEMA	
ALTER TABLE	ALL IN TABLESPACE name [OWNED BY role_name] SET TABLESPACE new_tablespace	VCIを含むテーブルスペースを指定できません。

コマンド	サブコマンド	説明
		操作が必要な場合は、いったんVCIをDROP INDEXで削除し、操作後に再度CREATE INDEXで作成してください。
	DROP [COLUMN] [IF EXISTS] column_name [RESTRICT | CASCADE]	VCIを含む列を指定できません。 操作が必要な場合は、いったんVCIをDROP INDEXで削除し、操作後に再度CREATE INDEXで作成してください。
	ALTER [COLUMN] column_name [SET DATA] TYPE data_type [COLLATE collation] [USING expression]	
	CLUSTER ON index_name	VCIを指定できません。 VCIではサポートしていない操作です。
	REPLICA IDENTITY {DEFAULT | USING INDEX index_name | FULL | NOTHING}	
CLUSTER	-	VCIを含むテーブル、およびVCIを指定できません。 操作が必要な場合は、いったんVCIをDROP INDEXで削除し、操作後に再度CREATE INDEXで作成してください。
CREATE INDEX	UNIQUE	VCI指定時は、左記のサブコマンドを指定できません。 VCIではサポートしていない操作です。
	CONCURRENTLY	
	[ASC | DESC]	
	[NULLS { FIRST | LAST }]	
	WITH	
	WHERE	
	INCLUDE	
CREATE OPERATOR CLASS	-	VCIを指定できません。 VCIではサポートしていない操作です。
CREATE OPERATOR FAMILY	-	
CREATE TABLE	EXCLUDE	
DROP INDEX	CONCURRENTLY	VCI指定時は、左記のサブコマンドを指定できません。 VCIではサポートしていない操作です。
REINDEX	-	VCIを指定できません。 VCIはデーモンによる自動メンテナンスでディスクスペース増加を防いでいるため、REINDEXは不要です。

クライアントアプリケーションコマンド

・ VCIに対して実行できない操作

コマンド	説明
clusterdb	VCIを含むテーブルに対して、クラスタ化の実行はできません。
reindexdb	--indexにVCIを指定できません。

10.2.2 データの事前ロード機能

インスタンス起動直後は、VCIのデータがバッファに乗っていないため、VCIを利用した1回目の集計処理に時間が掛かる場合があります。

このため、インスタンス起動後にVCIの集計処理を行う場合は、VCIのデータを事前にバッファにロードする、事前ロード機能を使用する必要があります。

事前ロード機能を使用する場合は、関数pgx_prewarm_vciをCREATE INDEXで作成したVCI単位に実行してください。

```
db01=# SELECT pgx_prewarm_vci('idx_vci');
```

参照

pgx_prewarm_vciの詳細は、“B.4 VCIデータのロード制御関数”を参照してください。

第11章　パラレルクエリ

Fujitsu Enterprise Postgresでは、以下についてパラレルクエリが強化されています。

- CPUの負荷計算
- クエリ実行時のワーカーの増加
- 統計情報ビューによる動作状況の表示

11.1 CPUの負荷計算

パラレルクエリの実行時に利用可能なCPUが不足する場合があります。この時点で動的ワーカーを追加しても、コンテキスト切り替えによるオーバーヘッドが発生する可能性があります。

Fujitsu Enterprise Postgresでは、パラレルクエリのワーカー数を決定する際に現時点のCPU負荷を考慮して、ワーカー数を決定します。

11.2 クエリ実行時のワーカーの増加

パラレルクエリでは、クエリの開始時にワーカーの不足やワーカーが存在しないなどの理由で、ワーカーを追加できずにCPUの過負荷が発生する可能性があります。

Fujitsu Enterprise Postgresでは、利用可能なワーカーが存在する場合に、クエリ実行中に追加のワーカーを割り当てることができます。これにより、クエリのパフォーマンスが向上します。

注意

パラレルクエリを行う場合にのみクエリ実行時のワーカー増加が可能です。

11.3 統計情報ビューによる動作状況の表示

既存の統計情報コレクタで動作状況を確認することができます。これらを監視することで、システム負荷の変化などによるパラレルクエリの動作状況を確認することができます。

以下の表に、統計情報ビューに追加された列を説明します。

pg_stat_all_tables

列	型	説明
parallel_query	bigint	テーブル上で初期化されたパラレルクエリの個数です。

以下のビューについても“pg_stat_all_tables”と同様に、“parallel_scan”の列が追加されています。

- pg_stat_sys_tables
- pg_stat_user_tables
- pg_stat_xact_all_tables
- pg_stat_xact_sys_tables
- pg_stat_xact_user_tables

参照

統計情報コレクタおよびビューについては、“PostgreSQL Documentation”の“Server Administration”の“The Statistics Collector”を参照してください。

また、パラレルクエリが動作すると、Fujitsu Enterprise Postgresのpgx_stat_sqlビューに値が設定されます。

参照

pgx_stat_sqlビューについては、“C.5 pgx_stat_sql”を参照してください。

第12章　高速データロード機能

高速データロード機能は、pgx_loaderコマンドを使用して、外部ファイルのデータをFujitsu Enterprise Postgresのテーブルに高速ロードする機能です。

注意

- 本機能は、シングルユーザーモードで使用することはできません。インスタンスが1つのプロセスで稼働しており、並列ワーカを起動できないためです。

12.1 高速データロード機能の導入

高速データロード機能の導入について説明します。

導入の流れ

1. 導入の検討
2. リソースの見積り
3. セットアップ

12.1.1 導入の検討

高速データロード機能は、与えられたCOPY FROMコマンドを並列実行することで、データロード処理の高速化を実現します。PostgreSQLのCOPY FROMコマンドよりも多くのリソースを使用するため、データベースシステムが十分なリソースを利用できない場合は、PostgreSQLのCOPY FROMコマンドよりもロード性能が劣化する場合があります。そのため、事前に以下の観点を考慮することで、有効性を判断してください。

データベースサーバのメモリ量

postgresql.confのshared_buffersの値が小さい場合、データベースサーバの共有メモリにキャッシュされるデータページ数が少なくなります。そのため、複数の並列ワーカによる同一データページへの書き込み処理の排他待ちが多発します。また、データページ数が少ないと、テーブルの拡張頻度が増大します。テーブルの拡張処理中は、テーブルへのアクセスが排他されるため(待機イベント名:extend)、ロード時間が増大します。この場合は、shared_buffersの値を大きくしてください。

参照

待機イベント名は、pg_stat_activityビューのwait_event列に格納されます。待機イベント名の詳細は、“PostgreSQL Documentation”の“The Statistics Collector”の“wait_event Description”を参照してください。

チェックポイント処理の発生頻度

短い間隔でチェックポイント処理が発生するとロード性能が低下します。データロード中に以下のようなメッセージがサーバログに出力された場合は、postgresql.confのmax_wal_sizeやcheckpoint_timeoutに大きな値を設定することで、チェックポイントの発生頻度を減らしてください。

例

```
LOG:  checkpoints are occurring too frequently (19 seconds apart)
HINT:  Consider increasing the configuration parameter "max_wal_size".
```

12.1.2 リソースの見積り

高速データロード機能が使用するメモリ量を見積ります。

高速データロード機能は、データロード処理を実行する並列ワーカの数を最大128まで指定可能です。並列ワーカの数に応じて、以下の資源が追加で必要です。

- データロード処理中に作成される動的共有メモリ

  高速データロード機能は、データロード処理中に動的共有メモリや共有メモリメッセージキューを作成します。これは、バックエンドと並列ワーカ間での外部データの授受、およびエラー通知に使用されます。

注意

“12.1.1 導入の検討”の“データベースサーバのメモリ量”にあるように、postgresql.confのshared_buffersの値が小さい場合、同一データページへの書き込み処理の排他待ちが多発します。待機中は、共有メモリメッセージキューから入力データを読み込めないため、共有メモリメッセージキューは常に入力データで満たされた状態となります。これにより、共有メモリメッセージキューへの書き込み処理が阻害されるため、データロード性能が劣化します。

参照

メモリの見積り式については、“導入ガイド(サーバ編)”の“高速データロード機能で使用するメモリの見積り式”を参照してください。

12.1.3 セットアップ

高速データロード機能のセットアップについて説明します。

セットアップの流れ

1. パラメータの設定
2. 拡張のインストール

12.1.3.1 パラメータの設定

高速データロード機能に必要なパラメータをpostgresql.confに設定します。

postgresql.confの以下のパラメータについては、すでに設定されている値に下表の設定値を追加します。postgresql.confの編集後は、インスタンスを起動、または再起動してください。

パラメータ名	設定値	パラメータの省略
max_prepared_transactions	データロード処理中に並列ワーカが準備するトランザクションの数を指定します。本機能で使用する並列ワーカの数以上でなければなりません。この値を、max_prepared_transactionsに加える必要があります。	省略不可
max_worker_processes	データロード処理を実行する並列ワーカの数を指定します。この値を、max_worker_processesに加える必要があります。	省略不可
max_parallel_workers	本機能を含む並列クエリで使用する並列ワーカの最大数を指定します。本機能で使用する並列ワーカの数以上でなければなりません。この値を、max_parallel_workersに加える必要があります。	省略不可

例

2つの高速データロード機能を並列度:4で同時実行した場合の例を示します。

```
max_prepared_transactions = 13  #例. 元が5であった場合、 5 + 2 x 4 = 13
max_worker_processes = 16       #例. 元が8であった場合、 8 + 2 x 4 = 16
max_parallel_workers = 12       #例. 元が4であった場合、 4 + 2 x 4 = 12
```

注意

上記の例に示すように、max_prepared_transactions、max_worker_processesおよびmax_parallel_workersには、本機能の同時実行数を乗算した値を設定してください。

また、以下のパラメータについては、値を確認します。

パラメータ名	設定値	パラメータの省略
dynamic_shared_memory_type	インスタンスが使用する動的共有メモリの実装を指定します。デフォルト値を推奨します。	省略不可

参照

パラメータの詳細は、“PostgreSQL Documentation”の“Resource Consumption”を参照してください。

12.1.3.2 拡張のインストール

CREATE EXTENSION文を実行し、高速データロード機能の拡張をインストールします。拡張のインストールはデータベース単位に必要です。

例

対象のデータベースを“postgres”とした場合の例を示します。

```
postgres=# CREATE EXTENSION pgx_loader;
CREATE EXTENSION
```

注意

- 高速データロード機能の拡張インストールは、スーパーユーザーのみ実行可能です。
- 高速データロード機能の拡張インストールは、publicスキーマに対してのみ実行可能です。

12.2 高速データロード機能の利用

高速データロード機能の利用方法について説明します。

12.2.1 データロード

外部ファイルのデータをFujitsu Enterprise Postgresのテーブルにロードするには、pgx_loaderコマンドのloadモードを実行します。

例

外部ファイル“/path/to/data.csv”(2000レコード)をテーブル“tbl”に並列度:3でロードした場合の例です。

```
$ pgx_loader load -j 3 -c "COPY tbl FROM '/path/to/data.csv' WITH CSV"
LOAD 2000
```

ポイント

外部ファイルに、書式や制約に違反する不良データが含まれる場合、データロードが途中で失敗し、夜間バッチ処理などの定型業務の遅延につながります。そのため、データロードを実行する前に、不良データを除去することを推奨します。

注意

本機能で挿入されたデータは、pg_dumpコマンドおよびpg_dumpallコマンドにより、COPYコマンドとしてダンプされます。

参照

- pgx_loaderコマンドの詳細は、“リファレンス”の“pgx_loader”を参照してください。
- 外部ファイルの配置先とアクセス権限の詳細は、“PostgreSQL Documentation”の“COPY”を参照してください。

12.2.2 進捗状況の確認

大きな外部ファイルを入力としたデータロードを実行する場合、ロード中に進捗情報を取得することで、処理が継続的に行われていることを確認することができます。進捗情報は、pgx_stat_progress_loaderビューにより取得することができます。本ビューでは、バックエンドプロセスと並列数分のワーカプロセスの進捗情報の合計が表示されます。SELECT文などでpgx_stat_progress_loaderビューを検索し、該当する行を特定します。pgx_loaderコマンドを実行した後に、pg_stat_activityビューを参照し、取得したPIDをキーにpgx_stat_progress_loaderビューの行を特定します。

例

1. pg_stat_activityビューを参照します。(9311 : バックエンドプロセス、9312,9313,9314 : ワーカプロセス)

```
postgres=# select pid, application_name, backend_type from pg_stat_activity
pid  | application_name |         backend_type
-----+------------------+------------------------------
6216 |                  | autovacuum launcher
6218 |                  | logical replication launcher
6271 | psql             | client backend
9311 | pgx_loader       | client backend
9312 |                  | parallel loader for PID 9311
9313 |                  | parallel loader for PID 9311
9314 |                  | parallel loader for PID 9311
6214 |                  | background writer
6213 |                  | checkpointer
6215 |                  | walwriter
```

2. pgx_stat_progress_loaderビューの情報を確認します。

```
postgres=# SELECT * FROM pgx_stat_progress_loader
pid  | datid | datname | relid | command  | type | bytes_processed | bytes_total | tuples_processed | tuples_excluded
-----+-------+---------+-------+----------+------+-----------------+-------------+------------------+
+-----------------
9311 |  222  | testdb  | 333   |COPY FROM | FILE |          192000 |    450000   |          189000  |            3000
```

pgx_stat_progress_loaderビューの詳細は、“C.8 pgx_stat_progress_loader”を参照してください。

注意

pgx_loaderコマンドを実行した際に、PostgreSQLのpg_stat_progress_copyビューには、バックエンドプロセスと並列数分のワーカプロセスの進捗情報が各々の行で出力されます。バックエンドプロセスの進捗情報のtuples_processed、tuples_excluded は0が表示されます。また、ワーカプロセスのbytes_processed、bytes_totalは0が表示されます。

```
postgres=# SELECT * FROM pg_stat_progress_copy
pid  | datid | datname | relid | command  | type | bytes_processed | bytes_total | tuples_processed | tuples_excluded
-----+-------+---------+-------+----------+------+-----------------+-------------+------------------+----------------
9311 |  222  | testdb  | 333   |COPY FROM | FILE |          192000 |    450000   |              0   |              0
9312 |  222  | testdb  | 333   |COPY FROM | FILE |               0 |         0   |          63000   |           1000
9313 |  222  | testdb  | 333   |COPY FROM | FILE |               0 |         0   |          63000   |           1000
9314 |  222  | testdb  | 333   |COPY FROM | FILE |               0 |         0   |          63000   |           1000
```

pg_stat_progress_copyビューの詳細は、“PostgreSQL Documentation”の“pg_stat_progress_copy View”を参照してください。

12.2.3 インスタンスの異常終了からのリカバリ

高速データロード機能の利用中に、サーバがダウンするなどのシステム障害が発生すると、本機能により準備されたトランザクションがインダウト状態になることがあります。このとき、トランザクションで占有した資源がロックされ、他のトランザクションから当該資源へのアクセスがブロックされて利用不可になります。

このような場合は、インダウト状態になっているトランザクションを確認し、インダウトトランザクションを解決します。

インダウトトランザクションの確認

インダウトトランザクションの確認方法について説明します。

1. pgx_loaderスキーマのpgx_loader_stateテーブルの参照

   pgx_loaderスキーマのpgx_loader_stateテーブルを参照して、インダウトトランザクションのグローバルトランザクション識別子(gid列)を取得します。state列が“rollback”の行がインダウトトランザクションに関する情報です。

   #### 例

   データベースロール“myrole”でテーブル“tbl”に関連するインダウトトランザクションのグローバルトランザクション識別子を取得する例を示します。この例では、グローバルトランザクション識別子“pgx_loader:9589”および“pgx_loader:9590”がインダウトトランザクションとなります。

   ```
   postgres=# SELECT gid, state FROM pgx_loader.pgx_loader_state WHERE
   postgres-# role_oid IN (SELECT oid FROM pg_roles WHERE rolname = 'myrole') AND
   postgres-# relation_oid IN (SELECT relid FROM pg_stat_all_tables WHERE
   postgres-# relname = 'tbl');
          gid       | state
   -----------------+-------------
    pgx_loader:9590 | rollback
    pgx_loader:9591 | commit
    pgx_loader:9589 | rollback
   (3 rows)
   ```

2. pg_prepared_xactsシステムビューの参照

   pg_prepared_xactsシステムビューを参照して、上記で取得したインダウトトランザクションの有無を再確認します。

   #### 例

   グローバルトランザクション識別子“pgx_loader:9589”および“pgx_loader:9590”をもつインダウトトランザクションの有無を再確認する例を示します。

```
postgres=# SELECT gid FROM pg_prepared_xacts WHERE gid IN ('pgx_loader:9589','pgx_loader:9590');
       gid
------------------
 pgx_loader:9590
 pgx_loader:9589
(2 rows)
```

参照

pgx_loader_stateテーブルの詳細は、“F.1 pgx_loader_state”を参照してください。

インダウトトランザクションの解決

インダウトトランザクションの解決には、pgx_loaderコマンドのrecoveryモードを使用します。

pgx_loaderコマンドのrecoveryモードを使用した後に、“インダウトトランザクションの確認”を実行して、インダウトトランザクションが解決されたことを確認してください。

例

テーブル“tbl”に関連するインダウトトランザクションを解決する例を示します。

```
$ pgx_loader recovery -t tbl
```

ポイント

pgx_loaderコマンドのrecoveryモードは、高速データロード機能が準備したトランザクションだけを解決します。本機能以外の、分散トランザクションを利用したアプリケーションが準備したトランザクションは“16.13 分散トランザクションの異常時の対処”に従って解決してください。

12.3 高速データロード機能のアンセットアップ

高速データロード機能のアンセットアップについて説明します。

12.3.1 拡張のアンインストール

DROP EXTENSION文を実行し、高速データロード機能の拡張をアンインストールします。拡張のアンインストールはデータベース単位に必要です。

例

対象のデータベースを“postgres”とした場合の例を示します。

```
postgres=# DROP EXTENSION pgx_loader;
DROP EXTENSION
```

注意

- pgx_loaderスキーマのpgx_loader_stateテーブルには、本機能の動作に必要な情報が格納されているため、1行でも存在した場合は、本機能の拡張をアンインストールしないでください。
- 本機能の拡張アンインストールは、スーパーユーザーのみ実行可能です。

・ 本機能の拡張アンインストールは、publicスキーマに対してのみ実行可能です。

第13章　Global Meta Cache機能

Global Meta Cache(GMC)機能は、pgx_global_metacacheパラメータを使用することでメタキャッシュを共有メモリ上にロードする機能です。これにより、システム全体で必要なメモリ量を減らすことができます。

13.1 利用方法

Global Meta Cache機能の利用方法について説明します。

13.1.1 Global Meta Cache機能を有効にするかどうかの判断

Global Meta Cacheは、メタキャッシュをプロセス間で共有する仕組みであるため、アクセスするリソース数やSQL接続数が多いシステムで効果があります。このリソース数とは、あるプロセスがアクセスするテーブル数、インデックス数、あるいはアクセスする全テーブルの全カラムの総数が主なものです。

特に、各プロセスのメタキャッシュの総サイズが搭載メモリのサイズを超えたり、あるいは、その多くを占めることでデータベースキャッシュやOSのファイルキャッシュへのメモリの割り当てが圧迫されたりする場合には、Global Meta Cacheの使用を検討してください。Global Meta Cacheを使用すると、共有メモリ上のメタキャッシュを参照するために、1つのSQL実行にかかる時間が伸びるかもしれませんが、データベースキャッシュなどに、より多くのメモリを割り当てられることによる効果の方が大きいことが期待できます。

Global Meta Cacheを使用する性能劣化すらも許容できない場合には、あるプロセスがアクセスするテーブルの数を制限するなどの工夫が必要です。

13.1.2 Global Meta Cacheのためのメモリの見積もり

Global Meta Cache機能を有効にするためには、pgx_global_metacacheパラメータにGlobal Meta Cahche専用の共有メモリ(以降、GMC領域)のサイズの上限値を指定してください。この上限値には、“付録A パラメータ”で見積もったサイズを指定することが理想的です。この値よりも小さな値でも動作することはできますが、“13.1.3 GMC領域の使われ方”を参照してデメリットがあることを理解してください。

13.1.3 GMC領域の使われ方

起動したときには、GMC領域のためのメモリはあまり使われていませんが、新たなメタキャッシュがGMC領域に配置されていくにしたがってGMC領域が拡大していきます。もし上限サイズにまで拡大した場合には、システムがあまり使用されていないと判断したメタキャッシュを破棄して、新たなメタキャッシュをGMC領域に配置します。

そのため、GMC領域は見積もり式の算出値よりも小さくても動作しますが、破棄されたメタキャッシュを再度利用する必要が生じたときに、メタキャッシュの再生成が行われます。もし、このようなことが頻繁に発生するようであれば、全体的に性能が劣化することに注意してください。

このようなことを考慮して、例えば、時間帯によってアクセスするテーブル群が異なり、その切り替わりの直後の時間帯の劣化ならば許容できるというような使い方をすれば、問題とはならないかもしれません。

いずれにしても、システムを稼働する前に十分にテストしてチューニングしてください。

13.1.4 Global Meta Cache機能の有効化

postgresql.confファイルを編集し、pgx_global_metacaheパラメータを設定します。postgresql.confファイルの編集後は、インスタンスを再起動してください。パラメータの詳細については、“付録A パラメータ”を参照してください。

パラメータ名	説明
pgx_global_metacache	共有メモリ上のGMC領域の最大メモリ量を指定します。0を設定するとGlobal Meta Cache機能は無効です。デフォルト値は0です。 Global Meta Cache機能を有効にするときの最小値は10MBです。

キャッシュが作成される際に、共有メモリ上のメタキャッシュの総量がpgx_global_metacacheで指定された値を超える場合、参照されておらず使用頻度が低いメタキャッシュがGMC領域から削除されます。なお、すべてのGMCが使用中で、GMC領域にキャッシュを作成できない場合、キャッシュは一時的にバックエンドプロセスのローカルメモリ上に作成されます。

例

postgresql.confでの設定例を示します。

```
pgx_global_metacache = 800MB
```

待機イベント

Global Meta Cache機能により待機イベントが発生する場合があります。待機イベントは、pg_stat_activityビューのwait_event列で確認できます。下記でGMC固有の待機イベントを説明します。

[GMC機能の待機イベント]

待機イベント型	待機イベント名	説明
LWLock	GlobalCatcache	GMC領域でのメタキャッシュを検索、追加および削除するために待機しています。
IPC	GMCSweep	GMC領域不足時に削除可能なメタキャッシュを選択するために待機しています。 GMCがすべて参照中で削除可能なメタキャッシュがない場合は、参照がなくなり削除可能なメタキャッシュを選択するまで待機しています。

注意

GMCSweepが大量に発生する場合は、pgx_global_metacacheの設定値を増やしてください。

参照

pg_stat_activityビューの詳細は、“PostgreSQL Documentation”の“Viewing Statistics”を参照してください。

13.1.5 リソースの見積り

Global Meta Cache機能で使用するメモリ量の見積り式については、“導入ガイド(サーバ編)”の“Global Meta Cache機能で使用するメモリの見積り式”を参照してください。

13.2 統計情報

Global Meta Cache機能の統計情報について説明します。

13.2.1 システムビュー

システムビューpgx_stat_gmcでGMC領域のキャッシュヒット率とサイズを確認できます。列の詳細は、“C.6 pgx_stat_gmc”を参照してください。

キャッシュヒット率が低くかつ現在のメモリ使用量がpgx_global_metacacheに近い値の場合は、性能が低下している可能性があるためpgx_global_metacacheの設定値を増やしてください。

統計情報の詳細は、“8.6 データベース活動状況の監視”を参照してください。

第14章　Local Meta Cache制限機能

Local Meta Cache制限機能は、長時間アクセスしていないLocal Meta Cacheを削除することでサイズを制限する機能です。

14.1 利用方法

Local Meta Cache制限機能の利用方法について説明します。

14.1.1 Local Meta Cache制限機能を有効にするかどうかの判断

“付録A パラメータ”を参照し、カタログキャッシュ、リレーションキャッシュとして使用するメモリの総量を見積もった結果、搭載メモリの量を超える、あるいは、その多くを占める場合には、本機能の使用を検討してください。

本機能を使うと、これまでは永久に保持していたメタキャッシュを破棄するという動作が加わります。このことで、破棄したメタキャッシュを再度参照しようとしたときに、メタキャッシュの再作成が必要となるため、本機能を使用しないときに比べて性能が悪くなります。

そのため、メタキャッシュの破棄について、以下を読んで理解してください。

- 14.1.3 Local Meta Cache制限機能を有効にした際のキャッシュ削除
- 14.1.4 Local Meta Cache制限機能の性能への影響とパラメータ調整
- Local Meta Cache制限機能に関するパラメータ

これらを考慮した上限値の設定方法は、“付録A パラメータ”の見積もり式に詳しく書かれています。

14.1.2 Local Meta Cache制限機能のパラメータ設定方法

Local Meta Cache制限機能を有効にするためには、pgx_catalog_cache_max_sizeパラメータとpgx_relation_cache_max_sizeパラメータを設定します。

パラメータ名	説明
pgx_catalog_cache_max_size	バックエンドプロセスがカタログキャッシュとして使用するメモリ量の上限値を指定します。8KB以上に設定することでカタログキャッシュ削除を有効にできます。0を設定するとカタログキャッシュ削除は無効となります。デフォルト値は0です。
pgx_relation_cache_max_size	バックエンドプロセスがリレーションキャッシュとして使用するメモリ量の上限値を指定します。8KB以上に設定することでリレーションキャッシュ削除を有効にできます。0を設定するとリレーションキャッシュ削除は無効となります。デフォルト値は0です。

 例

postgresql.confでの設定例を示します。

```
pgx_catalog_cache_max_size = 1MB
pgx_relation_cache_max_size = 1MB
```

14.1.3 Local Meta Cache制限機能を有効にした際のキャッシュ削除

本機能有効時のキャッシュ戦略は、指定された上限値の範囲内で可能な限りキャッシュを保持することです。もし、新しいキャッシュを保持することで、上限値を超えるのであれば、参照の局所性を考慮して、参照されていない時間が最も長いキャッシュから順に削除します。

しかし、アクティブなトランザクションが使用しているキャッシュを削除することはできないので、1つのトランザクションで多数のキャッシュを使用する場合には、上限値を超えてキャッシュを保持することもあります。この場合には、トランザクションの終了時にすべてのキャッシュを削除します。これは、メモリを解放するために必要だからです。

PostgreSQLでは、メモリ獲得を高速に行うために、ある程度の大きさのメモリブロックをOSから獲得しておき、そのブロックから小さなメモリを切り出して使用します。メタキャッシュのためのメモリも同じように切り出します。そのため、メモリブロック全体に散りばめられたメタキャッシュをすべて破棄することで、メモリブロックをOSに返却することが可能になります。このようなことが発生すると、次のSQLの実行が、メ

タキャッシュの再作成ために遅くなります。そのため、機能の上限値には、少なくとも1つのトランザクションが使用するメタキャッシュのサイズよりも大きい値を設定しておくべきです。

メタキャッシュのサイズが上限値を超えた際は、以下のメッセージが出力されます。

```
WARNING: could not reduce Cat/RelCacheMemoryContext size to AA kilobytes, reduced to BB kilobytes
HINT: consider increasing the configuration parameter pgx_catalog/relation_cache_max_size
```

（***AA***: 上限値、***BB***: 実際に使用されているメモリ量）

CatCacheMemoryContext、RelCacheMemoryContextはそれぞれカタログキャッシュ、リレーションキャッシュを保存するメモリ領域です。このメッセージが出力されたときは、上限値を増加させることを検討してください。

上限値を増やすことで、バックエンドプロセスによるメモリ消費量が許容値を超えるならば、1トランザクションでアクセスするテーブル数を減らすなど、実行するSQLを再考することや、使用されているメモリ量に合わせてメモリを増設することで対処する必要があります。

14.1.4 Local Meta Cache制限機能の性能への影響とパラメータ調整

メタキャッシュの再生成がどのくらい行われているのかを観察することで、目標の性能を達成できない原因が、上限値の低さにあるのかを判断できます。

以下のメッセージから、キャッシュヒット率を次のように計算してください。

```
キャッシュヒット率 ＝ キャッシュにヒットした回数 ÷ キャッシュを検索した回数
```

キャッシュのヒット率が、80%以上であれば、本機能が性能を阻害する主たる要因ではないでしょう。そうでなければ、上限値を上げて、性能が目標を達成できるかどうかを試してください。その際、リレーションキャッシュに配分の重点を移すことを最初に試みてください。SQLを実行するときには、カタログキャッシュを基にして生成したリレーションキャッシュを主に参照するので、リレーションキャッシュを多く残しておいた方が有利だからです。

```
カタログキャッシュ：catalog cache hit stats: search XX, hits YY
リレーションキャッシュ：relation cache hit stats: search XX, hits YY
```

（***XX***:キャッシュを検索した回数、***YY***:キャッシュにヒットした回数）

このメッセージは、トランザクションが終了した時に出力されます。ただし、頻繁に出力するとメッセージを出力すること自身によって性能が劣化してしまうので、以下のパラメータで出力間隔を調整することができます。

パラメータ名	説明
pgx_cache_hit_log_interval	トランザクションが終了した時に、前にメッセージを出力してから本パラメータに設定した時間を経過していたならば、メッセージを出力します。 0を設定するとトランザクションが終了するたびにメッセージを出力します。-1を設定すると出力は無効となります。デフォルト値は10minです。 pgx_catalog_cache_max_size、pgx_relation_cache_max_sizeが無効に設定されている場合も、対応するキャッシュのメッセージ出力は無効となります。 サーバーに接続した直後には、ユーザー認証などのために、ユーザーアプリケーションからのリクエストの前に、小さなトランザクションが発生します。これらについてのヒット率を知ることには意味がないので、サーバーに接続してから、本パラメータに設定した時間が経過した後に開始したトランザクションの終了時にメッセージを出力します。 同じ理由で、0のような小さい値を設定すると、このような小さなトランザクションの終了に伴ってメッセージが出力されることがあります。 メッセージがどのトランザクションに対応しているかは、先頭に出力される情報から確認できます。この情報は、パラメータlog_line_prefixの設定により変わります。

例

postgresql.confでの設定例を示します。

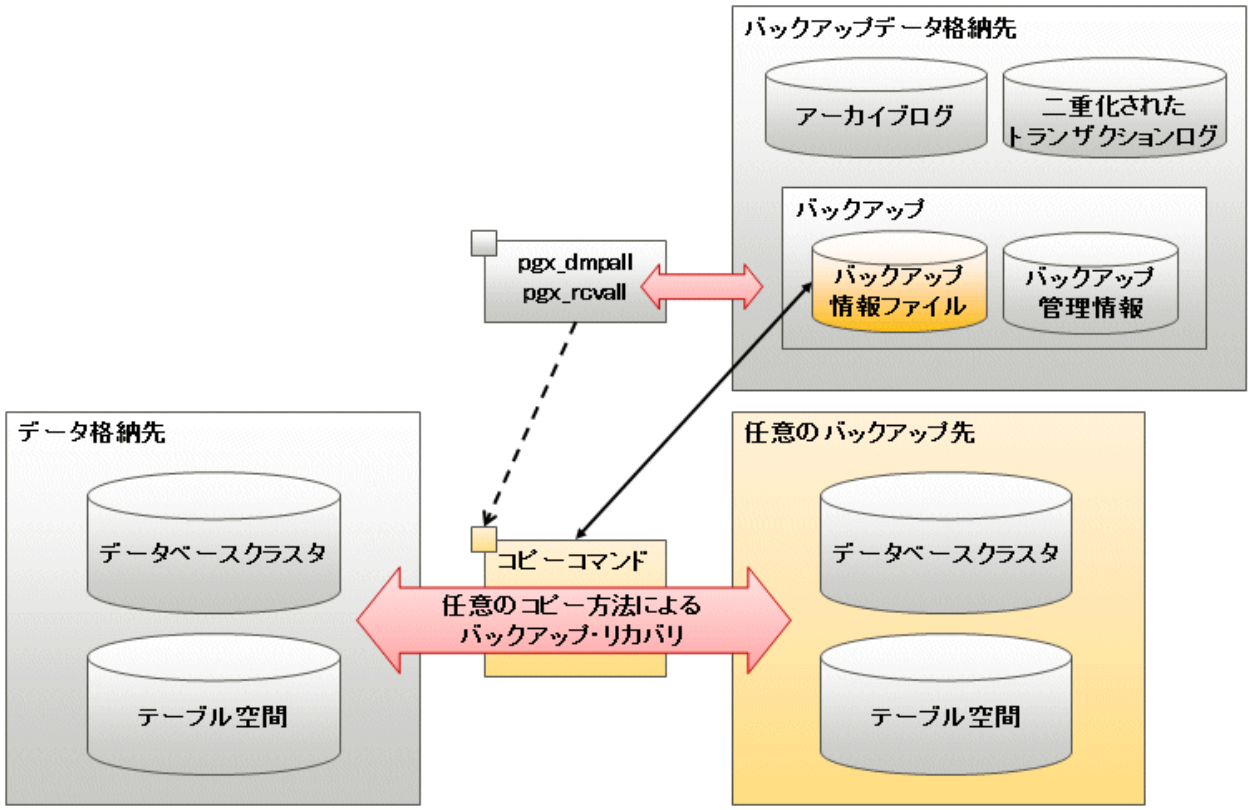
```
pgx_cache_hit_log_interval= 30min
```

第15章　コピーコマンドを使用したバックアップ/リカバリ

pgx_dmpallコマンドとpgx_rcvallコマンドでは、ユーザーが作成したコピーコマンドを使用することで、任意のバックアップ先へのバックアップおよび任意のバックアップ先からのリカバリを、任意のコピー方法で実施することができます。

コピーコマンドは、ユーザーが事前にデータベースクラスタやテーブル空間に対するコピー処理を実装した実行可能なスクリプト形式などで作成するものであり、pgx_dmpallコマンドとpgx_rcvallコマンドの実行時に呼び出されます。

コピーコマンドを使用したバックアップ、およびリカバリについて説明します。



ポイント

一部のテーブル空間のみをコピーコマンドによるバックアップ対象とすることも可能です。ただし、コピーコマンドでのバックアップ対象とならないデータベース資源は、バックアップデータ格納先のディレクトリにバックアップされます。

注意

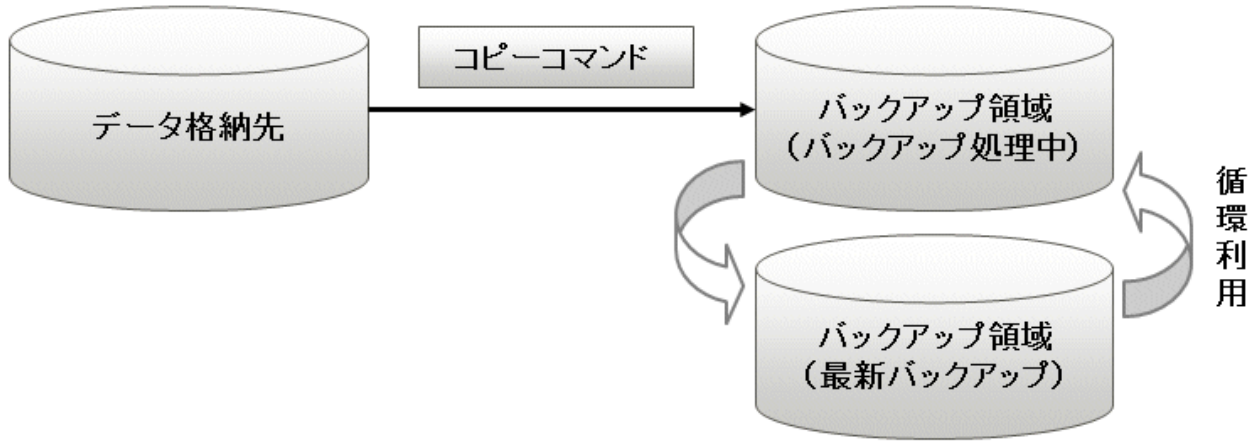
インスタンスをリカバリするためには、バックアップデータ格納先のディレクトリと任意のバックアップ先の何れも必要であり、二次媒体に退避するような場合には、これらを組み合わせて管理することが必要となります。

15.1 コピーコマンドの構成

バックアップおよびリカバリ運用を考慮したコピーコマンドの構成について説明します。

バックアップ領域の循環利用

コピーコマンドで使用するバックアップ領域は、万が一、バックアップ処理中にデータ格納先に異常が発生した場合に備えて2つ用意します。コピーコマンドでは、これらのバックアップ領域を循環利用しながら、バックアップ処理を行います。

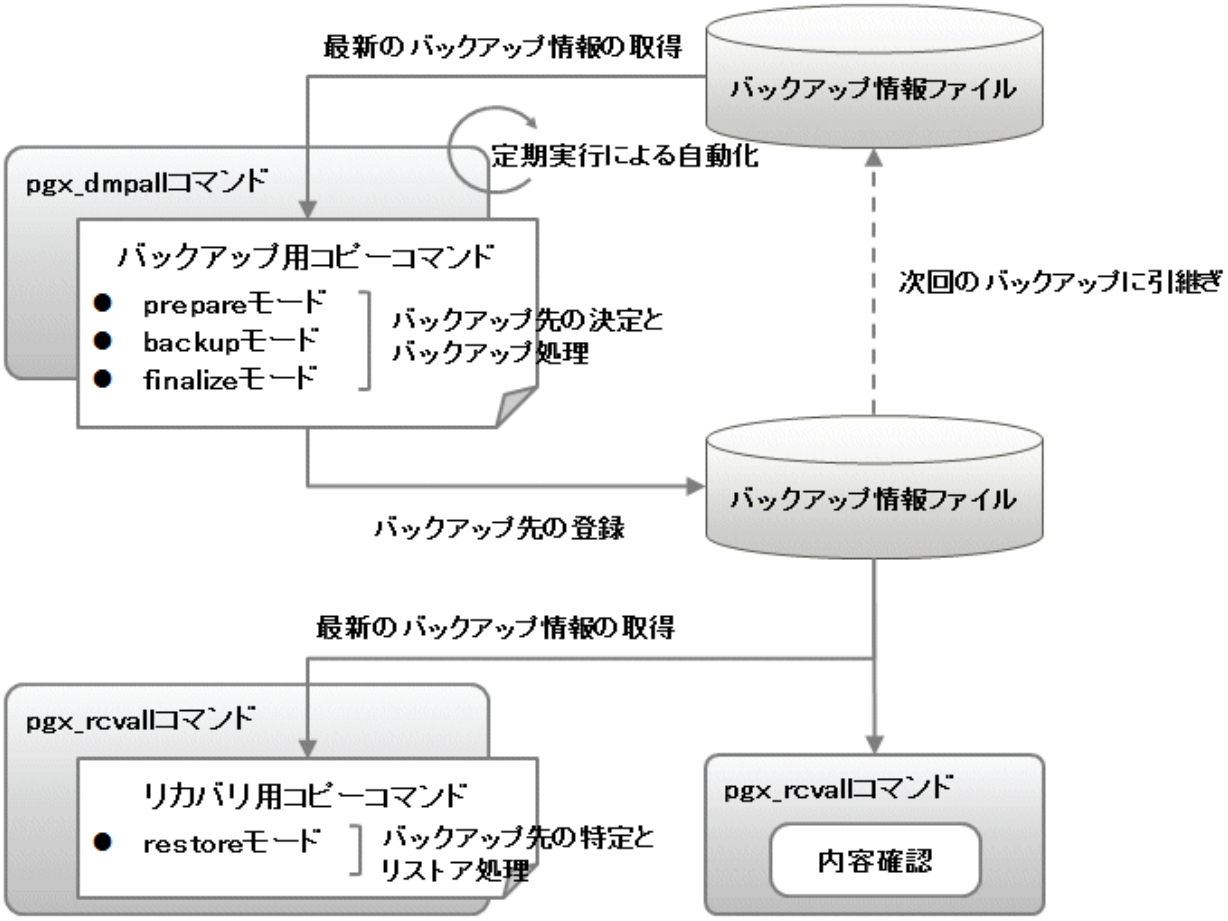


注意

コピーコマンドによるバックアップ先にバックアップデータ格納先のディレクトリを使用することはできません。

バックアップ情報ファイルを利用したバックアップ処理

コピーコマンドではバックアップ領域を循環利用する必要があるため、バックアップのたびにバックアップ先を決定しなければなりません。そのため、バックアップ先に関する任意の情報が登録可能なバックアップ情報ファイルを利用することで、バックアップ運用を自動化することができます。



参考

バックアップ情報ファイルは、pgx_dmpallコマンドがバックアップデータ格納先のディレクトリに用意したファイルであり、コピーコマンド内から自由に参照や更新が可能です。本ファイルは、pgx_dmpallコマンドが成功した最新のバックアップに関連付けて管理されるため、利用者が登録したコピーコマンドに関する最新のバックアップ情報を得ることができます。また、バックアップ情報ファイルの内容は、pgx_rcvallコマンドから表示することもできます。

バックアップ用コピーコマンドの構成

pgx_dmpallコマンドは、以下の3つモードで順番にバックアップ用コピーコマンドを呼び出します。そのため、バックアップ用コピーコマンドはそれぞれのモードで行うべき処理が実装されている必要があります。

- prepareモード

  2つのバックアップ領域のうち、今回のバックアップ対象の領域を決定します。

  前回のバックアップ時にバックアップ情報ファイルへ書き込まれた、最新のバックアップ先に関する情報を読み込むことで、今回のバックアップ対象の領域を決定できます。

- backupモード

  任意のコピー方法にて、prepareモードで決定したバックアップ領域に対してバックアップを行います。

- finalizeモード

  バックアップ情報ファイルに今回のバックアップ先に関する情報を書き込みます。

  これにより、次回のバックアップ時に、prepareモードで最新のバックアップ先を確認することができます。

注意

コピーコマンド内の各モード間のバックアップ情報の引継ぎは、一時ファイルを作成するなど、利用者の任意の方法で行ってください。

リカバリ用コピーコマンドの構成

pgx_rcvallコマンドは、以下のモードでリカバリ用コピーコマンドを呼び出します。そのため、リカバリ用コピーコマンドはそのモードで行うべき処理が実装されている必要があります。

- restoreモード

  任意のコピー方法にて、バックアップ用コピーコマンドで取得したバックアップ先からのリストア処理を実装します。

ポイント

コピーコマンドに引数として与えられるモードを参照することで、バックアップとリカバリを1つのコピーコマンドで実装することも可能です。

例

bashのスクリプト形式で作成する場合

```
case $1 in
    prepare)
        prepareモードで行う処理を実装します。
        ;;
    backup)
        backupモードで行う処理を実装します。
        ;;
    finalize)
        finalizeモードで行う処理を実装します。
        ;;
```

```
    restore)
        restoreモードで行う処理を実装します。
        ;;
esac
```

ポイント

コピーコマンドの開発を理解していただくことを目的としたサンプルとして、特定のディレクトリにデータベースクラスタとテーブル空間の配置先ディレクトリをバックアップするサンプルスクリプトを提供しています。

サンプルは以下のディレクトリに格納されています。

```
/インストールディレクトリ/share/copy_command.archive.sh.sample
```

15.2 コピーコマンドを使用したバックアップ操作

コピーコマンドを使用したバックアップは、通常のバックアップ手順に加えて、コピーコマンドを作成してpgx_dmpallコマンドに指定します。以下では、コピーコマンドを使用した場合の固有の手順について説明します。

バックアップの準備

バックアップを実施する前に、バックアップの準備を実施する必要があります。

以下の手順で実施してください。

1. バックアップ対象のデータベース資源の決定

   コピーコマンドでバックアップするデータベース資源を決定します。コピーコマンドでのバックアップは、以下のすべて、または一部を対象とすることができます。

   - データベースクラスタ
   - テーブル空間

   テーブル空間の一部をコピーコマンドでバックアップする場合は、対象のテーブル空間の名前を記述したファイルを作成します。ただし、すべてのテーブル空間をコピーコマンドでバックアップする場合は、本ファイルを作成する必要はありません。

   例

   テーブル空間tblspc1とtblspc2をコピーコマンドでバックアップする場合

   ```
   tblspc1
   tblspc2
   ```

2. バックアップ領域の用意

   手順1で決定したバックアップ対象のデータベース資源を退避するためのバックアップ領域を用意します。

3. コピーコマンドの作成

   バックアップ、およびリカバリ用のコピーコマンドを作成します。詳細は、“15.4 コピーコマンドのインタフェース”を参照してください。

バックアップの実行

pgx_dmpallコマンドの-Yオプションに、バックアップの準備の手順3で作成したバックアップ用コピーコマンドのファイルパス名を指定して、バックアップを実行します。

以下に、データベースクラスタを除く一部のテーブル空間のみをコピーコマンドでバックアップする操作例を示します。

例

```
$ pgx_dmpall -D /database/inst1 -Y '/database/command/backup.sh' --exclude-copy-cluster -P '/database/command/
tablespace_list.txt'
```

ポイント

- データベースクラスタをコピーコマンドでバックアップしない場合は、--exclude-copy-clusterオプションを指定します。
- 一部のテーブル空間をコピーコマンドでバックアップする場合は、バックアップの準備の手順1で作成したファイルのパス名を-Pオプションに指定します。

参照

- pgx_dmpallコマンドの詳細は、“リファレンス”の“pgx_dmpallを参照してください。

バックアップ状態の確認

pgx_rcvallコマンドを使用して、バックアップの状況を確認します。

pgx_rcvallコマンドに-lオプションを指定すると、バックアップデータの情報が表示されます。コピーコマンドでバックアップした場合は、コピーコマンドによるバックアップ対象資源の一覧が追加で表示されます。

例

```
$ pgx_rcvall -l -D /database/inst1
Date                Status   Dir                                Resources backed up by the copy command
2022-03-01 13:30:40 COMPLETE /backup/inst1/2022-03-01_13-30-40 pg_data,dbspace,indexspace
```

15.3 コピーコマンドを使用したリカバリ操作

コピーコマンドを使用したリカバリは、通常のリカバリの手順に加えて、コピーコマンドを作成してpgx_rcvallコマンドに指定します。以下では、コピーコマンドを使用した場合の固有の手順について説明します。

最新のバックアップ領域の確認

バックアップ情報ファイルから最新のバックアップ先を確認し、リカバリ可能な状態であることを確認します。

pgx_rcvallコマンドに--view-results-of-copyingオプションを指定して実行することで、バックアップ情報ファイルの内容が表示されます。

例

```
$ pgx_rcvall -D /database/inst1 --view-results-of-copying
```

リカバリの実行

pgx_rcvallコマンドの-Yオプションに、“15.2 コピーコマンドを使用したバックアップ操作”のバックアップの準備の手順3で作成したリカバリ用コピーコマンドのファイルパス名を指定して、リカバリを実行します。

以下に、データベースクラスタを除く一部のテーブル空間のみをコピーコマンドによりリカバリを行う操作例を示します。

例

```
$ pgx_rcvall -D /database/inst1 -B /backup/inst1 -Y '/database/command/recovery.sh'
```

ポイント

pgx_rcvallコマンドでは、最新のバックアップがコピーコマンドを使用して行われた場合、どのデータベース資源がコピーコマンドによってバックアップされたのか、またはバックアップデータ格納先のディレクトリにバックアップされたのかを自動で認識します。そのため、pgx_rcvallコマンドにはリカバリ用コピーコマンドを指定するだけで実行できます。

参照

pgx_rcvallコマンドの詳細は、“リファレンス”の“pgx_rcvall”を参照してください。

15.4 コピーコマンドのインタフェース

コピーコマンドには、以下の2種類のコマンドがあります。

- バックアップ用コピーコマンド
- リカバリ用コピーコマンド

各コピーコマンドのインタフェースについて、説明します。

15.4.1 バックアップ用コピーコマンド

機能

pgx_dmpallコマンドから呼び出されるユーザーコマンドです。

形式

pgx_dmpallコマンドがコピーコマンドを呼び出す際の引数は以下です。

操作モードがprepareの場合

```
コピーコマンド名 prepare 'バックアップ情報ファイルのパス名' 'バックアップ対象一覧ファイルのパス名'
```

操作モードがbackupの場合

```
コピーコマンド名 backup
```

操作モードがfinalizeの場合

```
コピーコマンド名 finalize 'バックアップ情報ファイルのパス名'
```

引数

- 操作モード

モード	説明
prepare	コピーコマンドでバックアップするための準備処理を実装します。 PostgreSQLのオンラインバックアップモードが開始される前に呼び出されます。
backup	バックアップの実処理を実装します。 PostgreSQLのオンラインバックアップモード中に呼び出されます。

モード	説明
finalize	バックアップの完了処理を実装します。 PostgreSQLのオンラインバックアップモードが終了した後に呼び出されます。

- バックアップ情報ファイルのパス名

  シングルクォートで囲まれた、最新のバックアップのバックアップ情報ファイルのファイルパス名です。バックアップが一度も行われていない場合は'-'となります。

- バックアップ対象一覧ファイルのパス名

  シングルクォートで囲まれた、コピーコマンドでのバックアップ対象となる資源名の一覧が出力されたファイルパス名です。資源名には、以下のいずれかが資源ごとに記述されます。

資源	記述内容
データベースクラスタ	pg_data
テーブル空間	テーブル空間の名前

例

データベースクラスタ、および名前がdbspace、indexspaceのテーブル空間をコピーコマンドのバックアップ対象とする場合、本ファイルの内容は以下のようになります。

```
pg_data
dbspace
indexspace
```

参考

pgx_dmpallコマンドがバックアップ対象一覧ファイルに出力する資源名の符号化方式は、本コマンドがclient_encoding設定パラメータにautoを指定してデータベースに接続した際の符号化方式であり、コマンド実行時のロケールに依存します。

操作モードによって、引数の数が異なります。操作モードごとの引数は以下のとおりです。

操作モード	第一引数	第二引数	第三引数
prepare	操作モード	バックアップ情報ファイルのパス名	バックアップ対象一覧ファイルのパス名
backup		なし	なし
finalize		バックアップ情報ファイルのパス名	

また、操作モードによって、バックアップ情報ファイルとバックアップ対象一覧ファイルのアクセス権限が異なります。操作モードごとのアクセス権限は以下のとおりです。

操作モード	バックアップ情報ファイル	バックアップ対象一覧ファイル
prepare	インスタンス管理者のみ参照可能	インスタンス管理者のみ参照可能
backup	－	－
finalize	インスタンス管理者のみ参照と更新が可能	－

戻り値

戻り値	意味
0	正常終了

戻り値	意味
	pgx_dmpallコマンドは処理を継続します。
0以外	異常終了 pgx_dmpallコマンドが異常終了します。

説明

- コピーコマンドはpgx_dmpallコマンドを実行したOSユーザーの権限で動作します。このため、pgx_dmpallコマンドを実行するユーザーに対して、コピーコマンドの実行権限を付与してください。また、必要に応じてコピーコマンド内でユーザーの変更を行ってください。
- バックアップ情報ファイルに情報を書き込む場合は、コピーコマンドからリダイレクションなどの方法で、任意の文字列を書き込んでください。
- コピーコマンドは各モードごと呼び出されるため、各モードの処理をすべて実装してください。
- 複数の資源を同時にコピーしたい場合は、コピーコマンド内で並列にコピーするように実装してください。

注意

- バックアップ情報ファイルやバックアップ対象一覧ファイルは削除できません。また、権限も変更できません。
- コピーコマンドの標準出力や標準エラーは、pgx_dmpallコマンドを実行した端末に出力されます。
- コピーコマンドが無応答になった場合、pgx_dmpallコマンドも無応答となります。OSのコマンドでコピーコマンドの状態を確認し、無応答であると判断した場合は、OSのコマンドでコピーコマンドを強制停止してください。
- コピーコマンドの処理内容や処理結果を一時ファイルなどに取得し、コピーコマンドが異常終了した場合に、後から原因が調査できるように実装してください。
- prepareモードの場合のみ、PostgreSQLクライアントアプリケーションを利用してコピーコマンドからデータベースへアクセスすることが可能です。それ以外のモードの場合は、コピーコマンド内からFujitsu Enterprise Postgresのコマンドや PostgreSQLアプリケーションを実行しないでください。
- バックアップ開始時に共有メモリバッファ上のデータがディスクに書き出されている必要があるため、postgresql.confファイルのfsyncパラメータを有効にしてください。

15.4.2 リカバリ用コピーコマンド

機能

pgx_rcvallコマンドから呼び出されるユーザーコマンドです。

形式

pgx_rcvallコマンドがコピーコマンドを呼び出す際の引数は以下です。

```
コピーコマンド名 restore 'バックアップ情報ファイルのパス名' 'バックアップ対象一覧ファイルのパス名'
```

引数

- 操作モード

モード	説明
restore	リストア処理を実装します。

- バックアップ情報ファイルのパス名

  シングルクォートで囲まれた、バックアップ情報ファイルのファイルパス名です。

- バックアップ対象一覧ファイルのパス名

  シングルクォートで囲まれた、コピーコマンドでのリカバリ対象となる資源名の一覧が記述されたファイルパス名です。

バックアップ情報ファイルとバックアップ対象一覧ファイルのアクセス権限は以下のとおりです。

バックアップ情報ファイル	バックアップ対象一覧ファイル
インスタンス管理者のみ参照可能	インスタンス管理者のみ参照可能

戻り値

戻り値	意味
0	正常終了 pgx_rcvallコマンドは処理を継続します。
0以外	異常終了 pgx_rcvallコマンドが異常終了します。

説明

- コピーコマンドはpgx_rcvallコマンドを実行したOSユーザーの権限で動作します。このため、pgx_rcvallコマンドを実行するユーザーに対して、コピーコマンドの実行権限を付与してください。また、必要に応じてコピーコマンド内でユーザーの変更を行ってください。
- コピーコマンドはrestoreモードで1度だけ呼び出されます。
- 複数の資源を同時にコピーしたい場合は、コピーコマンド内で並列にコピーするように実装してください。

注意

- バックアップ情報ファイルやバックアップ対象一覧ファイルは削除できません。また、権限も変更できません。
- コピーコマンドの標準出力や標準エラーは、pgx_rcvallコマンドを実行した端末に出力されます。
- コピーコマンドが無応答になった場合、pgx_rcvallコマンドも無応答となります。OSのコマンドでコピーコマンドの状態を確認し、無応答であると判断した場合は、OSのコマンドでコピーコマンドを強制停止してください。
- コピーコマンドの処理内容や処理結果を一時ファイルなどに取得し、コピーコマンドが異常終了した場合に、後から原因が調査できるように実装してください。
- コピーコマンド内ではFujitsu Enterprise Postgresのコマンドや PostgreSQLアプリケーションを実行しないでください。
- データベースクラスタ配下のpostmaster.pid、pg_wal/サブディレクトリおよびpg_replslotなど、アーカイブログによるリカバリに不要なファイルやディレクトリがバックアップに含まれている場合があります。これらの不要なファイルやディレクトリがある場合は、リストア後にコピーコマンド内で削除してください。

第16章　異常時の対処

Fujitsu Enterprise Postgresの運用中における、データベースやアプリケーションに異常が発生した場合の対処方法について説明します。

異常の種類に応じて、データベースクラスタをリカバリする必要があります。リカバリは、以下の資源をリカバリします。

- データ格納先
- トランザクションログ格納先(データ格納先と別のディスクに格納している場合)
- バックアップデータ格納先

注意

実際にはディスクが故障していなくても、故障している場合と同じ入出力異常のメッセージが出力されることがあります。これらは、それぞれリカバリの方法が異なります。

ハードウェアの状態を確認して、以下のいずれかを選択してください。

- ディスクが故障している場合

  “16.1 ディスク障害(ハードウェア)からのリカバリ”を参照して対処してください。

- ディスクが故障していない場合

  “16.14 ディスク障害以外の入出力異常”を参照して対処してください。

  例えば、以下のような場合があります。

  ― 外部ディスクとの間のネットワーク異常

  ― 電源未投入やマウントによる異常

異常原因の特定方法

異常が発生した場合は、WebAdminのメッセージやシステムログおよびサーバログを参照して異常の原因を特定してください。

参照

サーバログの詳細は、“導入ガイド(サーバ編)”の“設定パラメータ”を参照してください。

リカバリの目安時間

各ディレクトリ配下にある資源のリカバリ目安時間の導出式を示します。

pgx_rcvallコマンドでコピーコマンドを使用する場合、リカバリ時間はコピーコマンドの実装内容に依存します。

- データ格納先、またはトランザクションログ格納先

  リカバリ時間 =（データ格納先の使用量 ＋ トランザクションログ格納先の使用量）÷ ディスク書込み性能 × 1.5

  ― データ格納先の使用量:データベースクラスタのディスク使用量

  ― トランザクションログ格納先の使用量:トランザクションログをデータベースクラスタの外に格納している場合のトランザクションログが消費しているディスク使用量

  ― ディスク書込み性能:運用を行うシステム環境における、1秒間あたりに書き込み可能な最大のデータ量(バイト/秒)の実測値

  ― 1.5:最も時間のかかるディスク書き込み以外の時間を見込んだ係数

- バックアップデータ格納先

リカバリ時間 = バックアップデータ格納先の使用量 ÷ ディスク書込み性能 × 1.5

  - バックアップデータ格納先の使用量:バックアップデータのディスク使用量
  - ディスク書込み性能:運用を行うシステム環境における、1秒間あたりに書き込み可能な最大のデータ量(バイト/秒)の実測値
  - 1.5:最も時間のかかるディスク書き込み以外の時間を見込んだ係数

16.1 ディスク障害(ハードウェア)からのリカバリ

データ格納ディスク、またはバックアップデータ格納ディスクでハードウェア障害が発生した場合に、障害が発生する直前の状態にデータベースクラスタをリカバリする方法について説明します。

以下の2つの方法でリカバリできます。

- 16.1.1 WebAdminを使用する場合
- 16.1.2 サーバコマンドを使用する場合

ポイント

データベースクラスタをリカバリした後、バックアップすることを推奨します。バックアップすることで、不要になったアーカイブログ(バックアップデータ格納先にコピーされたトランザクションログ)が削除されるため、ディスク容量の確保やリカバリ時間の短縮につながります。

16.1.1 WebAdminを使用する場合

障害が発生したディスクに応じて、以下のリカバリ手順でデータベースクラスタをリカバリしてください。

注意

スタンバイモードのストリーミングレプリケーションクラスタの一部であるインスタンスは、リカバリできません。

スタンバイインスタンスでディスク障害が起こった場合、インスタンスを消去して再作成することが必要となる場合があります。

“マスタモード”のストリーミングレプリケーションクラスタの一部であるインスタンスのリカバリは可能です。マスタインスタンスをリカバリすると、レプリケーションクラスタが機能しなくなり、マスタインスタンスとそのすべてのスタンバイインスタンス間でストリーミングレプリケーションが停止します。この場合、スタンバイインスタンスをスタンドアロンインスタンスの昇格するか、消去して再作成することができます。

データ格納ディスク、またはトランザクションログ格納ディスクに障害が発生した場合

以下の手順でデータ格納ディスク、またはトランザクションログ格納ディスクをリカバリしてください。

1. アプリケーションの停止

   データベースを利用しているアプリケーションを停止してください。

2. インスタンスの停止

   インスタンスを停止します。停止方法については、“2.1.1 WebAdminを使用する場合”を参照してください。インスタンスを停止せずにデータベースクラスタのリカバリをした場合は、WebAdminが自動的にインスタンスを停止します。

3. 障害ディスクのリカバリ

   ディスク交換を行ったあと、ボリュームの構成情報をリカバリします。

4. テーブル空間のディレクトリ作成

   バックアップを実施した以降にテーブル空間を定義している場合、そのディレクトリを作成します。

5. キーストアのリカバリとキーストアの自動オープンの有効化

   データベースに格納するデータを暗号化している場合は、以下を行ってください。

   ー データベースのバックアップを取得した時点のキーストアをリストアしてください。

   ー キーストアの自動オープンを有効にしてください。

6. データベースクラスタのリカバリ

   WebAdminにログインし、[インスタンス]タブで、画面の右下にあるエラーメッセージの[対処]をクリックします。

7. リカバリの実行

   [インスタンスのリストア]ダイアログが表示されますので、[はい]ボタンをクリックしてください。

   インスタンスのリストアが実行されます。リカバリが正常に完了すると、インスタンスが自動的に起動されます。

8. アプリケーションの再開

   データベースを利用しているアプリケーションを再開してください。

ポイント

ディスク異常の発生の仕方によってはWebAdminで異常が検出できない場合があります。

その場合は“16.10.3 その他の異常”を参照してリカバリしてください。

バックアップデータ格納ディスクに障害が発生した場合

以下の手順でバックアップデータ格納ディスクをリカバリしてください。

1. 障害ディスクのリカバリ

   ディスク交換を行ったあと、ボリュームの構成情報をリカバリします。

2. バックアップデータのリカバリ

   WebAdminにログインし、[インスタンス]タブの画面右下にあるエラーメッセージの[対処]をクリックします。

3. バックアップの実行

   バックアップデータをリカバリするために、バックアップを実行します。[バックアップ]ダイアログが表示されますので、[はい]ボタンをクリックしてください。バックアップが実行されます。バックアップを実行すると、インスタンスが自動的に起動されます。

ポイント

[状態を再確認する]をクリックした場合、データ格納先およびバックアップデータ格納先の資源を再度検証します。その結果により、以下のようになります。

- 異常を検出しなかった場合

  データ格納先およびバックアップデータ格納先の状態が「正常」に戻り、通常の操作を行えるようになります。

- 異常を検出した場合

  メッセージリストに再度エラーメッセージが表示されます。[対処]をクリックし、ダイアログに記されているエラー原因の解決方法にしたがって問題を解決してください。

16.1.2 サーバコマンドを使用する場合

障害が発生したディスクに応じて、以下のリカバリ手順でデータベースクラスタをリカバリしてください。

データ格納ディスク、またはトランザクションログ格納ディスクに障害が発生した場合

以下の手順でデータ格納ディスク、またはトランザクションログ格納ディスクをリカバリしてください。

1. アプリケーションの停止

   データベースを利用しているアプリケーションを停止してください。

2. インスタンスの停止

   インスタンスを停止します。停止方法については、“2.1.2 サーバコマンドを使用する場合”を参照してください。

   インスタンスの停止に失敗した場合は、“16.11 インスタンス停止失敗時の対処”を参照してください。

3. 障害ディスクのリカバリ

   ディスク交換を行ったあと、ボリュームの構成情報をリカバリします。

4. 格納先のディレクトリの作成

   ー データ格納ディスクに障害が発生した場合
   データ格納先のディレクトリを作成します。テーブル空間を定義していた場合、そのディレクトリも作成します。

   ー トランザクションログ格納ディスクに障害が発生した場合
   トランザクションログ格納先のディレクトリを作成します。

   例

   データ格納先のディレクトリを作成する場合

   ```
   $ mkdir /database/inst1
   $ chown fsepuser:fsepuser /database/inst1
   $ chmod 700 /database/inst1
   ```

参照

格納先のディレクトリの作成については、“導入ガイド(サーバ編)”の“セットアップ”の“資源配置用のディレクトリの準備”を参照してください。

5. キーストアのリカバリとキーストアの自動オープンの有効化

   データベースに格納するデータを暗号化している場合は、データベースのバックアップを取得した時点のキーストアをリストアしてください。キーストアの自動オープンは、必要に応じて設定してください。

6. データベースクラスタのリカバリ

   バックアップデータを使用して、データベースクラスタをリカバリします。

   pgx_rcvallコマンドには、以下を指定します。

   ー -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

   ー -Bオプションは、バックアップデータ格納先のディレクトリを指定します。

   例

```
> pgx_rcvall -D /database/inst1 -B /backup/inst1
```

注意

リカバリに失敗した場合は、表示されるエラーメッセージに従ってエラーの原因を取り除き、pgx_rcvallコマンドを再実行してください。

「pgx_rcvall:リカバリ中にエラーが発生しました」のメッセージがある場合は、その後にリカバリ実行時のログが出力されます。ログの最後の十数行以内にエラーの原因が出力されますので、そのメッセージに従ってエラーの原因を取り除き、pgx_rcvallコマンドを再実行してください。

また、リカバリ中に表示される以下のメッセージは、pgx_rcvallコマンドの正常な動作によって出力されるメッセージのため、ユーザーが意識する必要はありません。

```
FATAL:  データベースシステムは起動処理中です
```

7. インスタンスの起動

   インスタンスを起動します。インスタンスの起動方法については、“2.1.2 サーバコマンドを使用する場合”を参照してください。

8. アプリケーションの再開

   データベースを利用しているアプリケーションを再開してください。

バックアップデータ格納ディスクに障害が発生した場合

バックアップデータ格納ディスクをリカバリする手順を説明します。

対処方法には以下の2つの方法があります。

- インスタンスを起動したままリカバリする方法
- インスタンスを停止してリカバリする方法

インスタンス停止の有無による、実行する手順の違いを以下に示します。

No	手順	インスタンス停止の有無	
		なし	あり
1	トランザクションログ二重化の停止確認	○	ー

No	手順	インスタンス停止の有無	
		なし	あり
2	アーカイブログ出力の停止	○	－
3	アプリケーションの停止	－	○
4	インスタンスの停止	－	○
5	障害ディスクのリカバリ	○	○
6	バックアップデータ格納先のディレクトリ作成	○	○
7	アーカイブログ出力の再開	○	－
8	トランザクションログ二重化の再開	○	－
9	インスタンスの起動	－	○
10	バックアップの実行	○	○
11	アプリケーションの再開	－	○

○:実施
－:実施不要

手順を以下に示します。

インスタンスを起動したままリカバリする方法

1. トランザクションログ二重化の停止確認

   トランザクションログの二重化が停止されているかを、次のSQL関数で確認します。

```
 postgres=# SELECT pgx_is_wal_multiplexing_paused();
 pgx_is_wal_multiplexing_paused
--------------------
 t
(1 row)
```

   トランザクションログの二重化が停止していない場合は、次のSQL関数で停止してください。

```
postgres=# SELECT pgx_pause_wal_multiplexing();
LOG:  multiplexing of transaction log files has been stopped
pgx_pause_wal_multiplexing
----------------------------

(1 row)
```

2. アーカイブログ出力の停止

   バックアップ格納ディスクを交換するまでに時間がかかると、トランザクションログが蓄積されていきます。そして、データ格納ディスクまたはトランザクションログ格納ディスクが満杯になると、運用が継続できなくなる危険性があります。

   これを回避するために、以下の方法でアーカイブログの出力を停止してください。

   - archive_commandの変更

     アーカイブログを出力したものとみなすよう、例えば“echo skipped archiving WAL file %f”や“/bin/true”などの、必ず正常終了するコマンドを指定します。

     echoを指定すると、サーバログにメッセージが出力されるため、調査を行う上での指標になる可能性があります。

- 設定ファイルの再読込み

  pg_ctl reloadコマンド、またはSQL関数pg_reload_confを実行します。

なお、運用が継続できなくなる危険性が無く、単にエラーを出力させたくない場合は、archive_commandに空文字列('')を指定して設定ファイルの再読み込みを実施してください。

3. 障害ディスクのリカバリ

   ディスク交換を行ったあと、ボリュームの構成情報をリカバリします。

4. バックアップデータ格納先のディレクトリ作成

   バックアップデータ格納先のディレクトリを作成します。

   例

```
# mkdir /backup/inst1
# chown fsepuser:fsepuser /backup/inst1
# chmod 700 /backup/inst1
```

   バックアップデータ格納先のディレクトリの作成については、“3.2.2 サーバコマンドを使用する場合”を参照してください。

5. アーカイブログ出力の再開

   archive_commandの設定を元に戻し、設定ファイルを再読込みします。

6. トランザクションログ二重化の再開

   SQL関数pgx_resume_wal_multiplexingを実行します。

   例

```
SELECT pgx_resume_wal_multiplexing()
```

7. バックアップの実行

   pgx_dmpallコマンドを使用して、データベースクラスタをバックアップします。

   pgx_dmpallコマンドには、以下を指定します。

   - -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

   例

```
> pgx_dmpall -D /database/inst1
```

インスタンスを停止してリカバリする場合

1. アプリケーションの停止

   データベースを利用しているアプリケーションを停止してください。

2. インスタンスの停止

   インスタンスを停止します。停止方法については、“2.1.2 サーバコマンドを使用する場合”を参照してください。

   インスタンスの停止に失敗した場合は、“16.11 インスタンス停止失敗時の対処”を参照してください。

3. 障害ディスクのリカバリ

   ディスク交換を行ったあと、ボリュームの構成情報をリカバリします。

4. バックアップデータ格納先のディレクトリ作成

   バックアップデータ格納先のディレクトリを作成します。

   例

   ```
   # mkdir /backup/inst1
   # chown fsepuser:fsepuser /backup/inst1
   # chmod 700 /backup/inst1
   ```

   詳細については、“3.2.2 サーバコマンドを使用する場合”を参照してください。

5. インスタンスの起動

   インスタンスを起動します。起動方法については、“2.1.2 サーバコマンドを使用する場合”を参照してください。

6. バックアップの実行

   pgx_dmpallコマンドを使用して、データベースクラスタをバックアップします。

   pgx_dmpallコマンドには、以下を指定します。

   - -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

   例

   ```
   > pgx_dmpall -D /database/inst1
   ```

7. アプリケーションの再開

   データベースを利用しているアプリケーションを再開してください。

参照

- pgx_rcvallコマンド、およびpgx_dmpallコマンドの詳細は、“リファレンス”の“pgx_rcvall”および“pgx_dmpall”を参照してください。
- archive_modeの詳細は、“PostgreSQL Documentation”の“Server Administration”の“Write Ahead Log”を参照してください。
- pgx_resume_wal_multiplexingの詳細は、“B.1 WAL二重化制御関数”を参照してください。

16.2 データ破壊からのリカバリ

ディスク内のデータが論理的に破壊され、データベースが正常に動作しない場合、バックアップを取得した時点の状態にデータベースクラスタをリカバリできます。

以下の2つの方法でリカバリできます。

- 16.2.1 WebAdminを使用する場合
- 16.2.2 pgx_rcvallコマンドを使用する場合

注意

- データベースクラスタをリカバリした後、バックアップすることを推奨します。バックアップすることで、不要になったアーカイブログ(バックアップデータ格納先にコピーされたトランザクションログ)が削除されるため、ディスク容量の確保やリカバリ時間の短縮につながります。
- 過去の時点に復旧した場合、その復旧時点を起点とする新たな時系列(データベース更新の歴史)が始まります。リカバリが完了したときには、その復旧時点が新たな時系列における最新地点です。以後、最新状態にリカバリする場合には、この新たな時系列上のデータベース更新が再実行されます。

16.2.1 WebAdminを使用する場合

WebAdminを使用する場合、バックアップデータを利用してデータ破壊直前にリカバリしてください。

詳細は、“16.1.1 WebAdminを使用する場合”を参照してください。

16.2.2 pgx_rcvallコマンドを使用する場合

pgx_rcvallコマンドでバックアップ取得日時を指定してデータベースクラスタをリカバリします。そのあと、必要に応じてトランザクションを再実行し、データを復旧します。

以下の手順でデータ格納ディスクをリカバリしてください。

1. アプリケーションの停止

   データベースを利用しているアプリケーションを停止してください。

2. インスタンスの停止

   インスタンスを停止します。停止方法については、“2.1.2 サーバコマンドを使用する場合”を参照してください。

   インスタンスの停止に失敗した場合は、“16.11 インスタンス停止失敗時の対処”を参照してください。

3. バックアップ取得日時の確認

   バックアップデータ格納先に取得されているバックアップデータをpgx_rcvallコマンドで確認し、データが破壊される前の日時を特定します。

   pgx_rcvallコマンドには、以下を指定します。

   - -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。
   - -Bオプションは、バックアップデータ格納先のディレクトリを指定します。
   - -lオプションは、バックアップデータの情報を表示します。

   例

   ```
   > pgx_rcvall -D /database/inst1 -B /backup/inst1 -l
   Date                 Status          Dir
   2022-03-20 10:00:00  COMPLETE        /backup/inst1/2022-03-20_10-00-00
   ```

4. キーストアのリカバリとキーストアの自動オープンの有効化

   データベースに格納するデータを暗号化している場合は、データベースのバックアップを取得した時点のキーストアをリストアしてください。キーストアの自動オープンは、必要に応じて設定してください。

5. データベースクラスタのリカバリ

   pgx_rcvallコマンドを使用して、データベースクラスタを復旧します。

   pgx_rcvallコマンドには、以下を指定します。

   ー -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

   ー -Bオプションは、バックアップデータ格納先のディレクトリを指定します。

   ー -eオプションは、復旧する日時を指定します。時間は秒単位で指定します。

   例

   復旧時点に‘2022年3月20日 10時00分00秒’を指定した場合の実行例を示します。

```
> pgx_rcvall -D /database/inst1 -B /backup/inst1 -e "2022-03-20 10:00:00"
```

注意

リカバリに失敗した場合は、表示されるエラーメッセージに従ってエラーの原因を取り除き、pgx_rcvallコマンドを再実行してください。

「pgx_rcvall:リカバリ中にエラーが発生しました」のメッセージがある場合は、その後にリカバリ実行時のログが出力されます。ログの最後の十数行以内にエラーの原因が出力されますので、そのメッセージに従ってエラーの原因を取り除き、pgx_rcvallコマンドを再実行してください。

また、リカバリ中に表示される以下のメッセージは、pgx_rcvallコマンドの正常な動作によって出力されるメッセージのため、ユーザーが意識する必要はありません。

```
FATAL:  データベースシステムは起動処理中です
```

6. インスタンスの起動

   インスタンスを起動します。インスタンスの起動方法については、“2.1.2 サーバコマンドを使用する場合”を参照してください。

   必要に応じて、指定した復旧時点からトランザクション処理を再度実行した上で、データベースの運用を再開します。

7. アプリケーションの再開

   データベースを利用しているアプリケーションを再開してください。

参照

pgx_rcvallコマンドの詳細は、“リファレンス”の“pgx_rcvall”を参照してください。

16.3 ユーザーの誤操作からのリカバリ

ユーザーの誤操作によりデータ破壊が発生した場合に、データベースクラスタをリカバリする方法について説明します。

以下の2つの方法でリカバリできます。

- 16.3.1 WebAdminを使用する場合
- 16.3.2 pgx_rcvallコマンドを使用する場合

注意

- データベースクラスタをリカバリした後、バックアップすることを推奨します。バックアップすることで、不要になったアーカイブログ(バックアップデータ格納先にコピーされたトランザクションログ)が削除されるため、ディスク容量の確保やリカバリ時間の短縮につながります。
- 過去の時点に復旧した場合、その復旧時点を起点とする新たな時系列(データベース更新の歴史)が始まります。リカバリが完了したときには、その復旧時点が新たな時系列における最新地点です。以後、最新状態にリカバリする場合には、この新たな時系列上のデータベース更新が再実行されます。
- 有効なリストアポイントは、バックアップを取得した時系列上で作成したものです。つまり、過去の時点に復旧した場合、以降に設定したリストアポイントは利用できません。したがって、望みの過去データを復元できたら、バックアップを取得してください。

16.3.1 WebAdminを使用する場合

WebAdminを使用してバックアップ時点にリカバリすることができます。

注意

スタンバイモードのストリーミングレプリケーションクラスタの一部であるインスタンスは、リカバリできません。

スタンバイインスタンスでユーザーによる誤操作があった場合、インスタンスを消去して再作成することが必要となる場合があります。

“マスタモード”のストリーミングレプリケーションクラスタの一部であるインスタンスのリカバリは可能です。マスタインスタンスをリカバリすると、レプリケーションクラスタが機能しなくなり、マスタインスタンスとそのすべてのスタンバイインスタンス間でストリーミングレプリケーションが停止します。この場合、スタンバイインスタンスをスタンドアロンインスタンスに昇格するか、消去して再作成することができます。

以下の手順でデータ格納ディスク内のデータをリカバリしてください。

1. アプリケーションの停止

   データベースを利用しているアプリケーションを停止してください。

2. インスタンスの停止

   インスタンスを停止します。停止方法については、“2.1.1 WebAdminを使用する場合”を参照してください。

3. キーストアのリカバリとキーストアの自動オープンの有効化

   データベースに格納するデータを暗号化している場合は、以下を行ってください。

   ー データベースのバックアップを取得した時点のキーストアをリストアしてください。

   ー キーストアの自動オープンを有効にしてください。

4. データベースクラスタのリカバリ

   WebAdminにログインし、[インスタンス]タブでリカバリするインスタンスを選択して  をクリックします。

5. バックアップ時点へのリカバリの実行

   [インスタンスのリストア]ダイアログボックスで、[はい]をクリックしてください。

   リカバリが実行されます。リカバリが正常に完了すると、インスタンスが自動的に起動されます。

6. データベースの運用の再開

   必要に応じて、バックアップ時点から誤操作を行うまでのトランザクション処理を再度実行したうえで、データベースの運用を再開します。

16.3.2 pgx_rcvallコマンドを使用する場合

pgx_rcvallコマンドでは、サーバコマンドで作成したリストアポイントの時点にデータベースクラスタをリカバリします。リストアポイントの作成方法については、“3.2.2 サーバコマンドを使用する場合”の“リストアポイントの設定”を参照してください。

以下の手順でデータ格納ディスク内のデータをリカバリしてください。

1. アプリケーションの停止

   データベースを利用しているアプリケーションを停止してください。

2. インスタンスの停止

   インスタンスを停止します。停止方法については、“2.1.2 サーバコマンドを使用する場合”を参照してください。

   インスタンスの停止に失敗した場合は、“16.11 インスタンス停止失敗時の対処”を参照してください。

3. リストアポイントの確認

   バックアップデータ格納先に取得されているバックアップデータをpgx_rcvallコマンドで確認し、“3.2.2 サーバコマンドを使用する場合”で任意のファイルなどに記録したリストアポイントから、誤操作が行われる前のリストアポイントを特定します。

   pgx_rcvallコマンドには、以下を指定します。

   - -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。
   - -Bオプションは、バックアップデータ格納先のディレクトリを指定します。
   - -lオプションは、バックアップデータの情報を表示します。

   例

```
> pgx_rcvall -D /database/inst1 -B /backup/inst1 -l
Date                    Status          Dir
2022-03-01 10:00:00     COMPLETE        /backup/inst1/2022-03-01_10-00-00
```

4. キーストアのリカバリとキーストアの自動オープンの有効化

   データベースに格納するデータを暗号化している場合は、データベースのバックアップを取得した時点のキーストアをリストアしてください。キーストアの自動オープンは、必要に応じて設定してください。

5. データベースクラスタのリカバリ

   pgx_rcvallコマンドを使用して、データベースクラスタを復旧します。

   pgx_rcvallコマンドには、以下を指定します。

   - -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。
   - -Bオプションは、バックアップデータ格納先のディレクトリを指定します。
   - -nオプションは、指定したリストアポイントの時点にデータをリカバリします。

例

リストアポイントが"batch_20220303_1"の場合の実行例を示します。

```
> pgx_rcvall -D /database/inst1 -B /backup/inst1 -n batch_20220303_1
```

注意

リカバリに失敗した場合は、表示されるエラーメッセージに従ってエラーの原因を取り除き、pgx_rcvallコマンドを再実行してください。

「pgx_rcvall:リカバリ中にエラーが発生しました」のメッセージがある場合は、その後にリカバリ実行時のログが出力されます。ログの最後の十数行以内にエラーの原因が出力されますので、そのメッセージに従ってエラーの原因を取り除き、pgx_rcvallコマンドを再実行してください。

また、リカバリ中に表示される以下のメッセージは、pgx_rcvallコマンドの正常な動作によって出力されるメッセージのため、ユーザーが意識する必要はありません。

```
FATAL:  データベースシステムは起動処理中です
```

6. インスタンスの起動

   インスタンスを起動します。

   インスタンスの起動方法については、“2.1.2 サーバコマンドを使用する場合”を参照してください。

7. データベースの運用の再開

   必要に応じて、指定した復旧時点から誤った操作を行うまでのトランザクション処理を再度実行した上で、データベースの運用を再開します。

参照

pgx_rcvallコマンドの詳細は、“リファレンス”の“pgx_rcvall”を参照してください。

16.4 アプリケーション異常の対処

長時間、待ち状態のクライアントのコネクションが存在する場合、原因となるコネクションを切断することで、データベースの性能劣化を抑止できます。

以下の方法で切断するコネクションを特定できます。

- ビュー(pg_stat_activity) (“16.4.1 ビュー(pg_stat_activity)を使用する場合”参照)
- psコマンド (“16.4.2 psコマンドを使用する場合”参照)

コネクションの切断は、システム管理関数(pg_terminate_backend)を使用します。

16.4.1 ビュー(pg_stat_activity)を使用する場合

ビュー(pg_stat_activity)を使用する場合、以下の手順でコネクションを切断します。

1. psqlコマンドで、データベースpostgresに接続します。

```
> psql postgres
psql (<x>) (注1)
Type "help" for help.
```

注1: <*x*>には、本製品がベースとするPostgreSQL のバージョンが表示されます。

2. 長時間、待ち状態のクライアントのコネクションを切断します。

pg_terminate_backend()を利用して、長時間接続中のコネクションを切断します。

ただし、アプリケーションの互換性を維持することを考慮し、下記のSQL文中のシステムカタログや関数を直接参照したり使用しないようにしてください。

例

以下の例は、クライアントの待ち状態が、60分以上の場合、コネクションを切断します。

```
select pid,usename,application_name,client_addr,pg_terminate_backend(pid) from pg_stat_activity where backend_type
= 'client backend' and state='idle in transaction' and current_timestamp > cast(query_start + interval '60 minutes'
as timestamp);
-[ RECORD 1 ]--------+---------------
pid                  | 4684
usename              | fsepuser
application_name     | apl1
client_addr          | 192.11.11.1
pg_terminate_backend | t
```

参照

- pg_terminate_backendの詳細は、“PostgreSQL Documentation”の“The SQL Language”の“System Administration Functions”を参照してください。
- アプリケーションの互換性の維持に関する詳細は、“アプリケーション開発ガイド”の“アプリケーションの互換に関する注意事項”を参照してください。

16.4.2 psコマンドを使用する場合

標準的なUnixツール(psコマンド)を使用する場合、以下の手順でコネクションを切断します。

1. psコマンドを実行します。
   なお、“<*x*>”は、製品のバージョンを示します。

```
> ps axwfo user,pid,ppid,tty,command | grep postgres
fsepuser    19174 18027 pts/1                ¥_ grep postgres
fsepuser    20517     1 ?        /opt/fsepv<x>server64/bin/postgres -D /disk01/data
fsepuser    20518 20517 ?         ¥_ postgres: logger
fsepuser    20520 20517 ?         ¥_ postgres: checkpointer
fsepuser    20521 20517 ?         ¥_ postgres: background writer
fsepuser    20522 20517 ?         ¥_ postgres: walwriter
fsepuser    20523 20517 ?         ¥_ postgres: autovacuum launcher
fsepuser    20524 20517 ?         ¥_ postgres: archiver
fsepuser    20525 20517 ?         ¥_ postgres: logical replication launcher
fsepuser    18673 20517 ?         ¥_ postgres: fsepuser postgres 192.168.100.1(49448) idle
```

```
fsepuser    16643 20517 ?        ¥_ postgres: fsepuser db01 192.168.100.11(49449) UPDATE waiting
fsepuser    16644 20517 ?        ¥_ postgres: fsepuser db01 192.168.100.12(49450) idle in transaction
```

プロセスID(16643)は、UPDATE文で長時間接続したコネクションや、資源を占有したコネクション(waiting)の可能性があります。

2. 長時間、待ち状態のクライアントのコネクションを切断します。

   pg_terminate_backend()を利用して、上記1.で特定したプロセスIDのコネクションを切断します。

   プロセスIDが16643のコネクションを切断する例を以下に示します。

   ただし、アプリケーションの互換性を維持することを考慮し、下記のSQL文中の関数を使用しないようにしてください。

```
postgres=# SELECT pg_terminate_backend (16643);
 pg_terminate_backend
----------------------
t
(1 row)
```

参照

- pg_terminate_backendの詳細は、“PostgreSQL Documentation”の“The SQL Language”の“System Administration Functions”を参照してください。
- アプリケーションの互換性の維持に関する詳細は、“アプリケーション開発ガイド”の“アプリケーションの互換に関する注意事項”を参照してください。

16.5 アクセス異常の対処

アクセスが拒否された場合、以下のディレクトリにインスタンス管理者が操作できる権限を与え、再度操作を実行してください。また、ディスク異常などによりファイルシステムが読み取り専用でマウントされていないか、システムログおよびサーバログを参照して確認してください。読み取り専用でマウントされている場合は、正しくマウントし再度操作を実行してください。

- データ格納先
- テーブル空間格納先
- トランザクションログ格納先
- バックアップデータ格納先

参照

ディレクトリに必要な権限については、“導入ガイド(サーバ編)”の“セットアップ”の“資源配置用ディレクトリの準備”を参照してください。

16.6 データ格納先の容量不足時の対処

データ格納先の容量が不足した場合は、まずディスク上に不要なファイルがないかを確認し、不要なファイルを削除して業務を継続できるようにしてください。

不要なファイルを削除しても問題を解消できない場合は、容量の大きなディスクへのデータの移行が必要になります。

データの移行には、以下の2つの方法があります。

- 16.6.1 テーブル空間を使用する方法
- 16.6.2 容量の大きいディスクにディスク交換する方法

16.6.1 テーブル空間を使用する方法

Fujitsu Enterprise Postgresでは、テーブル空間を使用して、テーブルやインデックスなどのデータベースオブジェクトの格納先を別のディスクに変更できます。

以下に、手順を示します。

1. テーブル空間の作成

   CREATE TABLESPACEコマンドを使用して、新たに用意したディスクにテーブル空間を作成します。

2. テーブル空間の変更

   ALTER TABLEコマンドを使用して新たに定義したテーブル空間にテーブルを変更します。

参照

CREATE TABLESPACEコマンドおよびALTER TABLEコマンドの詳細については、“PostgreSQL Documentation”の“Reference”の“SQL Commands”を参照してください。

16.6.2 容量の大きいディスクにディスク交換する方法

容量の大きいディスクへの交換を実施する場合は、バックアップおよびリカバリ機能を使用してデータ格納先の資源を移行する必要があります。

バックアップとリカバリ操作は、以下の2つの方法で行うことができます。

- 16.6.2.1 WebAdminを使用する場合
- 16.6.2.2 サーバコマンドを使用する場合

以降で、それぞれの操作でディスク交換とデータ格納先の資源の移行を行う場合の手順について説明します。

注意

- ディスク交換を実施する前に、データベースを利用しているアプリケーションおよびインスタンスを停止してください。
- リカバリ操作後は、データベースクラスタのバックアップを推奨します。バックアップすることで、不要になったアーカイブログ(バックアップデータ格納先にコピーされたトランザクションログ)が削除されるため、ディスク容量の確保やリカバリ時間の短縮につながります。

16.6.2.1 WebAdminを使用する場合

WebAdminを使用してディスク交換とデータ格納先の資源の移行を行う場合の手順を以下に示します。

1. ファイルの退避

   データ格納先のディスク配下に必要なファイルが存在する場合、ファイルを退避します。データ格納先の退避は不要です。

2. アプリケーションの停止

   データベースを利用しているアプリケーションを停止してください。

3. データベースクラスタのバックアップ

   最新のデータ格納先の資源をバックアップします。バックアップ方法については、“3.2.1 WebAdminを使用する場合”を参照してください。

4. インスタンスの停止

   インスタンスを停止します。停止方法については、“2.1.1 WebAdminを使用する場合”を参照してください。

5. 容量の大きいディスクへの交換

   ディスクを交換します。その後、ボリュームの構成情報をリカバリします。

6. データベースクラスタのリカバリ

   WebAdminにログインし、リカバリ操作を行います。操作方法については、“16.1.1 WebAdminを使用する場合”の“データ格納ディスク、またはトランザクションログ格納ディスクに障害が発生した場合”の手順4“テーブル空間のディレクトリ作成”から手順7“リカバリの実行”を参照してください。リカバリが正常に完了すると、インスタンスが自動的に起動されます。

7. アプリケーションの再開

   データベースを利用しているアプリケーションを再開してください。

8. ファイルの復元

   手順1“ファイルの退避”で退避したファイルを元に戻します。

16.6.2.2 サーバコマンドを使用する場合

サーバコマンドを使用してディスク交換とデータ格納先の資源の移行を行う場合の手順を以下に示します。

1. ファイルの退避

   データ格納先のディスク配下に必要なファイルが存在する場合、ファイルを退避します。データ格納先の退避は不要です。

2. アプリケーションの停止

   データベースを利用しているアプリケーションを停止してください。

3. データベースクラスタのバックアップ

   サーバコマンドを使用して最新のデータ格納先資源をバックアップします。バックアップ方法については、“3.2.2 サーバコマンドを使用する場合”を参照してください。

4. インスタンスの停止

   バックアップ完了後、インスタンスを停止します。インスタンスの停止方法については、“2.1.2 サーバコマンドを使用する場合”を参照してください。

   インスタンスの停止に失敗した場合は、“16.11 インスタンス停止失敗時の対処”を参照してください。

5. 容量の大きいディスクへ交換

   ディスクを交換します。その後、ボリュームの構成情報をリカバリします。

6. データ格納先のディレクトリ作成

   データ格納先のディレクトリを作成します。テーブル空間を定義していた場合、そのディレクトリも作成してください。

   例

```
$ mkdir /database/inst1
$ chown fsepuser:fsepuser /database/inst1
$ chmod 700 /database/inst1
```

7. キーストアのリカバリとキーストアの自動オープンの有効化

   データベースに格納するデータを暗号化している場合は、データベースのバックアップを取得した時点のキーストアをリストアしてください。キーストアの自動オープンは、必要に応じて設定してください。

8. データベースクラスタのリカバリ

   pgx_rcvallコマンドを使用して、データベースクラスタを復旧します。

   - -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。
   - -Bオプションは、バックアップデータ格納先のディレクトリを指定します。

   例

```
> pgx_rcvall -D /database/inst1 -B /backup/inst1
```

注意

リカバリに失敗した場合は、表示されるエラーメッセージに従ってエラーの原因を取り除き、pgx_rcvallコマンドを再実行してください。

「pgx_rcvall:リカバリ中にエラーが発生しました」のメッセージがある場合は、その後にリカバリ実行時のログが出力されます。ログの最後の十数行以内にエラーの原因が出力されますので、そのメッセージに従ってエラーの原因を取り除き、pgx_rcvallコマンドを再実行してください。

また、リカバリ中に表示される以下のメッセージは、pgx_rcvallコマンドの正常な動作によって出力されるメッセージのため、ユーザーが意識する必要はありません。

```
FATAL:  データベースシステムは起動処理中です
```

参照

pgx_rcvallコマンドの詳細は、“リファレンス”の“pgx_rcvall”を参照してください。

9. インスタンスの起動

   インスタンスを起動してください。

   インスタンスの起動方法については、“2.1.2 サーバコマンドを使用する場合”を参照してください。

10. アプリケーションの再開

    データベースを利用しているアプリケーションを再開してください。

11. ファイルの復元

    手順1“ファイルの退避”で退避したファイルを元に戻します。

16.7 バックアップデータ格納先の容量不足時の対処

バックアップデータ格納先の容量が不足した場合は、まずディスク上に不要なファイルがないかを確認し、不要なファイルを削除するようにしてください。その後、必要であればバックアップを実施してください。

不要なファイルを削除しても問題を解消できない場合は、以下の対処を実施してください。

- 16.7.1 バックアップデータを一時退避する方法
- 16.7.2 容量の大きいディスクにディスク交換する方法

16.7.1 バックアップデータを一時退避する方法

一時的にバックアップデータを別のディレクトリに移動して退避し、バックアップデータ格納先のディスク容量を確保して、バックアップを正常に取得できるようにする方法です。

容量の大きいディスクを準備するまでに時間がかかる場合は、この方法を採用してください。

バックアップデータ格納先の容量が不足すると、アーカイブログをバックアップデータ格納先に格納できなくなります。これに伴い、データ格納先またはトランザクションログ格納先にトランザクションログが溜まり続けます。

そのため対処に時間がかかると、トランザクションログの格納先が満杯になり、業務が継続できない状態に陥ることがあります。

このような状態に陥らないようにするため、バックアップデータ格納先の容量を確保して、アーカイブログを格納できるようにします。

対処には以下の2つの方法があります。

- 16.7.1.1 WebAdminを使用する場合
- 16.7.1.2 サーバコマンドを使用する場合

16.7.1.1 WebAdminを使用する場合

以下の手順でバックアップデータ格納ディスクをリカバリしてください。

1. バックアップデータの一時退避

   バックアップデータを別のディレクトリに移動し、一時退避するとともにバックアップデータ格納先ディレクトリの容量を確保します。

   退避を行う理由は、万が一、復旧するまでの間にデータ格納先のデータが破損した場合でも復旧できるようにするためです。退避先のディスクがなく、データ格納先の破損の危険性がないと判断した場合は、バックアップデータを削除してください。

   下記に、バックアップデータ格納先のディレクトリ(/backup/inst1)にあるバックアップデータを/mnt/usb/backup配下に退避する場合の例を示します。

   例

```
> mkdir /mnt/usb/backup/
> mv /backup/inst1/* /mnt/usb/backup/
```

2. データベースクラスタのバックアップ

   データ格納先に最新のリソースをバックアップします。詳細は“3.2.1 WebAdminを使用する場合”を参照してください。

3. 一時退避したバックアップデータの削除

   バックアップが正常に完了した場合、一時退避したバックアップデータは不要になるため削除します。

   下記に、/mnt/usbに一時退避したバックアップデータを削除する場合の例を示します。

   例

```
> rm -rf /mnt/usb/backup
```

16.7.1.2 サーバコマンドを使用する場合

バックアップデータ格納ディスクをリカバリする手順を説明します。

対処方法には以下の2つの方法があります。

- インスタンスを起動したままリカバリする方法
- インスタンスを停止してリカバリする方法

インスタンス停止の有無による、実行する手順の違いを以下に示します。

No	手順	インスタンス停止の有無	
		なし	あり
1	トランザクションログ二重化の停止	○	－
2	アーカイブログ出力の停止	○	－
3	アプリケーションの停止	－	○
4	インスタンスの停止	－	○
5	バックアップデータの一時退避	○	○
6	アーカイブログ出力の再開	○	－
7	トランザクションログ二重化の再開	○	－
8	インスタンスの起動	－	○
9	バックアップの実行	○	○
10	アプリケーションの再開	－	○
11	一時退避したバックアップデータの削除	○	○

○:実施
－:実施不要

手順を以下に示します。

インスタンスを起動したままリカバリする方法

1. トランザクションログ二重化の停止

   トランザクションログの二重化を停止します。

```
postgres=# SELECT pgx_pause_wal_multiplexing();
LOG:  multiplexing of transaction log files has been stopped
pgx_pause_wal_multiplexing
---------------------------

(1 row)
```

2. アーカイブログ出力の停止

   バックアップ格納ディスクを交換するまでに時間がかかると、トランザクションログが蓄積されていきます。そして、データ格納ディスクまたはトランザクションログ格納ディスクが満杯になると、運用が継続できなくなる危険性があります。

   これを回避するために、以下の方法でアーカイブログの出力を停止させてください。

   ー archive_commandパラメータの変更

     アーカイブログを出力したものとみなすよう、例えば“echo skipped archiving WAL file %f”や“/bin/true”などの、必ず正常終了するコマンドを指定します。

     echoを指定すると、サーバログにメッセージが出力されるため、調査を行う上での指標になる可能性があります。

   ー 設定ファイルの再読込み

     pg_ctl reloadコマンド、またはSQL関数pg_reload_confを実行します。

   なお、運用が継続できなくなる危険性が無く、単にエラーを出力させたくない場合は、archive_commandに空文字列('')を指定して設定ファイルの再読み込みを実施してください。

3. バックアップデータの一時退避

   バックアップデータを別のディレクトリに移動し、退避するとともにバックアップデータ格納先ディレクトリの容量を確保します。

   退避を行う理由は、万が一、次の手順を行うまでの間にデータ格納先のデータが破損した場合でも復旧できるようにするためです。退避先のディスクがなく、データ格納先の破損の危険性がないと判断した場合は、バックアップデータを削除してください。

   下記に、バックアップデータ格納先のディレクトリ(/backup/inst1)にあるバックアップデータを/mnt/usb/backup配下に退避する場合の例を示します。

   例

```
> mkdir /mnt/usb/backup/
> mv /backup/inst1/* /mnt/usb/backup/
```

4. アーカイブログ出力の再開

   archive_commandの設定を元に戻し、設定ファイルを再読込みします。

5. トランザクションログ二重化の再開

   SQL関数pgx_resume_wal_multiplexingを実行します。

   例

```
SELECT pgx_resume_wal_multiplexing()
```

6. バックアップの実行

   pgx_dmpallコマンドを使用して、データベースクラスタをバックアップします。

   pgx_dmpallコマンドには、以下を指定します。

ー -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

例

```
> pgx_dmpall -D /database/inst1
```

7. 一時退避したバックアップデータの削除

バックアップが正常に完了した場合、一時退避したバックアップデータは不要になるため削除します。

下記に、/mnt/usbに一時退避したバックアップデータを削除する場合の例を示します。

例

```
> rm -rf /mnt/usb/backup
```

インスタンスを停止してリカバリする方法

1. アプリケーションの停止

データベースを利用しているアプリケーションを停止してください。

2. インスタンスの停止

インスタンスを停止します。停止方法については、“2.1.2 サーバコマンドを使用する場合”を参照してください。

インスタンスの停止に失敗した場合は、“16.11 インスタンス停止失敗時の対処”を参照してください。

3. バックアップデータの一時退避

バックアップデータを別のディレクトリに移動し、退避するとともにバックアップデータ格納先ディレクトリの容量を確保します。

退避を行う理由は、万が一、次の手順を行うまでの間にデータ格納先のデータが破損した場合でも復旧できるようにするためです。退避先のディスクがなく、データ格納先の破損の危険性がないと判断した場合は、バックアップデータを削除してください。

下記に、バックアップデータ格納先のディレクトリ(/backup/inst1)にあるバックアップデータを/mnt/usb/backup配下に退避する場合の例を示します。

例

```
> mkdir /mnt/usb/backup/
> mv /backup/inst1/* /mnt/usb/backup/
```

4. インスタンスの起動

インスタンスを起動します。起動方法については、“2.1.2 サーバコマンドを使用する場合”を参照してください。

5. バックアップの実行

pgx_dmpallコマンドを使用して、データベースクラスタをバックアップします。

pgx_dmpallコマンドには、以下を指定します。

ー -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

例

```
> pgx_dmpall -D /database/inst1
```

6. アプリケーションの再開

   データベースを利用しているアプリケーションを再開してください。

7. 一時退避したバックアップデータの削除

   バックアップが正常に完了した場合、一時退避したバックアップデータは不要になるため削除します。

   下記に、/mnt/usbに一時退避したバックアップデータを削除する場合の例を示します。

   例

```
> rm -rf /mnt/usb/backup
```

参照

- pgx_rcvallコマンド、およびpgx_dmpallコマンドの詳細は、“リファレンス”の“pgx_rcvall”および“pgx_dmpall”を参照してください。
- archive_commandの詳細は、“PostgreSQL Documentation”の“Server Administration”の“Write Ahead Log”を参照してください。
- pgx_is_wal_multiplexing_paused、およびpgx_resume_wal_multiplexingの詳細は、“B.1 WAL二重化制御関数”を参照してください。

16.7.2 容量の大きいディスクにディスク交換する方法

再び容量が不足することのないように、バックアップデータ格納先のディスクを容量の大きいディスクに交換する方法です。

ディスクを交換した後にバックアップを行い、正しいバックアップを取得します。

バックアップ操作は、以下の2つの方法で行うことができます。

- 16.7.2.1 WebAdminを使用する場合
- 16.7.2.2 サーバコマンドを使用する場合

注意

- ディスク交換を実施する前に、データベースを利用しているアプリケーションを停止してください。

16.7.2.1 WebAdminを使用する場合

以下の手順でバックアップデータ格納ディスクをリカバリしてください。

1. ファイルの退避

   バックアップデータ格納先のディスク配下に必要なファイルが存在する場合、ファイルを退避します。

2. バックアップデータの一時退避

   バックアップデータを別のディレクトリに退避します。

   退避を行う理由は、万が一、次の手順を行うまでの間にデータ格納先のデータが破損した場合でも復旧できるようにするためです。
   退避先のディスクがなく、データ格納先の破損の危険性がないと判断した場合は、バックアップデータを削除してください。

下記に、バックアップデータ格納先のディレクトリ(/backup/inst1)にあるバックアップデータを/mnt/usb/backup配下に退避する場合の例を示します。

例

```
> mkdir /mnt/usb/backup/
> mv /backup/inst1/* /mnt/usb/backup/
```

3. 容量の大きいディスクへの交換

   ディスクを交換します。その後、ボリュームの構成情報をリカバリします。

4. バックアップの実行

   WebAdminにログインし、リカバリ操作を行います。“16.1.1 WebAdminを使用する場合”の“バックアップデータ格納ディスクに障害が発生した場合”の手順2“バックアップデータのリカバリ”および手順3“バックアップの実行”を参照してください。

5. ファイルの復元

   手順1“ファイルの退避”で退避したファイルを元に戻します。

6. 一時退避したバックアップデータの削除

   バックアップが正常に完了した場合、一時退避したバックアップデータは不要になるため削除します。

   下記に、/mnt/usbに一時退避したバックアップデータを削除する場合の例を示します。

   例

```
> rm -rf /mnt/usb/backup
```

16.7.2.2 サーバコマンドを使用する場合

バックアップデータ格納ディスクをリカバリする手順を説明します。

対処方法には以下の2つの方法があります。

- インスタンスを起動したままリカバリする方法
- インスタンスを停止してリカバリする方法

インスタンス停止の有無による、実行する手順の違いを以下に示します。

No	手順	インスタンス停止の有無	
		なし	あり
1	ファイルの退避	○	○
2	バックアップデータの一時退避	○	○
3	トランザクションログ二重化の停止確認	○	－
4	アーカイブログ出力の停止	○	－
5	アプリケーションの停止	－	○
6	インスタンスの停止	－	○
7	容量の大きなディスクに交換	○	○

No	手順	インスタンス停止の有無	
		なし	あり
8	バックアップデータ格納先のディレクトリ作成	○	○
9	アーカイブログ出力の再開	○	－
10	トランザクションログ二重化の再開	○	－
11	インスタンスの起動	－	○
12	バックアップの実行	○	○
13	アプリケーションの再開	－	○
14	ファイルの復元	○	○
15	一時退避したバックアップデータの削除	○	○

○:実施
－:実施不要

手順を以下に示します。

インスタンスを起動したままリカバリする方法

1. ファイルの退避

   バックアップデータ格納先のディスク配下に必要なファイルが存在する場合、ファイルを退避します。バックアップデータ格納先の退避は不要です。

2. バックアップデータの一時退避

   バックアップデータを別のディレクトリに退避します。

   退避を行う理由は、万が一、次の手順を行うまでの間にデータ格納先のデータが破損した場合でも復旧できるようにするためです。退避先のディスクがなく、データ格納先の破損の危険性がないと判断した場合は、バックアップデータを削除してください。

   下記に、バックアップデータ格納先のディレクトリ(/backup/inst1)にあるバックアップデータを/mnt/usb/backup配下に退避する場合の例を示します。

   例

   ```
   > mkdir /mnt/usb/backup/
   > mv /backup/inst1/* /mnt/usb/backup/
   ```

3. トランザクションログ二重化の停止確認

   トランザクションログの二重化が停止されているかを、次のSQL関数で確認します。

   ```
    postgres=# SELECT pgx_is_wal_multiplexing_paused();
    pgx_is_wal_multiplexing_paused
   --------------------------------
    t
   (1 row)
   ```

   トランザクションログの二重化が停止していない場合は、次のSQL関数で停止してください。

   ```
   postgres=# SELECT pgx_pause_wal_multiplexing();
   LOG:  multiplexing of transaction log files has been stopped
   pgx_pause_wal_multiplexing
   ----------------------------

   (1 row)
   ```

4. アーカイブログ出力の停止

   バックアップ格納ディスクを交換するまでに時間がかかると、トランザクションログが蓄積されていきます。そして、データ格納ディスクまたはトランザクションログ格納ディスクが満杯になると、運用が継続できなくなる危険性があります。

   これを回避するために、以下の方法でアーカイブログの出力を停止してください。

   - archive_commandパラメータの変更

     アーカイブログを出力したものとみなすよう、例えば“echo skipped archiving WAL file %f”や“/bin/true”などの、必ず正常終了するコマンドを指定します。

     echoを指定すると、サーバログにメッセージが出力されるため、調査を行う上での指標になる可能性があります。

   - 設定ファイルの再読込み

     pg_ctl reloadコマンド、またはSQL関数pg_reload_confを実行します。

   なお、運用が継続できなくなる危険性が無く、単にエラーを出力させたくない場合は、archive_commandに空文字列('')を指定して設定ファイルの再読み込みを実施してください。

5. 容量の大きいディスクへの交換

   ディスクを交換します。その後、ボリュームの構成情報をリカバリします。

6. バックアップデータ格納先のディレクトリ作成

   バックアップデータ格納先のディレクトリを作成します。

   例

   ```
   # mkdir /backup/inst1
   # chown fsepuser:fsepuser /backup/inst1
   # chmod 700 /backup/inst1
   ```

   詳細については、“3.2.2 サーバコマンドを使用する場合”を参照してください。

7. アーカイブログ出力の再開

   archive_commandの設定を元に戻し、設定ファイルを再読込みします。

8. トランザクションログ二重化の再開

   SQL関数pgx_resume_wal_multiplexingを実行します。

   例

   ```
   SELECT pgx_resume_wal_multiplexing()
   ```

9. バックアップの実行

   pgx_dmpallコマンドを使用して、データベースクラスタをバックアップします。

   pgx_dmpallコマンドには、以下を指定します。

   - -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

   例

   ```
   > pgx_dmpall -D /database/inst1
   ```

10. ファイルの復元

    手順1“ファイルの退避”で退避したファイルを元に戻します。

11. 一時退避したバックアップデータの削除

    バックアップが正常に完了した場合、一時退避したバックアップデータは不要になるため削除します。

    下記に、/mnt/usbに一時退避したバックアップデータを削除する場合の例を示します。

    例

```
> rm -rf /mnt/usb/backup
```

インスタンスを停止してリカバリする方法

1. ファイルの退避

   バックアップデータ格納先のディスク配下に必要なファイルが存在する場合、ファイルを退避します。バックアップデータ格納先の退避は不要です。

2. バックアップデータの一時退避

   バックアップデータを別のディレクトリに退避します。

   退避を行う理由は、万が一、次の手順を行うまでの間にデータ格納先のデータが破損した場合でも復旧できるようにするためです。退避先のディスクがなく、データ格納先の破損の危険性がないと判断した場合は、バックアップデータを削除してください。

   下記に、バックアップデータ格納先のディレクトリ(/backup/inst1)にあるバックアップデータを/mnt/usb/backup配下に退避する場合の例を示します。

   例

```
> mkdir /mnt/usb/backup/
> mv /backup/inst1/* /mnt/usb/backup/
```

3. アプリケーションの停止

   データベースを利用しているアプリケーションを停止してください。

4. インスタンスの停止

   インスタンスを停止します。インスタンスの停止方法については、“2.1.2 サーバコマンドを使用する場合”を参照してください。

   インスタンスの停止に失敗した場合は、“16.11 インスタンス停止失敗時の対処”を参照してください。

5. 容量の大きいディスクへの交換

   ディスクを交換します。その後、ボリュームの構成情報をリカバリします。

6. バックアップデータ格納先のディレクトリ作成

   バックアップデータ格納先のディレクトリを作成します。

例

```
# mkdir /backup/inst1
# chown fsepuser:fsepuser /backup/inst1
# chmod 700 /backup/inst1
```

詳細については、“3.2.2 サーバコマンドを使用する場合”を参照してください。

7. インスタンスの起動

   インスタンスを起動します。起動方法については、“2.1.2 サーバコマンドを使用する場合”を参照してください。

8. バックアップの実行

   pgx_dmpallコマンドを使用して、データベースクラスタをバックアップします。

   pgx_dmpallコマンドには、以下を指定します。

   - -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。

   例

```
> pgx_dmpall -D /database/inst1
```

9. アプリケーションの再開

   データベースを利用しているアプリケーションを再開してください。

10. ファイルの復元

    手順1“ファイルの退避”で退避したファイルを元に戻します。

11. 一時退避したバックアップデータの削除

    バックアップが正常に完了した場合、一時退避したバックアップデータは不要になるため削除します。

    下記に、/mnt/usbに一時退避したバックアップデータを削除する場合の例を示します。

    例

```
> rm -rf /mnt/usb/backup
```

参照

- pgx_rcvallコマンド、およびpgx_dmpallコマンドの詳細は、“リファレンス”の“pgx_rcvall”および“pgx_dmpall”を参照してください。
- archive_modeの詳細は、“PostgreSQL Documentation”の“Server Administration”の“Write Ahead Log”を参照してください。
- pgx_is_wal_multiplexing_paused、およびpgx_resume_wal_multiplexingの詳細は、“B.1 WAL二重化制御関数”を参照してください。

16.8 トランザクションログ格納先の容量不足時の対処

トランザクションログ格納先の容量が不足した場合は、まずディスク上に不要なファイルがないかを確認し、不要なファイルを削除して業務を継続できるようにしてください。

不要なファイルを削除しても問題を解消できない場合は、容量の大きなディスクへのデータの移行が必要になります。

16.8.1 容量の大きいディスクにディスク交換する方法

容量の大きいディスクへの交換を実施する場合は、バックアップおよびリカバリ機能を使用してトランザクションログ格納先の資源を移行する必要があります。

バックアップとリカバリ操作は、以下の2つの方法で行うことができます。

- 16.8.1.1 WebAdminを使用する場合
- 16.8.1.2 サーバコマンドを使用する場合

以降で、それぞれの操作でディスク交換とトランザクションログ格納先の資源の移行を行う場合の手順について説明します。

- ディスク交換を実施する前に、データベースを利用しているアプリケーションおよびインスタンスを停止してください。
- リカバリ操作後は、データベースクラスタのバックアップを推奨します。バックアップすることで、不要になったアーカイブログ(バックアップデータ格納先にコピーされたトランザクションログ)が削除されるため、ディスク容量の確保やリカバリ時間の短縮につながります。

16.8.1.1 WebAdminを使用する場合

WebAdminを使用してディスク交換とトランザクションログ格納先の資源の移行を行う場合の手順を以下に示します。

1. ファイルの退避

   トランザクションログ格納先のディスク配下に必要なファイルが存在する場合、ファイルを退避します。トランザクションログ格納先の退避は不要です。

2. データベースクラスタのバックアップ

   最新のデータ格納先の資源、およびトランザクションログ格納先の資源をバックアップします。バックアップ方法については、“3.2.1 WebAdminを使用する場合”を参照してください。

3. アプリケーションの停止

   データベースを利用しているアプリケーションを停止してください。

4. インスタンスの停止

   インスタンスを停止します。停止方法については、“2.1.1 WebAdminを使用する場合”を参照してください。インスタンスを停止せずにデータベースクラスタのリカバリをした場合は、WebAdminが自動的にインスタンスを停止します。

5. 容量の大きいディスクへの交換

   ディスクを交換します。その後、ボリュームの構成情報をリカバリします。

6. テーブル空間のディレクトリ作成

   バックアップを実施した以降にテーブル空間を定義している場合、そのディレクトリを作成します。

7. キーストアのリカバリとキーストアの自動オープンの有効化

   データベースに格納するデータを暗号化している場合は、以下を行ってください。

   ー データベースのバックアップを取得した時点のキーストアをリストアしてください。

   ー キーストアの自動オープンを有効にしてください。

8. データベースクラスタのリカバリ

   WebAdminにログインし、リカバリ操作を行います。操作方法については、“16.1.1 WebAdminを使用する場合”の“データ格納ディスク、またはトランザクションログ格納ディスクに障害が発生した場合”の手順4“テーブル空間のディレクトリ作成”から手順7“リカバリの実行”を参照してください。リカバリが正常に完了すると、インスタンスが自動的に起動されます。

9. アプリケーションの再開

   データベースを利用しているアプリケーションを再開してください。

10. ファイルの復元

    手順1“ファイルの退避”で退避したファイルを元に戻します。

16.8.1.2 サーバコマンドを使用する場合

サーバコマンドを使用してディスク交換とトランザクションログ格納先の資源の移行を行う場合の手順を以下に示します。

1. ファイルの退避

   トランザクションログ格納先のディスク配下に必要なファイルが存在する場合、ファイルを退避します。トランザクションログ格納先の退避は不要です。

2. データベースクラスタのバックアップ

   サーバコマンドを使用して最新のデータ格納先資源、およびトランザクションログ格納先資源をバックアップします。バックアップ方法については、“3.2.2 サーバコマンドを使用する場合”を参照してください。

3. アプリケーションの停止

   データベースを利用しているアプリケーションを停止してください。

4. インスタンスの停止

   バックアップ完了後、インスタンスを停止します。インスタンスの停止方法については、“2.1.2 サーバコマンドを使用する場合”を参照してください。

   インスタンスの停止に失敗した場合は、“16.11 インスタンス停止失敗時の対処”を参照してください。

5. 容量の大きいディスクへ交換

   ディスクを交換します。その後、ボリュームの構成情報をリカバリします。

6. トランザクションログ格納先のディレクトリ作成

   トランザクションログ格納先のディレクトリを作成します。テーブル空間を定義していた場合、そのディレクトリも作成してください。

   例

```
# mkdir /tranlog/inst1
# chown fsepuser:fsepuser /tranlog/inst1
# chmod 700 /tranlog/inst1
```

7. キーストアのリカバリとキーストアの自動オープンの有効化

   データベースに格納するデータを暗号化している場合は、データベースのバックアップを取得した時点のキーストアをリストアしてください。キーストアの自動オープンは、必要に応じて設定してください。

8. データベースクラスタのリカバリ

   pgx_rcvallコマンドを使用して、データベースクラスタを復旧します。

   - -Dオプションは、データ格納先のディレクトリを指定します。-Dオプションを省略した場合、PGDATA環境変数の値が使用されます。
   - -Bオプションは、バックアップデータ格納先のディレクトリを指定します。

   例

```
> pgx_rcvall -D /database/inst1 -B /backup/inst1
```

注意

リカバリに失敗した場合は、表示されるエラーメッセージに従ってエラーの原因を取り除き、pgx_rcvallコマンドを再実行してください。

「pgx_rcvall:リカバリ中にエラーが発生しました」のメッセージがある場合は、その後にリカバリ実行時のログが出力されます。ログの最後の十数行以内にエラーの原因が出力されますので、そのメッセージに従ってエラーの原因を取り除き、pgx_rcvallコマンドを再実行してください。

また、リカバリ中に表示される以下のメッセージは、pgx_rcvallコマンドの正常な動作によって出力されるメッセージのため、ユーザーが意識する必要はありません。

```
FATAL:  データベースシステムは起動処理中です
```

参照

pgx_rcvallコマンドの詳細は、“リファレンス”の“pgx_rcvall”を参照してください。

9. インスタンスの起動

   インスタンスを起動してください。

   インスタンスの起動方法については、“2.1.2 サーバコマンドを使用する場合”を参照してください。

10. アプリケーションの再開

    データベースを利用しているアプリケーションを再開してください。

11. ファイルの復元

    手順1“ファイルの退避”で退避したファイルを元に戻します。

16.9 各格納先ディスクの異常

各格納先ディスクで異常および資源破壊が発生した場合は、システムログおよびサーバログから異常原因を特定して原因を取り除いてください。

ディスクの異常が以下の組み合わせで発生した場合は、リカバリすることはできません。

インスタンスを再作成して運用環境を再構築してください。

データ 格納ディスク	トランザクションログ 格納ディスク	バックアップデータ 格納ディスク
異常	―	異常
―	異常	異常

参照

インスタンスの作成と運用環境の構築については、“導入ガイド（サーバ編）”の“セットアップ”を参照してください。

16.10 インスタンス起動失敗時の対処

インスタンスの起動に失敗した場合は、システムログおよびサーバログを参照して原因を特定してください。

WebAdminを利用する場合は、原因の対処を実施した後に[対処]ボタンをクリックして[状態を再確認する]をクリックし、インスタンスが正常な状態にあることを確認してください。

以降によくある原因と対処を説明します。

16.10.1 設定ファイルの誤り

設定ファイルをテキストエディタで直接編集した場合や、WebAdminで設定内容の変更を行った場合は、システムログおよびサーバログを参照して以下のファイルに関するメッセージが出力されていないかを確認してください。

- postgresql.conf
- pg_hba.conf

参照

設定ファイルのパラメータの詳細については、以下を参照してください。

- “導入ガイド（サーバ編）”の“設定パラメータ”
- “付録A パラメータ”
- “PostgreSQL Documentation”の“Server Administration”の“Server Configuration”および“Client Authentication”

16.10.2 電源未投入やマウントによる異常

各格納先ディスクのディスク装置の電源を投入し忘れたり、自動マウントの設定を忘れていたために、サーバの再起動などにより、マウントが解除されている場合、インスタンスの起動が失敗します。

“16.14.2 電源未投入やマウントによる異常”を参照して対処してください。

16.10.3 その他の異常

システムログおよびサーバログを参照しても対処できない場合やインスタンスが起動できない場合のリカバリ手順を説明します。

以下の2つの方法でリカバリを行うことができます。

- 16.10.3.1 WebAdminを使用する場合
- 16.10.3.2 サーバコマンドを使用する場合

ただし、バックアップデータ格納先に異常がある場合はリカバリできません。解決できない場合は当社技術員(サポート)に連絡してください。

16.10.3.1 WebAdminを使用する場合

以下の手順でリカバリしてください。

1. データ格納先ディレクトリおよびトランザクションログ格納先ディレクトリの削除

   データ格納先ディレクトリおよびトランザクションログ格納先ディレクトリを退避してから削除します。

2. 状態の再確認

   WebAdminにログインし、[インスタンス]タブのエラーメッセージの[対処]をクリックします。

   [状態を再確認する]をクリックし、各格納先の資源を再度検証します。

3. リカバリの実行

   WebAdminが異常を検出したあと、データベースクラスタを復旧します。

   詳細は、“16.2.1 WebAdminを使用する場合”を参照してください。

16.10.3.2 サーバコマンドを使用する場合

以下の手順でリカバリしてください。

1. データ格納先ディレクトリおよびトランザクションログ格納先ディレクトリの削除

   データ格納先ディレクトリおよびトランザクションログ格納先ディレクトリを退避してから削除します。

2. リカバリの実行

   pgx_rcvallコマンドを使用して、データベースクラスタを復旧します。

   詳細は、“16.2.2 pgx_rcvallコマンドを使用する場合”を参照してください。

16.11 インスタンス停止失敗時の対処

インスタンスの停止に失敗した場合は、システムログおよびサーバログを参照して原因を特定してください。

対処を実施してもインスタンスの停止ができない場合は以下の操作によりインスタンスを停止してください。

以下の2つの方法があります。

- 16.11.1 WebAdminを使用する場合
- 16.11.2 サーバコマンドを使用する場合

16.11.1 WebAdminを使用する場合

[インスタンス]タブから  をクリックし、停止モードの“Fastモード”または“Immediateモード”を選択してインスタンスを停止します。インスタンスが停止できない場合は、WebAdminがサーバプロセスを強制停止します。

停止モードの詳細については、“2.1.1 WebAdminを使用する場合”を参照してください。

16.11.2 サーバコマンドを使用する場合

以下の3つの方法があります。

- Fastモードによる停止

  バックアップが実行中であった場合はバックアップを終了させ、すべての実行中のトランザクションをロールバックして、クライアントとの接続を強制的に切断した後、インスタンスを停止します。

- Immediateモードによる停止

  インスタンスを即座に強制的に終了させます。インスタンスの再起動時にはクラッシュリカバリが実行されます。

- サーバプロセスの強制停止

  他の方法を実施しても停止できない場合に、サーバプロセスを確実に停止させます。

16.11.2.1 Fastモードによる停止

pg_ctlコマンドに“-m fast”を指定してインスタンスを停止します。

この方法を実施しても停止が失敗する場合は、“16.11.2.2 Immediateモードによる停止”、または“16.11.2.3 サーバプロセスの強制停止”に従った停止を実施してください。

 例

```
> pg_ctl stop -D /database/inst1 -m fast
```

16.11.2.2 Immediateモードによる停止

pg_ctlコマンドに“-m immediate”を指定してインスタンスを停止します。

この方法を実施しても停止が失敗する場合は、“16.11.2.3 サーバプロセスの強制停止”に従って停止を行ってください。

例

```
> pg_ctl stop -D /database/inst1 -m immediate
```

16.11.2.3 サーバプロセスの強制停止

Fastモード、Immediateモードのいずれの方法でも停止できない場合は、pg_ctlコマンドのkillパラメータ、またはkillコマンドを使用してサーバプロセスを強制停止します。

手順を以下に示します。

1. psコマンドを実行します
   なお、“<x>”は、製品のバージョンを示します。

```
> ps axwfo user,pid,ppid,tty,command | grep postgres
fsepuser    19174 18027 pts/1                ¥_ grep postgres
fsepuser    20517     1 ?        /opt/fsepv<x>server64/bin/postgres -D /database/inst1
fsepuser    20518 20517 ?         ¥_ postgres: logger
fsepuser    20520 20517 ?         ¥_ postgres: checkpointer
fsepuser    20521 20517 ?         ¥_ postgres: background writer
fsepuser    20522 20517 ?         ¥_ postgres: walwriter
fsepuser    20523 20517 ?         ¥_ postgres: autovacuum launcher
fsepuser    20524 20517 ?         ¥_ postgres: archiver
fsepuser    20525 20517 ?         ¥_ postgres: logical replication launcher
```

   プロセスID(20517)がサーバプロセスになります。

2. サーバプロセスを強制停止します。

   インスタンス管理者で、サーバプロセスの強制停止を行います。

   pg_ctlコマンドの場合

```
> pg_ctl kill SIGQUIT 20517
```

   killコマンドの場合

```
> kill -s SIGQUIT 20517
```

16.12 ストリーミングレプリケーションのスタンバイインスタンス作成失敗時の対処

WebAdminを使ってスタンバイインスタンスの作成に失敗した場合は、システムログやサーバログを参照して異常の原因を特定してください。

WebAdminを使ってスタンバイインスタンスを作成する際にエラーがあった場合、作成途中のスタンバイインスタンスの作成が再開して完了する見込みはありません。

エラーの原因を修正し、作成途中のスタンバイインスタンスを削除してから、新規にスタンバイインスタンスを作成してください。この提案は以下のような想定に基づいています。

- インスタンスの作成が完了していないため、データベースに接続しているアプリケーションが無い
- スタンバイインスタンスでエラーがあり、起動していない
- スタンバイインスタンスのバックアップが無いため、リカバリできない

参照

インスタンスの削除方法についての詳細は、“導入ガイド(サーバ編)”の“インスタンスの削除”を参照してください。

16.13 分散トランザクションの異常時の対処

分散トランザクションを利用したアプリケーションの運用時に、サーバがダウンするなどのシステム異常が発生すると、トランザクションがインダウト状態になることがあります。

このとき、トランザクションで占有した資源がロックされ、他のトランザクションから当該資源へのアクセスがブロックされて利用不可になります。

以降にインダウトトランザクションの確認方法と解決方法を説明します。

インダウトトランザクションの確認方法

分散トランザクションを利用したアプリケーションの動作中に、サーバまたはクライアントがダウンした場合、インダウトトランザクションが発生している可能性があります。

確認方法を以下に示します。

サーバがダウンした場合

1. サーバの再起動時のログに、以下のようなメッセージが出力されていると、インダウトトランザクションが発生していると判断できます。

   例

   ```
   LOG:  準備されたトランザクション2103を復旧しています
   ```

2. システムビューpg_prepared_xactsを参照して、準備されたトランザクションに関する情報を取得します。

   準備されたトランザクションの一覧にあるトランザクション識別子(pg_prepared_xacts のtransactionカラム)が、再起動時のログから取得したインダウトトランザクションのトランザクション識別子と一致している場合、その行がインダウトトランザクションに関する情報です。

   例

   ```
   postgres=# select * from pg_prepared_xacts;
    transaction |   gid     |   prepared |  owner   | database
   -------------+-----------+------------+----------+----------
   2103 | 374cc221-f6dc-4b73-9d62-d4fec9b430cd | 2022-03-06 16:28:48.471+08 | postgres | postgres  (1 row)
   ```

   transactionカラムが2103の行に、インダウトトランザクションに関する情報が出力されています。

クライアントがダウンした場合

クライアントが1つも接続していないときに、pg_prepared_xacts に準備されたトランザクションが存在した場合、そのトランザクションはインダウト状態であると判断できます。

クライアントが1つ以上接続しているときに、pg_prepared_xactsに準備されたトランザクションが存在しても、インダウト状態かどうか判断できません。この場合、下記のクエリで、取得したデータベース名、ユーザー名、PREPARE TRANSACTION を実行した時刻とアクセス先のテーブル名の情報から、インダウトトランザクションを特定してください。

```
select gid,x.database,owner,prepared,l.relation::regclass as relation from pg_prepared_xacts x left join pg_locks l on
l.virtualtransaction = '-1/'||x.transaction and l.locktype='relation';
```

これだけでは特定できない場合は、十分に時間が経過した後に、再度pg_prepared_xacts を調べてください。

前回調べたときから継続しているトランザクションがあれば、そのトランザクションがインダウト状態である可能性が高いと判断できます。

ポイント

ここで説明したように、確実にインダウトトランザクションを特定する汎用的な方法は存在しません。

これまでに述べた方法で特定できるように、何らかの補助的な情報の採取(例:クライアント側でのロギング)や、運用方法(例:業務毎にデータベースユーザーを割り当てる)を検討してください。

インダウトトランザクションの解決方法

前述のシステムビューpg_prepared_xactsからインダウトトランザクションのグローバルトランザクション識別子(pg_prepared_xacts のgidカラム)を取得し、ROLLBACK PREPARED文またはCOMMIT PREPARED文を発行することで、インダウトトランザクションを解決します。

例

- インダウトトランザクションをロールバックする場合

```
postgres=# rollback prepared '374cc221-f6dc-4b73-9d62-d4fec9b430cd';
ROLLBACK PREPARED
```

- インダウトトランザクションをコミットする場合

```
postgres=# commit prepared '374cc221-f6dc-4b73-9d62-d4fec9b430cd';
COMMIT PREPARED
```

16.14 ディスク障害以外の入出力異常

実際にはディスクが故障していなくても、故障している場合と同じ入出力異常のメッセージが出力されることがあります。

これには、以下のような場合があります。それぞれについて対処を示します。

- 16.14.1 外部ディスクとの間のネットワーク異常
- 16.14.2 電源未投入やマウントによる異常

16.14.1 外部ディスクとの間のネットワーク異常

外部ディスクとの間のネットワーク経路に発生する異常です。

システムログやサーバログの内容、さらにディスク装置のランプ、ネットワーク配線、およびネットワークカードの状態などから異常の原因を特定したあと、問題のある機器の取り替えなどの処置を行って、異常を取り除いてください。

16.14.2 電源未投入やマウントによる異常

ディスク装置の電源を投入し忘れたり、ディスクの自動マウントの設定を忘れていた、あるいはマウントを誤って解除してしまったために発生する異常です。

システムログおよびサーバログの内容や、ディスク装置の電源ランプ、ディスクのマウント状態などを確認し、問題があった場合は、対処を行ってください。

マウントが解除されてしまっている場合は、誤ってマウントを解除してしまったか、あるいは、OSの起動時に自動でマウントを行うよう設定されていない可能性があります。この場合は、自動でマウントを行うように設定してください。

16.15 異常検知と対処

コマンドラインインタフェースで以下の操作を行うと、WebAdminで異常が発生します。

- postgresql.confのportとbackup_destinationパラメータの変更
- WebAdminを使用して追加されたクラスタレプリケーションのMirroring Controller設定の変更

WebAdminの異常検出とその際に発生する事象について説明します。

16.15.1 ポート番号とバックアップ格納パスの異常

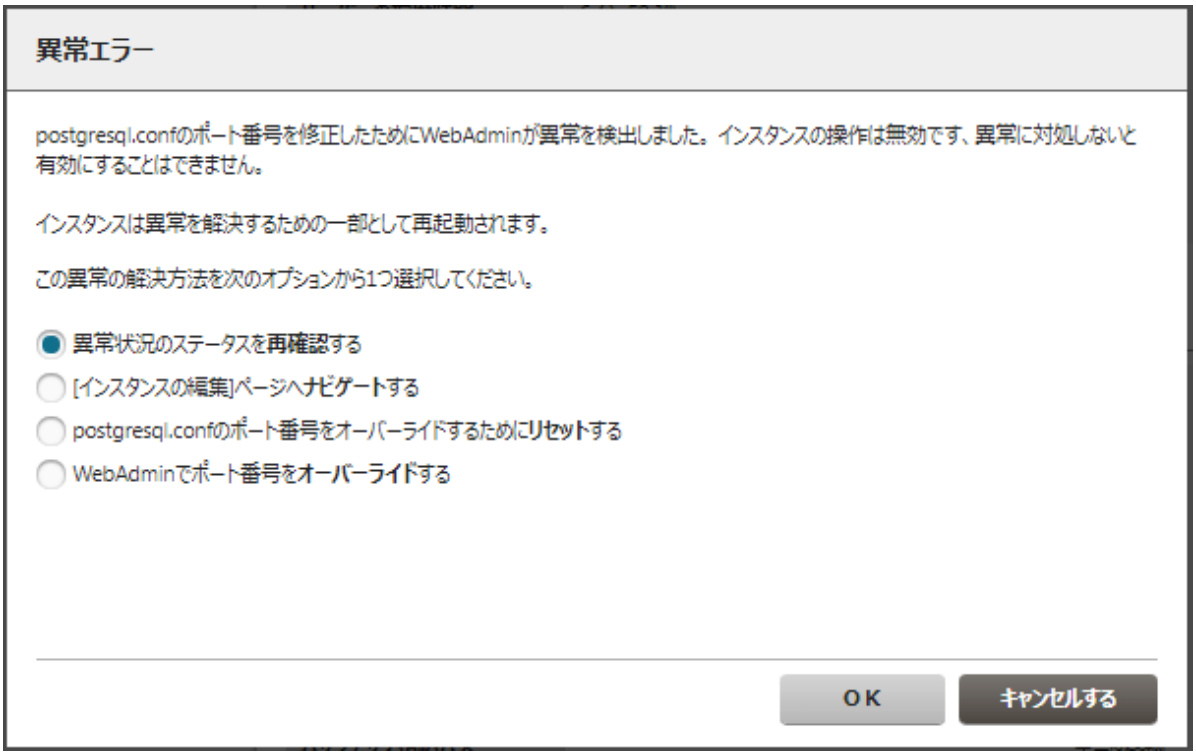
WebAdminの[ポート番号]と[バックアップ格納パス]のどちらか、または両方の値がpostgresql.confの対応するパラメータ(portとbackup_destination)と異なる場合に異常が発生します。

WebAdminは、参照のためにインスタンスが選択されるときや、インスタンスの操作が行われるときに異常を確認します。選択されているインスタンスのみの異常を検出します。

ポート番号とバックアップ格納パスのどちらか、または両方に異常が検出されると、以下の事象が起こります。

- [インスタンスを編集する]、[インスタンスのリフレッシュ]、[Mirroring Controllerを削除する]以外のすべてのインスタンス操作のボタンが無効になります。
- インスタンスアイコンに赤いエラーステータスインジケータが表示されます。
- バックアップ格納パス固有の異常である場合、赤いエラーステータスインジケーターが[バックアップ格納パス]のディスクアイコンに表示され、[バックアップ格納ステータス]が“異常”になります。
- WebAdminが何らかの異常を検出したというメッセージが、関連する[対処]ボタンと共に[メッセージ]欄に表示されます。

[対処]をクリックします。[異常エラー]ダイアログボックスが表示されます。



必要なオプションを選択して[OK]をクリックし、異常エラーに対処します。

[インスタンスを編集する]ページについては、“導入ガイド(サーバ編)”の“インスタンス情報の編集”を参照してください。

注意

異常対処中に重大なエラーが起きた場合、表示はされますが、インスタンスを前の状態に戻すロールバックはサポートされていません。

16.15.2 Mirroring Controllerの異常

以下の状況になるとMirroring Controller の異常が発生します。

- Mirroring Controller管理フォルダか構成ファイルが削除された

- Mirroring Controller管理フォルダか構成ファイルの権限が以下のように変更された
  - インスタンス管理者のMirroring Controller設定へのアクセスが拒否される
  - インスタンス管理者以外のユーザーにMirroring Controller構成ファイルへのアクセス権限がある

WebAdminはMirroring Controllerのステータスの確認時に異常を確認します。

Mirroring Controllerの異常が検出されると以下の事象が起こります。

- "Mirroring Controllerを削除する"以外のすべてのMirroring Controller機能がレプリケーションクラスタに対して無効になります。
- [Mirroring Controllerのステータス]が"異常"になります。
- 以下のいずれかのメッセージが[メッセージ]欄に表示されます。
  - "Mirroring Controller管理フォルダまたは設定ファイル'パス'にアクセスできませんでした。Mirroring Controllerの機能が無効にされています。Mirroring Controllerを削除し、再度追加することを検討してください。"
  - "Mirroring Controller管理フォルダまたは設定ファイル'パス'が見つかりませんでした。Mirroring Controllerの機能が無効にされています。Mirroring Controllerを削除し、再度追加することを検討してください。"

付録A　パラメータ

Fujitsu Enterprise Postgresのpostgresql.confファイルに設定するパラメータについて説明します。

postgresql.confファイルは、データ格納先に配置されています。

参考

4バイト符号付き整数の最大値は、OSにより異なります。使用するOSの定義に従ってください。

- core_directory（文字列）

  コアファイルを出力するディレクトリを指定します。本パラメータを指定しない場合は、データ格納先のディレクトリを指定したものとみなされます。本パラメータは、インスタンス起動時にパラメータ指定することでのみ設定できます。インスタンス起動中に動的変更することはできません。

- core_contents（文字列）

  コアファイルに含める内容を指定します。

  ー full：サーバプロセスのメモリの内容をすべてコアファイルに出力します。

  ー none：コアファイルを出力しません。

  ー minimum：サーバプロセスのうち、共有メモリ以外の内容をコアファイルに出力します。コアファイルのサイズは小さくなります。ただし、場合によってはコアファイルが出力された原因の調査に必要な情報が十分に得られないことがあります。

  本パラメータを指定しない場合は、“minimum”を指定したものとみなされます。本パラメータは、インスタンス起動時にパラメータ指定することでのみ設定できます。インスタンス起動中に動的変更することはできません。

- keystore_location（文字列）

  キーストアのファイルを格納するディレクトリを指定します。他のデータベースクラスタとは異なる場所を指定してください。本パラメータは、インスタンス起動時にパラメータ指定することでのみ設定できます。インスタンス起動中に動的変更することはできません。

  tde_kms.kms_conninfo_fileパラメータと同時に指定することはできません。

- tde_kms.plugin_path（文字列）

  プラグインを使用して鍵管理システムに接続する場合に、プラグインを格納するディレクトリを絶対パスで指定します。このディレクトリにはデータベース管理者だけがプラグインを格納できるようにしてください。インスタンス起動時にパラメータ指定することでのみ設定できます。

- tde_kms.enable_shared_dek（論理型）

  透過的データ暗号化における暗号化テーブル空間ごとのデータ暗号化キーをバックエンドプロセス間で共有することを有効もしくは無効にします。デフォルトはoffです。インスタンス起動時にパラメータ指定することでのみ設定できます。

- tde_kms.max_shared_dek（数値）

  透過的データ暗号化においてテーブル空間ごとのデータ暗号化キーを共有する場合の共有されるデータ暗号化キーの最大数を指定します。デフォルトは1000です。インスタンス起動時にパラメータ指定することでのみ設定できます。

- tde_kms.kms_conninfo_file（文字列）

  鍵管理サービスをキーストアとして使用する場合に、鍵管理システムの接続情報を記述したファイルを指定します。keystore_locationパラメータと同時に指定することはできません。

鍵管理システムの接続情報ファイルは以下のいずれかの形式で作成してください。

```
kmip    kms-name    address    port    auth-method    [auth-options]
custom  kms-name    plugin-name      [plugin-options]   [extra-args]
```

鍵管理システムに接続する方法として以下のいずれかを指定します。

－ kmip

この鍵管理システムにKMIPプロトコルを使用してアクセスします。この鍵管理システムにある暗号化キーを持ち出して使用します。

－ custom

プラグインモジュールを利用して鍵管理システムにアクセスします。暗号化/復号処理はプラグイン内、または鍵管理システム内で行われます。Fujitsu Enterprise Postgresは暗号鍵を直接的には使用しません。

タイプkmipの場合

－ kms-name

鍵管理システムに付けた鍵管理システム名で、マスタ暗号化キーの宣言やキーストアのオープン時に指定します。鍵管理システムの名称はこのファイル内で一意である必要があります。鍵管理システム名は、a-zで始まりa-z、数字(0-9)およびアンダースコアから構成される63文字以下の文字列でなければなりません。大文字と小文字は同一視されます。

－ address

鍵管理サービスのホスト名またはIPアドレスを指定します。

－ port

鍵管理サービスがサービスを待ち受けるポート番号を指定します。

－ auth-method

鍵管理サービスにおける認証方式を指定します。

－ auth-options

auth-methodに応じて、そのあとに認証方式のオプションを指定します。オプションはname=value形式のフィールドで複数指定することができます。

タイプkmipの鍵管理サービスを使用する場合の認証方式

cert

証明書を使用してKMIPサーバとクライアントであるFujitsu Enterprise Postgres側の相互を認証します。auth-optionsに以下を指定できます。

- sslcert

クライアント証明書のファイル名を指定します。対応するフォーマットはPEM形式です。

- sslkey

クライアント証明書に対して使用される秘密化キーのファイル名を指定します。対応するフォーマットはPEM形式です。ファイルをパスフレーズで暗号化する場合は、最大1023バイト以下のパスフレーズを使用してください。

- sslkeypassphrase-obf

sslkeyで指定した秘密化キーファイルの難読化されたパスフレーズが格納されるファイルを指定します。このオプションを指定することで、サーバ起動時にキーストアを自動でオープンできるようになります。難読化されたファイルの作成はpgx_keystoreコマンドで行います。省略できます。

- sslrootcert

SSL認証局の証明書のファイル名を指定します。対応するフォーマットはPEM形式です。接続先のサーバ証明書を検証するために利用されます。

例

```
kmip   mykmipsvr  mykmipsvr.example.com   5696   cert   sslcert=postgres.crt   sslkey=postgres.key
sslrootcert=root.crt
```

注意

cert認証では、接続先のサーバが接続しようとしているサーバと同一かどうかを確認しません。sslrootcertで指定する認証局の証明書で署名されているサーバ証明書が使われているサーバであれば、正しい接続先と見なします。この動作による問題を回避するためには、KMIPサーバは独自CAまたは自己署名証明書を利用することを検討してください。

タイプcustomの場合

- kms-name

  鍵管理システムに付けた鍵管理システム名で、マスタ暗号化キーの宣言やキーストアのオープン時に指定します。鍵管理システムの名称はこのファイル内で一意である必要があります。鍵管理システム名は、a-zで始まりa-z、数字(0-9)およびアンダースコアから構成される63文字以下の文字列でなければなりません。大文字と小文字は同一視されます。

- plugin-name

  プラグインの名称を指定します。tde_kms.plugin_pathで指定したディレクトリにあるプラグインの名称と同じ名前の実行可能ファイルをプラグインモジュールとして使用します。

- plugin-options

  プラグインのその他のオプションを指定します。オプションはname=value形式のフィールドで複数指定することができます。以下のオプション名を指定できます。

  - kms-secret-obf

    透過的データ暗号化を利用してキーストアの自動オープンを有効化する場合に、難読化されたKMSシークレットが格納されたファイルを指定します。難読化されたファイルの作成はpgx_keystoreコマンドで行います。難読化されたファイルの内容はFujitsu Enterprise Postgresが復号しプラグインに渡されます。透過的データ暗号化を利用してキーストアの自動オープンを有効にしない場合は省略できます。

- extra-args

  指定したシェルコマンドに渡す追加の引数を、arg=value形式で指定します。複数のextra-argsを指定した場合、その順序でvalueがシェルコマンドに渡されます。

例

```
custom   mykms  mykms  arg=--profile arg=user1
```

- tablespace_encryption_algorithm(文字列)

  作成するテーブル空間の暗号化アルゴリズムを指定します。有効な値はAES128、AES256、またはnoneです。noneを指定した場合、暗号化は行いません。デフォルト値はnoneです。暗号化を行う場合、AES256の指定を推奨します。スーパーユーザーのみこの設定を変更できます。

- backup_destination(文字列)

  pgx_dmpallコマンドがバックアップデータを格納するディレクトリを絶対パスで指定します。他のデータベースクラスタとは異なる場所を指定してください。本パラメータは、インスタンス起動時にパラメータ指定することでのみ設定できます。インスタンス起動中に動的変更することはできません。

このディレクトリは、バックアップするデータ格納先のディレクトリ、テーブル空間ディレクトリ、およびトランザクションログ格納先のディレクトリの外に配置してください。このディレクトリの内容はデータベースシステムが管理するため、利用者は任意のファイルを格納しないように注意してください。

- search_path(文字列)

  Oracleデータベース互換のSUBSTR関数を利用する場合は、“search_path”パラメータに、“oracle”および“pg_catalog”を設定してください。“oracle”は、“pg_catalog”よりも前に指定する必要があります。

  例

```
search_path = '"$user", public, oracle, pg_catalog'
```

  参考

  - search_pathは、スキーマ検索パスの優先順位を指定する機能です。Oracleデータベース版のSUBSTR関数はoracleスキーマに定義されます。
  - search_pathについては、“PostgreSQL Documentation”の“Server Administration”の“Statement Behavior”を参照してください。

- track_waits(文字列)

  pgx_stat_lwlockおよびpgx_stat_latchに対し、統計情報の収集を有効にします。

  - on:統計情報の収集を有効にします。
  - off:統計情報の収集を無効にします。

  本パラメータを指定しない場合は、“on”を指定したものとみなされます。

  スーパーユーザーのみ、この設定を変更できます。

- track_sql (文字列)

  pgx_stat_sqlに対し、統計情報の収集を有効にします。

  - on:統計情報の収集を有効にします。
  - off:統計情報の収集を無効にします。

  本パラメータを指定しない場合は、“on”を指定したものとみなされます。

  スーパーユーザーのみ、この設定を変更できます。

インメモリ機能に関するパラメータ

- reserve_buffer_ratio (数値)

  ステーブルバッファテーブルに使用する共有メモリの割合をパーセンテージで指定します。

  - 最小値:0
  - 最大値:80

  本パラメータを指定しない場合は、0を指定したものとみなされます。

・ vci.cost_threshold (数値)

VCIを利用した実行計画を選択する最も低いコストです。VCIを利用しない最良の実行計画のコスト値がこの値より低い場合には、そのVCIを利用しない実行計画を選択します。

－ 最小値:0

－ 最大値:4バイト符号付き整数の最大値

範囲外の値が指定された場合は、18000が設定されます。

本パラメータを指定しない場合は、18000を指定したものとみなされます。

・ vci.control_max_workers (数値)

VCIを管理するバックグラウンドワーカー数を指定します。インスタンス全体のワーカ数はmax_worker_processesにより制限されるため、max_worker_processesにここで設定した値を足してください。

－ 最小値:1

－ 最大値:8388607

範囲外の値が指定された場合は、8が設定されます。

本パラメータを指定しない場合は、8を指定したものとみなされます。

・ vci.enable (文字列)

VCIの有効、または無効を指定します。

－ on:VCIを有効にします。

－ off:VCIを無効にします。

本パラメータを指定しない場合は、“on”を指定したものとみなされます。

・ vci.log_query (文字列)

vci.max_local_rosで指定したメモリが不足したことでVCI利用ができなかった場合の、ログ出力の有効、または無効を指定します。

－ on:ログ出力を有効にします。

－ off:ログ出力を無効にします。

本パラメータを指定しない場合は、“off”を指定したものとみなされます。

・ vci.maintenance_work_mem (数値)

CREATE INDEXなどのVCIのメンテナンス時に使用するメモリの最大サイズを指定します。

－ 最小値:1MB

－ 最大値:4バイト符号付き整数の最大値

範囲外の値が指定された場合は、256MBが設定されます。

本パラメータを指定しない場合は、256MBを指定したものとみなされます。

・ vci.max_local_ros (数値)

VCI検索時に使用するメモリの最大サイズを指定します。

－ 最小値:64MB

－ 最大値:4バイト符号付き整数の最大値

範囲外の値が指定された場合は、64MBが設定されます。

本パラメータを指定しない場合は、64MBを指定したものとみなされます。

・ vci.max_parallel_degree (数値)

並列検索で使用するバックグラウンドワーカーの最大数を指定します。インスタンス全体のワーカ数はmax_worker_processesにより制限されるため、max_worker_processesにここで設定した値を足してください。

並列検索で使用するバックグラウンドワーカーの最大数は、-8388607～8388607の範囲で指定可能です。

－ 1以上の整数:指定した並列度で検索処理を実施します。

－ 0:並列検索処理を停止します。

－ 負数:環境から取得した最大CPU数から指定値を引いた並列度で、検索処理を実施します。

範囲外の値が指定された場合は、0が設定されます。

本パラメータを指定しない場合は、0を指定したものとみなされます。

・ vci.shared_work_mem (数値)

VCI並列検索時に使用するメモリの最大サイズを指定します。

－ 最小値:32MB

－ 最大値:4バイト符号付き整数の最大値

範囲外の値が指定された場合は、1GBが設定されます。

本パラメータを指定しない場合は、1GBを指定したものとみなされます。

Global Meta Cache機能に関するパラメータ

・ pgx_global_metacache (数値)

GMC領域のメモリサイズを指定します。

次の計算式で算出された値を指定してください。

算出された値よりも小さい値を指定しても動作しますが、メタキャッシュがGMC領域に入りきらなくなるかもしれません。

その場合にはシステムが不要だと思ったメタキャッシュを破棄しますが、再度そのメタキャッシュが必要な場合にはメタキャッシュを展開する必要があるので、十分な性能を発揮できなくなります。

10MBよりも小さい値で、かつ、機能を無効にする0以外の値ではGlobal Meta Cache機能が動作できないためにデータベースの起動が失敗します。

0を設定するとGlobal Meta Cache機能は無効になります。デフォルトは0です。

この設定を変更するときはデータベースの再起動が必要です。

```
GMC領域のメモリサイズ
  = Max(10MB,
     (全ユーザーテーブル個数 × 0.4 KB
       + 全ユーザーインデックス個数 × 0.3 KB
       + 全ユーザーカラム数 × 0.8 KB) × 1.5 (注))

  注) 安全係数 (1.5)
  テーブル定義変更やシステムカタログの行が変更された場合に、一時的に変更前のGMCと変更後のGMCが共有メモリ上に存在する場合を考慮した値です。
```

・ track_gmc (文字列)

pgx_stat_gmcに対し、統計情報の収集を有効にします。

ー on:統計情報の収集を有効にします。

ー off:統計情報の収集を無効にします。

本パラメータを指定しない場合は、“on”を指定したものとみなされます。

スーパーユーザーのみ、この設定を変更できます。

Local Meta Cache制限機能に関するパラメータ

- pgx_catalog_cache_max_size(数値)

  バックエンドプロセスがカタログキャッシュとして使用するメモリ量の上限値を指定します。

  8KB以上に設定することでカタログキャッシュ削除を有効にできます。

  0を設定するとカタログキャッシュ削除は無効となります。デフォルト値は0です。

  単位を指定しない場合はKBとして扱われます。

  ー 最小値:8KB

  ー 最大値:4バイト符号付き整数の最大値

  パラメータ設定のための計算をする際、キャッシュサイズを決める要因を、テーブル数、インデックス数、カラム数として計算します。カタログキャッシュやリレーションキャッシュとして保持されるものには、データベース、ロール、あるいは、プロシジャなどのオブジェクトも含まれますが、これらの影響は上記の要因と比べて小さいためです。また、与えられたメモリをカタログキャッシュとリレーションキャッシュとで分配するので、pgx_relacion_cache_max_sizeの計算方法も含んでいます。

注意

ここでの計算方法は、どのバックエンドも同じようなアクセスをし、そのトランザクションも同程度の数の資源にアクセスすることを前提としています。特異的なバックエンドやトランザクションが少数ある場合には、それを誤差として除外して考えてください。

1. あるバックエンドプロセスが使用できるメモリ量を決めます。搭載メモリからデータベースキャッシュなどのシステム全体が必要とするメモリサイズを引いて、残りをコネクション数で除算するなどして決めてください。
2. 最も良い性能が得られる値として、バックエンドが、その寿命の中でアクセスする全て資源についてのカタログキャッシュを保持したときのカタログキャッシュの総メモリサイズを以下の計算式で計算します。

   Global Meta Cacheの有効/無効により、メモリ量は変わります。Global Meta Cacheを有効にしていると、キャッシュの大部分が共有メモリ上に配置されるので、必要とされるメモリ量が小さくなります。

   ```
   Global Meta Cache有効時：
     （アクセスするテーブル数 + アクセスするインデックス数 + アクセスするカラム数）
         × 0.1KB × 1.5（注）
   Global Meta Cache無効時：
     {アクセスするテーブル数 × 0.5KB(pg_classのタプルサイズ)
         + アクセスするインデックス数 × 0.5KB(pg_indexのタプルサイズ)
         + アクセスするカラム数 × 1.0KB(pg_statisticのタプルサイズ)} × 1.5（注）
   ```

   注）安全係数(1.5)
   システムカタログには、可変長の型を持つカラムがあります。例えば、テーブル数に乗算している定数値はpg_classのタプルサイズですが、pg_classのrelnameは可変長データです。
   すべての定義について、細かに計算することは現実的ではないので、上記の式に50%を上乗せしています。

3. 2.と同様に、リレーションキャッシュについて以下の計算式で計算します。

   ```
   （1.4KB × アクセスするテーブル数 + 2.4KB × アクセスするインデックス数）× 1.5（注）
   ```

   注）安全係数(1.5)
   リレーションキャッシュは、テーブルやインデックスの定義を利用しやすいように構造化したものであり、様々なオブジェクトへのポインタを保持しており、それらのオブジェクトも含めてサイズが計上されます。また、テーブル定義によってメモリ割り当てされるオブジェクトの種類やその大きさが変わるため、可変長となります。
   すべての定義に対応して計算することは現実的ではないため、50%を上乗せしています。

4. もし、1.の値 ≧ 2.の値 + 3.の値ならば、バックエンドプロセスは、許容された範囲ですべてのキャッシュを保持することが可能なので、キャッシュを制限する必要はありません。安全のためにキャップをしたいならば、2.の値をpgx_catalog_cache_max_sizeに設定し、3.の値をpgx_relation_cache_max_sizeに設定します。
5. 1.の値 ＜ 2.の値 + 3.の値ならば、キャッシュを制限する必要があります。しかし、1つのトランザクションが使用するキャッシュのサイズをこのパラメータで制限することはできません。したがって、以降のステップを踏んでください。
6. あるトランザクションが使用するカタログキャッシュを2.の計算式で計算します。
7. あるトランザクションが使用するリレーションキャッシュを3.の計算式で計算します。
8. 1.の値 ＜ 6.の値 + 7.の値ならば、1.の値を大きくする必要があります。つまり、場合によっては、搭載メモリを増やしたり、コネクション数を少なくする必要があります。
9. 1.の値 ≧ 6.の値 + 7.の値ならば、本パラメータでキャッシュを制限することで1.の条件を満たせます。1.の値を、2.と3.の比で分配して、パラメータに設定します。2.に分配した値をpgx_catalog_cache_max_sizeに設定し、3.に分配した値をpgx_relation_cache_max_sizeに設定します。
10. 9.で算出した値は、暫定的な値です。もし、目標の性能を満たすことができないならば、リレーションキャッシュに配分の重点を移すことを最初に試みてください。SQLを実行するときには、カタログキャッシュを元にして生成したリレーションキャッシュを主に参照するので、リレーションキャッシュを多く残しておいた方が有利だからです。それでも性能を満たすことができなければ、“14.1.4 Local Meta Cache制限機能の性能への影響とパラメータ調整”を参照しながら、パラメータを調整してください。

注意

テーブルをパーティショニングしている場合には、注意が必要です。

SQL文で親テーブルを指定するか、子テーブルを指定するかでキャッシュされる定義が変わります。特に、親テーブルを指定した場合には、すべての子テーブルの定義がキャッシュされることに注意してください。なぜなら、SQL文で親テーブルを指定する場合には、目的のデータが格納される子テーブルを決定するために、すべての子テーブルの定義を知る必要があるからです。なお、親テーブルのカラム情報は、キャッシュされません。

親テーブルを指定する場合:

```
アクセスするテーブル数 ＝ アクセスする親テーブル数 ＋ 定義された子テーブル数
カラム数 ＝ 定義されたカラム数 × 定義された子テーブル数
```

子テーブルを直接指定する場合:

```
アクセスするテーブル数 ＝ 実際にアクセスする子テーブル数
カラム数 ＝ 定義されたカラム数 × 実際にアクセスする子テーブル数
```

例）

親テーブルT(インデックス数1、カラム数3)が子テーブルT1～T5(それぞれインデックス数1、カラム数3)に分割されているとします。SQLに親テーブルTを指定し、クエリの対象となるデータが存在する子テーブルがT1とT2に限定されていて、データアクセス時にT1とT2上に定義されたインデックスを使用した場合には、以下のように計算します。

```
テーブル数 = 1(親テーブル) + 5(子テーブル) = 6
インデックス数 = 2(アクセスするインデックス)
カラム数 = 3(カラム数) × 5(子テーブル) = 15
```

SQLに子テーブルT1とT2を指定し、データアクセス時にT1とT2上に定義されたインデックスを使用した場合には、以下のように計算します。

```
テーブル数 = 2(子テーブル)
インデックス数 = 2(アクセスするインデックス)
カラム数 = 3(カラム数) × 2(子テーブル) = 6
```

- pgx_relation_cache_max_size(数値)

  バックエンドプロセスがリレーションキャッシュとして使用するメモリ量の上限値を指定します。

8KB以上に設定することでリレーションキャッシュ削除を有効にできます。

0を設定するとリレーションキャッシュ削除は無効となります。デフォルト値は0です。

単位を指定しない場合はKBとして扱われます。

ー 最小値:8KB

ー 最大値:4バイト符号付き整数の最大値

パラメータ設定のための計算方法は、pgx_catalog_cache_max_sizeの計算方法を参照してください。

- pgx_cache_hit_log_interval(数値)

  バックエンドプロセスごとのキャッシュ参照状況を表すメッセージを出力する時間間隔を指定します。

  トランザクションが終了した時に、前にメッセージを出力してから、本パラメータに設定した時間を経過していたならば、メッセージを出力します。

  0を設定するとトランザクションが終了するたびにメッセージを出力します。

  -1を設定すると出力は無効となります。デフォルト値は10minです。

  単位を指定しない場合はmsとして扱われます。

  pgx_catalog_cache_max_size、pgx_relation_cache_max_sizeが無効に設定されている場合も、対応するキャッシュのメッセージ出力は無効となります。

  サーバーに接続した直後には、ユーザー認証などのために、ユーザーアプリケーションからのリクエストの前に、小さなトランザクションが発生します。これらについてのヒット率を知ることには意味がないので、サーバーに接続してから、本パラメータに設定した時間が経過した後に開始したトランザクションの終了時にメッセージを出力します。

  同じ理由で、0のような小さい値を設定すると、このような小さなトランザクションの終了に伴ってメッセージが出力されることがあります。

  メッセージがどのトランザクションに対応しているかは、先頭に出力される情報から確認できます。

  この情報は、パラメータlog_line_prefixの設定により変わります。

  ー 最小値:0

  ー 最大値:2147483647ms

参照

上記以外のpostgresql.confのパラメータについては、“PostgreSQL Documentation”の“Server Administration”の“Server Configuration”を参照してください。

付録B　システム管理関数

Fujitsu Enterprise Postgresのシステム管理関数について説明します。

参照

その他のシステム管理関数の詳細は、“PostgreSQL Documentation”の“The SQL Language”の“System Administration Functions”を参照してください。

B.1　WAL二重化制御関数

以下の表は、WAL二重化に基づくバックアップ／リカバリに使用できる関数を示しています。

表B.1 WAL二重化制御関数

名前	戻り型	説明
pgx_pause_wal_multiplexing()	void	WAL多重化を停止する
pgx_resume_wal_multiplexing()	void	WAL多重化を再開する
pgx_is_wal_multiplexing_paused()	boolean	WALの多重化が停止されていれば真を返す

WALの多重化が構成されていない場合、これらの関数はエラーを返却します。postgresql.confのbackup_destinationパラメータを設定すると、WALの多重化が構成されます。

これらの関数は、スーパーユーザーのみ実行可能です。

B.2　透過的データ暗号化制御関数

以下の表は、透過的データ暗号化に使用できる関数を示しています。

表B.2 透過的データ暗号化制御関数

名前	戻り型	説明
pgx_open_keystore(passphrase text) pgx_open_keystore(sslpassphrase => text)	void	キーストアをオープンする
pgx_set_master_key(passphrase text)	void	マスタ暗号化キーを設定する
pgx_declare_external_master_key(kms_name => text, key_id => text, sslpassphrase => text)	void	鍵管理システムに存在する暗号化キーを透過的データ暗号化のマスタ暗号化キーとして設定する
pgx_set_keystore_passphrase(old_passphrase text, new_passphrase text)	void	キーストアのパスフレーズを変更する

B.2.1　pgx_open_keystore

pgx_open_keystoreは、キーストアをオープンします。

この関数は、スーパーユーザーのみ実行可能です。また、トランザクションブロック内でこの関数を実行することはできません。

ファイルベースのキーストアの場合

pgx_open_keystoreは、指定したパスフレーズを使ってキーストアをオープンします。キーストアをオープンすると、マスタ暗号化キーがデータベースサーバのメモリにロードされます。これにより、暗号化データへのアクセスや暗号化テーブル空間を作成できます。キーストアがすでにオープンしている場合、この関数はエラーを返却します。

鍵管理システムをキーストアとして使用する場合

pgx_open_keystoreは、既に使用することを宣言された鍵管理システム上のマスタ暗号化キーを利用可能な状態にします(キーストアをオープンします)。マスタ暗号化キーを使用すると宣言されていない場合、キーストアはオープンできません。

キーストアがすでにオープンしている場合、入力された認証情報を利用して再度、鍵管理システムに接続します。

鍵管理システムに接続するための認証情報を指定します。引数は名前付け表記で指定してください。引数に渡す情報は使用する鍵管理システムによって異なります。

鍵管理システム情報ファイルに難読化された資格情報ファイルが指定されている場合、そのファイルは新しい資格情報で再作成されます。

タイプkmipの鍵管理システムを使用する場合

以下の引数を名前付け表記で指定します。

- sslpassphrase text

  KMIPサーバに接続する際のクライアント証明書用秘密鍵ファイルのパスフレーズを指定します。秘密鍵ファイルにパスフレーズが設定されていない場合は省略できます。

タイプcustomの鍵管理システムを使用する場合

以下の引数を名前付け表記で指定します。

- kms_secret text

  プラグインに渡される秘密にすべき情報です。鍵管理システムの利用に不要な場合は省略できます。省略できるかはプラグインの実装に依存します。

実行例

クライアント証明書秘密鍵ファイルのパスフレーズmykmippassphraseを名前付け表記で指定する場合

```
SELECT pgx_open_keystore( sslpassphrase => 'mykmippassphrase' );
```

B.2.2 pgx_set_master_key

pgx_set_master_keyは、マスタ暗号化キーを生成してファイルベースのキーストアに格納します。

キーストアがまだ存在しない場合、キーストアを作成します。すでにキーストアが存在する場合、マスタ暗号化キーを変更します。もしキーストアがオープンされていなければ、キーストアがオープンされます。

パスフレーズは8～200バイトの文字列です。

この関数は、スーパーユーザーのみ実行可能です。また、トランザクションブロック内でこの関数を実行することはできません。キーストアはオープンしていてもオープンしていなくてもかまいません。

B.2.3 pgx_declare_external_master_key

pgx_declare_external_master_keyは、鍵管理システムに存在する暗号化キーを透過的データ暗号化のマスタ暗号化キーとして使用することを宣言します。すでにマスタ暗号化キーが存在する場合、マスタ暗号化キーを変更します。すでにマスタ暗号化キーが存在する場合、キーストアがオープンされている必要があります。

引数には、マスタ暗号化キーを特定するための情報を指定します。引数は名前付け表記で指定してください。引数に渡す情報は使用する鍵管理システムによって異なります。

この関数は、スーパーユーザーのみ実行可能です。また、トランザクションブロック内でこの関数を実行することはできません。

この関数は、拡張モジュール'tde_kms'をインストールした場合に利用可能です。

以下の引数を名前付け表記で指定します。

- kms_name text

  鍵管理システム接続情報ファイルに指定した鍵管理システム名を指定します。省略できません。

- key_id text

  暗号化キーに付けられた鍵IDを指定します。省略できません。

- sslpassphrase text

  KMIPサーバに接続する際のクライアント証明書用秘密鍵ファイルのパスフレーズを指定します。秘密鍵ファイルにパスフレーズが設定されていない場合は省略できます。kms_nameで指定される鍵管理システムのタイプがkmipでない場合は無視されます。

- kms_secret text

  プラグインに渡される秘密にすべき情報です。鍵管理システムの利用に不要な場合は省略できます。省略できるかはプラグインの実装に依存します。kms_nameで指定される鍵管理システムのタイプがcustomでない場合は無視されます。

例

```
SELECT pgx_declare_external_master_key( kms_name => 'mykmipsvr', key_id => 'a0eebc99-9c0b-0000-0000-000000000000',
sslpassphrase => 'mykmippassphrase' );
```

B.2.4 pgx_set_keystore_passphrase

pgx_set_keystore_passphraseはファイルベースのキーストアのパスフレーズを変更します。

old_passphraseには現在のパスフレーズを、new_passphraseには新しいパスフレーズを指定します。

パスフレーズは8～200バイトの文字列です。

この関数は、スーパーユーザーのみ実行可能です。また、トランザクションブロック内でこの関数を実行することはできません。キーストアはオープンしていてもオープンしていなくてもかまいません。

B.3 データ秘匿化機能制御関数

以下の表は、データ秘匿化機能に使用できる関数を示しています。

表B.3 データ秘匿化機能制御関数

名前	戻り型	説明
pgx_alter_confidential_policy	boolean	秘匿化ポリシーを変更する
pgx_create_confidential_policy	boolean	秘匿化ポリシーを作成する
pgx_drop_confidential_policy	boolean	秘匿化ポリシーを削除する
pgx_enable_confidential_policy	boolean	秘匿化ポリシーを有効／無効にする
pgx_update_confidential_values	boolean	秘匿化種別に全秘匿化を指定した場合の改訂後の文字を変更する

B.3.1 pgx_alter_confidential_policy

機能

秘匿化ポリシーを変更します。

書式

変更内容によって書式が異なります。書式は以下のとおりです。

・ 共通の書式

```
common_arg:
[schema_name    := 'schema_name',]
table_name    := 'table_name',
policy_name     := 'policy_name'
```

・ 秘匿化ポリシーに秘匿化対象を追加

```
pgx_alter_confidential_policy(
common_arg,
[action        := 'ADD_COLUMN', ]
column_name    := 'column_name'
[, function_type    := 'FULL'] |
[, function_type    := 'PARTIAL', partial_opt] |
[, function_type    := 'REGEXP', regexp_opt]
)


partial_opt:
function_parameters    := 'masking_format'


regexp_opt:
regexp_pattern        := 'regexp_pattern',
regexp_replacement        := 'regexp_replacement',
[, regexp_flags        := 'regexp_flags']
```

・ 秘匿化ポリシーから秘匿化対象を削除

```
pgx_alter_confidential_policy(
common_arg,
action        := 'DROP_COLUMN',
column_name    := 'column_name'
)
```

・ 秘匿化条件を変更

```
pgx_alter_confidential_policy(
common_arg,
action        := 'MODIFY_EXPRESSION',
expression    := 'expression'
)
```

・ 秘匿化対象に設定されている秘匿化ポリシーの内容を変更

```
pgx_alter_confidential_policy(
common_arg,
action        := 'MODIFY_COLUMN',
column_name    := 'column_name'
[, function_type    := 'FULL'] |
[, function_type    := 'PARTIAL', partial_opt] |
[, function_type    := 'REGEXP', regexp_opt]
)


partial_opt:
function_parameters    := 'masking_format'


regexp_opt:
regexp_pattern        := 'regexp_pattern',
regexp_replacement        := 'regexp_replacement',
[, regexp_flags        := 'regexp_flags']
```

- 秘匿化ポリシーの説明を変更

```
pgx_alter_confidential_policy(
common_arg,
action        := 'SET_POLICY_DESCRIPTION',
policy_description    := 'policy_description'
)
```

- 秘匿化対象の説明を変更

```
pgx_alter_confidential_policy(
common_arg,
action        := 'SET_COLUMN_DESCRIPTION',
column_name    := 'column_name',
column_description    := 'column_description'
)
```

引数

変更内容によって引数が異なります。詳細は以下のとおりです。

- 共通の引数

引数を指定可能な秘匿化種別	引数	データ型	意味	デフォルト値
すべて	schema_name	varchar(63)	秘匿化ポリシーが適用されているテーブルのスキーマ名	'public'
	table_name	varchar(63)	秘匿化ポリシーが適用されているテーブル名	省略不可
	policy_name	varchar(63)	秘匿化ポリシー名	省略不可

- 秘匿化ポリシーに秘匿化対象を追加

引数を指定可能な秘匿化種別	引数	データ型	意味	デフォルト値
すべて	action	varchar(63)	'ADD_COLUMN'	'ADD_COLUMN'
	column_name	varchar(63)	秘匿化対象名	省略不可
	function_type	varchar(63)	秘匿化種別 ・ 'FULL':全秘匿化 ・ 'PARTIAL':部分秘匿化 ・ 'REGEXP':正規表現秘匿化	'FULL'
部分秘匿化	function_parameters	varchar(1024)	部分秘匿化の秘匿化形式	省略不可
正規表現秘匿化	regexp_pattern	varchar(1024)	正規表現による改訂方法	省略不可
	regexp_replacement	varchar(1024)	正規表現の改訂後の文字	省略不可
	regexp_flags	varchar(20)	正規表現のフラグ	NULL

- 秘匿化ポリシーから秘匿化対象を削除

引数を指定可能な秘匿化種別	引数	データ型	意味	デフォルト値
すべて	action	varchar(63)	'DROP_COLUMN'	省略不可
	column_name	varchar(63)	秘匿化対象名	省略不可

・ 秘匿化条件を変更

引数を指定可能な秘匿化種別	引数	データ型	意味	デフォルト値
すべて	action	varchar(63)	'MODIFY_EXPRESSION'	省略不可
	expression	varchar(1024)	変更する秘匿化条件	省略不可

・ 秘匿化対象に設定されている秘匿化ポリシーの内容を変更

引数を指定可能な秘匿化種別	引数	データ型	意味	デフォルト値
すべて	action	varchar(63)	'MODIFY_COLUMN'	省略不可
	column_name	varchar(63)	秘匿化対象名	省略不可
	function_type	varchar(63)	秘匿化種別 ・ 'FULL':全秘匿化 ・ 'PARTIAL':部分秘匿化 ・ 'REGEXP':正規表現秘匿化	'FULL'
部分秘匿化	function_parameters	varchar(1024)	部分秘匿化の秘匿化形式	省略不可
正規表現秘匿化	regexp_pattern	varchar(1024)	正規表現による改訂方法	省略不可
	regexp_replacement	varchar(1024)	正規表現の改訂後の文字	省略不可
	regexp_flags	varchar(20)	正規表現のフラグ	NULL

・ 秘匿化ポリシーの説明を変更

引数を指定可能な秘匿化種別	引数	データ型	意味	デフォルト値
すべて	action	varchar(63)	'SET_POLICY_DESCRIPTION'	省略不可
	policy_description	varchar(1024)	秘匿化ポリシーの説明	省略不可

・ 秘匿化対象の説明を変更

引数を指定可能な秘匿化種別	引数	データ型	意味	デフォルト値
すべて	action	varchar(63)	'SET_COLUMN_DESCRIPTION'	省略不可
	column_name	varchar(63)	秘匿化対象名	省略不可
	column_description	varchar(1024)	秘匿化対象の説明	省略不可

引数の省略可否は以下のとおりです。

引数	省略可否									
	ADD_COLUMN			DROP_COLUMN	MODIFY_EXPRESSION	MODIFY_COLUMN			SET_POLICY_DESCRIPTION	SET_COLUMN_DESCRIPTION
	全秘匿化	部分秘匿化	正規表現秘匿化			全秘匿化	部分秘匿化	正規表現秘匿化		
schema_name	○	○	○	○	○	○	○	○	○	○
table_name	×	×	×	×	×	×	×	×	×	×
policy_name	×	×	×	×	×	×	×	×	×	×

引数	省略可否									
	ADD_COLUMN			DROP_COLUMN	MODIFY_EXPRESSION	MODIFY_COLUMN			SET_POLICY_DESCRIPTION	SET_COLUMN_DESCRIPTION
	全秘匿化	部分秘匿化	正規表現秘匿化			全秘匿化	部分秘匿化	正規表現秘匿化		
action	○	○	○	×	×	×	×	×	×	×
column_name	×	×	×	×	-	×	×	×	-	×
function_type	○	×	×	-	-	○	×	×	-	-
expression	-	-	-	-	×	-	-	-	-	-
policy_description	-	-	-	-	-	-	-	-	×	-
column_description	-	-	-	-	-	-	-	-	-	×
function_parameters	-	×	-	-	-	-	×	-	-	-
regexp_pattern	-	-	×	-	-	-	-	×	-	-
regexp_replacement	-	-	×	-	-	-	-	×	-	-
regexp_flags	-	-	○	-	-	-	-	○	-	-

○:省略可 ×:省略不可 -:指定した場合は無視されます

戻り値

戻り値	意味
TRUE	正常終了
FALSE	異常終了

実行例1

秘匿化ポリシーp1に秘匿化対象c2を追加する場合

```
postgres=# select pgx_alter_confidential_policy(table_name := 't1', policy_name := 'p1', action := 'ADD_COLUMN',
column_name := 'c2', function_type := 'PARTIAL', function_parameters := 'VVVFVVVVFVVVV, VVV-VVVV-VVVV, *, 4, 11');
 pgx_alter_confidential_policy
-------------------------------
 t
(1 row)
```

実行例2

秘匿化ポリシーp1から秘匿化対象c1を削除する場合

```
postgres=# select pgx_alter_confidential_policy(table_name := 't1', policy_name := 'p1', action := 'DROP_COLUMN',
column_name := 'c1');
 pgx_alter_confidential_policy
-------------------------------
 t
(1 row)
```

実行例3

秘匿化ポリシーp1に対し、秘匿化条件を変更する場合

```
postgres=# select pgx_alter_confidential_policy(table_name := 't1', policy_name := 'p1', action := 'MODIFY_EXPRESSION',
expression := 'false');
```

```
 pgx_alter_confidential_policy
--------------------------------
 t
(1 row)
```

実行例4

秘匿化対象c2に設定されている秘匿化ポリシーp1の内容を変更する場合

```
postgres=# select pgx_alter_confidential_policy(table_name := 't1', policy_name := 'p1', action := 'MODIFY_COLUMN',
column_name := 'c2', function_type := 'FULL');
 pgx_alter_confidential_policy
--------------------------------
 t
(1 row)
```

実行例5

秘匿化ポリシーp1の説明を変更する場合

```
postgres=# select pgx_alter_confidential_policy(table_name := 't1', policy_name := 'p1', action :=
'SET_POLICY_DESCRIPTION', policy_description := 'this policy is an example.');
 pgx_alter_confidential_policy
--------------------------------
 t
(1 row)
```

実行例6

秘匿化対象c2の説明を変更する場合

```
postgres=# select pgx_alter_confidential_policy(table_name := 't1', policy_name := 'p1', action :=
'SET_COLUMN_DESCRIPTION', column_name := 'c2', column_description := 'c2 column is FULL.');
 pgx_alter_confidential_policy
--------------------------------
 t
(1 row)
```

説明

- pgx_alter_confidential_policyシステム管理関数の引数の順序は任意に指定可能です。
- actionのパラメーターは以下を指定できます。actionパラメーターを省略した場合は、ADD_COLUMNが適用されます。

パラメーター	説明
ADD_COLUMN	秘匿化ポリシーに秘匿化対象を追加します。
DROP_COLUMN	秘匿化ポリシーから秘匿化対象を削除します。
MODIFY_EXPRESSION	expressionを変更します。
MODIFY_COLUMN	秘匿化対象に設定されている秘匿化ポリシーの内容を変更します。
SET_POLICY_DESCRIPTION	policy_descriptionを変更します。
SET_COLUMN_DESCRIPTION	column_descriptionを変更します。

- function_parametersはfunction_typeがPARTIALの場合に有効です。function_typeがPARTIAL以外の場合は無視されます。
- 以下の値はfunction_typeがREGEXPの場合に有効です。function_typeがREGEXP以外の場合は無視されます。
  - regexp_pattern
  - regexp_replacement
  - regexp_flags

参照

- 引数に指定する文字列は、“PostgreSQL Documentation”の“String Constants”を参照してください。
- regexp_pattern、regexp_replacement、regexp_flagsに指定できる値は、“PostgreSQL Documentation”の“POSIX Regular Expressions”のpattern、replacement、flagsを参照してください。

B.3.2 pgx_create_confidential_policy

機能

秘匿化ポリシーを作成します。

書式

秘匿化種別によって書式が異なります。書式は以下のとおりです。

```
pgx_create_confidential_policy(
[schema_name        := 'schema_name',]
table_name          := 'table_name',
policy_name         := 'policy_name',
expression          := 'expression'
[, enable             := 'policy_status']
[, policy_description    := 'policy_description']
[, column_name        := 'column_name'
     [, function_type    := 'FULL'] |
     [, function_type    := 'PARTIAL', partial_opt] |
     [, function_type    := 'REGEXP', regexp_opt]
     [, column_description    := 'column_description']
])


partial_opt:
function_parameters    := 'masking_format'


regexp_opt:
regexp_pattern        := 'regexp_pattern',
regexp_replacement        := 'regexp_replacement',
[, regexp_flags        := 'regexp_flags']
```

引数

詳細は以下のとおりです。

引数を指定可能な秘匿化種別	引数	データ型	意味	デフォルト値
すべて	schema_name	varchar(63)	秘匿化ポリシーを作成するテーブルのスキーマ名	'public'
	table_name	varchar(63)	秘匿化ポリシーを作成するテーブル名	省略不可
	policy_name	varchar(63)	秘匿化ポリシー名	省略不可
	expression	varchar(1024)	秘匿化条件	省略不可
	enable	boolean	秘匿化ポリシーの状態 ・ 't':有効 ・ 'f':無効	't'
	policy_description	varchar(1024)	秘匿化ポリシーに関する説明	NULL

引数を指定可能な秘匿化種別	引数	データ型	意味	デフォルト値
	column_name	varchar(63)	秘匿化対象名	NULL
	function_type	varchar(63)	秘匿化種別 ・ 'FULL':全秘匿化 ・ 'PARTIAL':部分秘匿化 ・ 'REGEXP':正規表現秘匿化	'FULL'
	column_description	varchar(1024)	秘匿化対象に関する説明	NULL
部分秘匿化	function_parameters	varchar(1024)	部分秘匿化の秘匿化形式	省略不可
正規表現秘匿化	regexp_pattern	varchar(1024)	正規表現による改訂方法	省略不可
	regexp_replacement	varchar(1024)	正規表現の改訂後の文字	省略不可
	regexp_flags	varchar(20)	正規表現のフラグ	NULL

引数の省略可否は以下のとおりです。

引数	省略可否		
	全秘匿化	部分秘匿化	正規表現秘匿化
schema_name	○	○	○
table_name	×	×	×
policy_name	×	×	×
expression	×	×	×
enable	○	○	○
policy_description	○	○	○
column_name	○	○	○
function_type	○	○	○
column_description	○	○	○
function_parameters	-	×	-
regexp_pattern	-	-	×
regexp_replacement	-	-	×
regexp_flags	-	-	○

○:省略可 ×:省略不可 -:指定した場合は無視されます

戻り値

戻り値	意味
TRUE	正常終了
FALSE	異常終了

実行例1

秘匿化対象が含まれない秘匿化ポリシーp1を作成する場合

```
postgres=# select pgx_create_confidential_policy(table_name := 't1', policy_name := 'p1', expression := '1=1');
 pgx_create_confidential_policy
--------------------------------
```

```
 t
(1 row)
```

実行例2

秘匿化種別が全秘匿化である秘匿化対象c1を含む秘匿化ポリシーp1を作成する場合

```
postgres=# select pgx_create_confidential_policy(schema_name := 'public', table_name := 't1', policy_name := 'p1',
expression := '1=1', enable := 't', policy_description := 'this policy is an example.', column_name := 'c1',
function_type := 'FULL', column_description := 'c1 column is FULL.');
 pgx_create_confidential_policy
--------------------------------
 t
(1 row)
```

実行例3

秘匿化種別が部分秘匿化である秘匿化対象c2を含む秘匿化ポリシーp1を作成する場合

```
postgres=# select pgx_create_confidential_policy( table_name := 't1', policy_name := 'p1', expression := '1=1',
column_name := 'c2', function_type := 'PARTIAL', function_parameters := 'VVVFVVVVFVVVV, VVV-VVVV-VVVV, *, 4, 11');
 pgx_create_confidential_policy
--------------------------------
 t
(1 row)
```

実行例4

秘匿化種別が正規表現秘匿化である秘匿化対象c3を含む秘匿化ポリシーp1を作成する場合

```
postgres=# select pgx_create_confidential_policy( table_name := 't1', policy_name := 'p1', expression := '1=1',
column_name := 'c3', function_type := 'REGEXP', regexp_pattern := '(.*)(@.*)', regexp_replacement := 'xxx¥2',
regexp_flags := 'g');
 pgx_create_confidential_policy
--------------------------------
 t
(1 row)
```

説明

- pgx_create_confidential_policyシステム管理関数の引数の順序は任意に指定可能です。
- column_nameを省略した場合、秘匿化対象が含まれない秘匿化ポリシーのみが作成されます。
- 各テーブルには1つの秘匿化ポリシーを作成可能です。秘匿化ポリシーに秘匿化対象を追加する場合は、pgx_alter_confidential_policyシステム管理関数で秘匿化対象の追加を行ってください。
- function_parametersはfunction_typeがPARTIALの場合に有効です。function_typeがPARTIAL以外の場合は無視されます。
- 以下の値はfunction_typeがREGEXPの場合に有効です。function_typeがREGEXP以外の場合は無視されます。
  - regexp_pattern
  - regexp_replacement
  - regexp_flags

秘匿化ポリシーを適用するテーブルを削除した場合、秘匿化ポリシーも削除してください

参照

- 引数に指定する文字列は、“PostgreSQL Documentation”の“String Constants”を参照してください。
- regexp_pattern、regexp_replacement、regexp_flagsに指定できる値は、“PostgreSQL Documentation”の“POSIX Regular Expressions”のpattern、replacement、flagsを参照してください。

B.3.3 pgx_drop_confidential_policy

機能

秘匿化ポリシーを削除します。

書式

```
pgx_drop_confidential_policy(
[schema_name    := 'schema_name', ]
table_name      := 'table_name',
policy_name     := 'policy_name'
)
```

引数

詳細は以下のとおりです。

引数	データ型	意味	デフォルト値
schema_name	varchar(63)	秘匿化ポリシーを削除するテーブルのスキーマ名	'public'
table_name	varchar(63)	秘匿化ポリシーを削除するテーブル名	省略不可
policy_name	varchar(63)	秘匿化ポリシー名	省略不可

引数の省略可否は以下のとおりです。

引数	省略可否
schema_name	○
table_name	×
policy_name	×

○:省略可 ×:省略不可

戻り値

戻り値	意味
TRUE	正常終了
FALSE	異常終了

実行例

秘匿化ポリシーp1を削除する場合

```
postgres=# select pgx_drop_confidential_policy(table_name := 't1', policy_name := 'p1');
 pgx_drop_confidential_policy
------------------------------
 t
(1 row)
```

説明

pgx_drop_confidential_policyシステム管理関数の引数の順序は任意に指定可能です。

注意

秘匿化ポリシーを適用するテーブルを削除した場合、秘匿化ポリシーも削除してください

参照

引数に指定する文字列は、“PostgreSQL Documentation”の“String Constants”を参照してください。

B.3.4 pgx_enable_confidential_policy

機能

秘匿化ポリシーを有効／無効にします。

書式

```
pgx_enable_confidential_policy(
[schema_name     := 'schema_name', ]
table_name       := 'table_name',
policy_name      := 'policy_name',
enable           := 'policy_status'
)
```

引数

詳細は以下のとおりです。

引数	データ型	意味	デフォルト値
schema_name	varchar(63)	秘匿化ポリシーを有効／無効にするテーブルのスキーマ名	'public'
table_name	varchar(63)	秘匿化ポリシーを有効／無効にするテーブル名	省略不可
policy_name	varchar(63)	秘匿化ポリシー名	省略不可
enable	boolean	秘匿化ポリシーの状態 ・ 't':有効 ・ 'f':無効	省略不可

引数の省略可否は以下のとおりです。

引数	省略可否
schema_name	○
table_name	×
policy_name	×
enable	×

○:省略可　×:省略不可

戻り値

戻り値	意味
TRUE	正常終了
FALSE	異常終了

実行例

秘匿化ポリシーp1を有効にする場合

```
postgres=# select pgx_enable_confidential_policy(table_name := 't1', policy_name := 'p1', enable := 't');
 pgx_enable_confidential_policy
--------------------------------
 t
(1 row)
```

説明

pgx_enable_confidential_policyシステム管理関数の引数の順序は任意に指定可能です。

参照

引数に指定する文字列は、“PostgreSQL Documentation”の“String Constants”を参照してください。

B.3.5 pgx_update_confidential_values

機能

秘匿化種別に全秘匿化を指定した場合の改訂後の文字を変更します。

書式

```
pgx_update_confidential_values(
[number_value    := 'number_value']
[, char_value    := 'char_value']
[, varchar_value    := 'varchar_value']
[, date_value    := 'date_value']
[, ts_value    := 'ts_value']
)
```

引数

詳細は以下のとおりです。

引数	データ型	意味
number_value	integer	数値型における改訂後の文字
char_value	varchar(1)	char型における改訂後の文字
varchar_value	varchar(1)	varchar型における改訂後の文字
date_value	date	date型における改訂後の文字
ts_value	timestamp	timestamp型における改訂後の文字

戻り値

戻り値	意味
TRUE	正常終了
FALSE	異常終了

実行例

char型、およびvarchar型の改訂後の文字を'*'にする場合

```
postgres=# select pgx_update_confidential_values(char_value := '*', varchar_value := '*');
 pgx_update_confidential_values
--------------------------------
 t
(1 row)
```

説明

- pgx_update_confidential_valuesシステム管理関数の引数の順序は任意に指定可能です。
- pgx_update_confidential_valuesシステム管理関数には1つ以上の引数を指定してください。省略した引数については、改訂後の文字は変更されません。

参照

引数に指定する文字列は、“PostgreSQL Documentation”の“String Constants”を参照してください。

B.4 VCIデータのロード制御関数

以下の表は、VCIのデータをバッファキャッシュ上にロードするために使用する関数を示しています。

表B.4 VCIデータのロード制御関数

名前	戻り型	説明
pgx_prewarm_vci(vci_index regclass)	int8	VCIデータをバッファキャッシュ上にロードする

pgx_prewarm_vciは、指定したVCIのデータをバッファキャッシュ上にロードし、ロードしたVCIデータのブロック数を返却します。

インスタンス起動直後は、VCIデータがバッファキャッシュ上に読み込まれていないため、VCIを使用した集計処理に時間が掛かる場合があります。そのため、インスタンス起動後にpgx_prewarm_vciを実行することにより、1回目の集計処理を高速化することができます。

また、pgx_prewarm_vciが返却するブロック数と1ブロックのサイズを掛け合わせて使用することで、事前ロードで使用するメモリ使用量を確認することができます。

VCIインデックスに対する参照権限、およびpg_prewarm関数の実行権限がない場合は、本関数を実行できません。

B.5 高速データロード制御関数

以下の表は、高速データロード機能に使用できる関数を示しています。

表B.5 高速データロード制御関数

名前	戻り型	説明
pgx_loader	bigint	動的共有メモリの作成および並列ワーカを起動して、データをロードする
pgx_loader_recovery	smallint	インダウトトランザクションを解決する

pgx_loaderコマンドは内部的に上記の関数を実行します。

付録C　システムビュー

Fujitsu Enterprise Postgresのシステムビューについて説明します。

参照

その他のシステムビューの詳細は、“PostgreSQL Documentation”の“Internals”の“System Views”を参照してください。

C.1　pgx_tablespaces

pgx_tablespacesビューはテーブル空間の暗号化に関する情報を提供します。

表C.1 pgx_tablespacesビュー

列	型	参照先	説明
spctablespace	oid	pg_tablespace.oid	テーブル空間のOID
spcencalgo	text		テーブル空間の暗号化アルゴリズム

spcencalgo列は次のいずれかの値を示します。

- none　:テーブル空間は暗号化されていません
- AES128 :キー長128ビットのAES
- AES256 :キー長256ビットのAES

C.2　pgx_stat_lwlock

pgx_stat_lwlockビューは軽量ロックの内容ごとに1行の形で、発生に関する統計情報を示します。

表C.2 pgx_stat_lwlockビュー

列	型	説明
lwlock_name	name	軽量ロックの名前です。
total_waits	bigint	軽量ロックが原因で待ちが発生した回数です。
total_wait_time	double precision	軽量ロックが原因で待ち期間として費やされた、ミリ秒単位の総時間です。
stats_reset	timestamp with timezone	これらの統計情報がリセットされた最終時刻です。

C.3　pgx_stat_latch

pgx_stat_latchビューはFujitsu Enterprise Postgres内部の待機情報について、内容ごとに1行の形で、発生に関する統計情報を示します。

表C.3 pgx_stat_latchビュー

列	型	説明
latch_name	name	ラッチの名前です。
total_waits	bigint	ラッチが原因で待ちが発生した回数です。
total_wait_time	double precision	ラッチが原因で待ち期間として費やされた、ミリ秒単位の総時間です。
stats_reset	timestamp with timezone	これらの統計情報がリセットされた最終時刻です。

C.4 pgx_stat_walwriter

pgx_stat_walwriterビューはWALの書込みに関する統計情報を1行のみで表示します。

表C.4 pgx_stat_walwriterビュー

列	型	説明
dirty_writes	bigint	WALレコードの追加時にWALバッファが満杯だったため、古いWALバッファをディスクに書き込んだ回数です。
writes	bigint	WALの書込み回数です。
write_blocks	bigint	WALの書込みブロック数です。
total_write_time	double precision	WALの書込みに費やされた、ミリ秒単位の総時間です。
stats_reset	timestamp with timezone	これらの統計情報がリセットされた最終時刻です。

C.5 pgx_stat_sql

pgx_stat_sqlビューはSQL文の種類ごとに1行の形で、SQL文の発生回数に関する情報を示します。

表C.5 pgx_stat_sqlビュー

列	型	説明
selects	bigint	SELECT文が実行された回数です。 データベース多重化運用を行っている場合、Mirroring ControllerのSELECT文実行回数が加算されます。Mirroring Controllerは、サーバ定義ファイルのheartbeat_intervalに指定した間隔(ミリ秒)でSELECT文を実行します。
inserts	bigint	INSERT文が実行された回数です。
deletes	bigint	DELETE文が実行された回数です。
updates	bigint	UPDATE文が実行された回数です。
selects_with_parallelism	bigint	SELECT文において並列検索が実行された回数です。
inserts_with_parallelism	bigint	未使用です。
deletes_with_parallelism	bigint	未使用です。
updates_with_parallelism	bigint	未使用です。
copies_with_parallelism	bigint	未使用です。
declares	bigint	DECLARE文が実行された回数です。(カーソルのOPEN回数)
fetches	bigint	FETCH文が実行された回数です。
checkpoints	bigint	CHECKPOINT文が実行された回数です。
clusters	bigint	CLUSTER文が実行された回数です。
copies	bigint	COPY文が実行された回数です。
reindexes	bigint	REINDEX文が実行された回数です。
truncates	bigint	TRUNCATE文が実行された回数です。
locks	bigint	ロックが発生した回数です。
stats_reset	timestamp with timezone	これらの統計情報がリセットされた最終時刻です。

C.6 pgx_stat_gmc

pgx_stat_gmcビューはGMC領域に関する情報を提供します。

表C.6 pgx_stat_gmcビュー

列	型	説明
searches	bigint	キャッシュ表を検索した回数です。
hits	bigint	キャッシュ表にヒットした回数です。
size	bigint	GMC領域の現在の使用メモリ量（バイト単位）です。
stats_reset	timestamp with time zone	これらの統計情報がリセットされた最終時刻です。

C.7 pgx_following_async_walsenders_pid

pgx_following_async_walsenders_pidビューは同期スタンバイサーバへのトランザクションログの送信順序を確認する非同期スタンバイサーバのWAL送信プロセスのプロセスIDを提供します。

表C.7 pgx_following_async_walsenders_pidビュー

列	型	参照先	説明
pid	integer	pg_stat_replication.pid	同期スタンバイサーバへのトランザクションログの送信順序を確認する非同期スタンバイサーバのWAL送信プロセスのプロセスIDです。

C.8 pgx_stat_progress_loader

pgx_stat_progress_loaderビューはpgx_loaderコマンドの進捗情報を提供します。

pgx_stat_progress_loaderビューには、pgx_loaderを実行した際に、バックエンドプロセスと並列数分のワーカプロセスの進捗情報の合計が表示されます。

表C.8 pgx_stat_progress_loaderビュー

列	型	説明
pid	integer	バックエンドのプロセスIDです。
datid	Oid	接続対象となるデータベースのOidです。
datname	name	接続対象となるデータベース名です。
relid	Oid	ロード処理の対象となるテーブルのOidです。
command	text	ロード処理を実施するコマンドです。 (pgx_loaderの場合は常に「COPY FROM」になる)
type	text	ロード処理のデータ元のタイプです。
bytes_processed	bigint	ロード完了のデータのサイズです。 (バックエンドとワーカの合計)
bytes_total	bigint	ロードするデータのサイズです。 (バックエンドとワーカの合計)
tuples_processed	bigint	ロード完了の行数です。 (バックエンドとワーカの合計)
tuples_excluded	bigint	ロード処理でスキップされた行数です。 (バックエンドとワーカの合計)

付録D　透過的データ暗号化機能が利用するテーブル

透過的データ暗号化機能が利用するテーブルについて説明します。

D.1 pgx_tde_master_key

鍵管理システムをキーストアとして使用する場合に、使用しているマスタ暗号化キーに関する情報を提供します。

列	型	説明
local_key_id	integer	Fujitsu Enterprise Postgres内部で使用される暗号化キーの通番
kms_name	text	鍵管理システム名
key_name	text	現在使われていません
key_id	text	鍵管理システムにおける鍵ID
start_time	timestamp	鍵が有効化された時刻
status	enum	in-use: 現在使われているマスタ暗号化キー used:過去に使用されていたマスタ暗号化キー scheduled:マスタ暗号化キーの更新が要求され、使用する予定である新しい暗号化キー

実行例

```
postgres=# select * from pgx_tde_master_key;
 local_key_id | kms_name  | key_name|                key_id                |     start_time      | status
--------------+-----------+---------+--------------------------------------+---------------------+---------
            1 | mykmipsvr |         | a0eebc99-9c0b-0000-0000-000000000000 | 2022-12-16 05:43:49| in-use
(1 row)
```

注意

列の順番が変更となったり、列が追加される可能性があるため、選択リストに“*”を指定した問い合わせを使用しないでください。

付録E　データ秘匿化機能が利用するテーブル

データ秘匿化機能が利用するテーブルについて説明します。

注意

これらのテーブルはデータ秘匿化機能制御関数によって更新されるため、データ秘匿化機能が利用するテーブルをSQL文により直接更新しないでください。

E.1　pgx_confidential_columns

秘匿化ポリシーが設定されている秘匿化対象に関する情報を提供します。

列	型	説明
schema_name	varchar(63)	秘匿化ポリシーが適用されているテーブルのスキーマ名
table_name	varchar(63)	秘匿化ポリシーが適用されているテーブル名
policy_name	varchar(63)	秘匿化ポリシー名
column_name	varchar(63)	秘匿化対象名
function_type	varchar(63)	秘匿化種別 ・ 'FULL' : 全秘匿化 ・ 'PARTIAL' : 部分秘匿化 ・ 'REGEXP' : 正規表現秘匿
function_parameters	varchar(1024)	部分秘匿化の秘匿化形式
regexp_pattern	varchar(1024)	正規表現による改訂方法
regexp_replacement	varchar(1024)	正規表現の改訂後の文字
regexp_flags	varchar(20)	正規表現のフラグ
column_description	varchar(1024)	秘匿化対象の説明

実行例

```
postgres=# select * from pgx_confidential_columns;
 schema_name | table_name | policy_name | column_name | function_type |          function_parameters           |
regexp_pattern | regexp_replacement | regexp_flags | column_description
-------------+------------+-------------+-------------+---------------+----------------------------------------
+----------------+--------------------+-------------+--------------------
 public      | t1         | p1          | c1          | FULL          |                                        |
|                |                    |             |
 public      | t1         | p1          | c2          | PARTIAL       | VVVFVVVVFVVVV, VVV-VVVV-VVVV, *, 4, 11 |
|                |                    |             |
(2  row)
```

E.2　pgx_confidential_policies

秘匿化ポリシーの内容に関する情報を提供します。

列	型	説明
schema_name	varchar(63)	秘匿化ポリシーが適用されているテーブルのスキーマ名
table_name	varchar(63)	秘匿化ポリシーが適用されているテーブル名

列	型	説明
policy_name	varchar(63)	秘匿化ポリシー名
expression	varchar(1024)	秘匿化条件
enable	boolean	秘匿化ポリシーの状態 ・ 't' : 有効 ・ 'f' : 無効
policy_description	varchar(1024)	秘匿化ポリシーの説明

実行例

```
postgres=# select * from pgx_confidential_policies;
 schema_name | table_name | policy_name | expression | enable | policy_description
-------------+------------+-------------+------------+--------+--------------------
 public      | t1         | p1          | 1=1        | t      |
(1 row)
```

E.3 pgx_confidential_values

秘匿化種別に全秘匿化を指定した場合の改訂後の文字に関する情報を提供します。

列	型	説明	改訂後の文字のデフォルト値
number_value	integer	数値型のデフォルト値	0
char_value	varchar(1)	char型のデフォルト値	空白
varchar_value	varchar(1)	varchar型のデフォルト値	空白
date_value	date	date型のデフォルト値	'1970-01-01'
timestamp_value	timestamp	timestamp型のデフォルト値	'1970-01-01 00:00:00'

実行例

```
postgres=# select * from pgx_confidential_values;
 number_value | char_value | varchar_value | date_value |      ts_value
--------------+------------+---------------+------------+---------------------
            0 |            |               | 1970-01-01 | 1970-01-01 00:00:00
(1 row)
```

付録F　高速データロード機能が利用するテーブル

高速データロード機能が利用するテーブルについて説明します。

F.1　pgx_loader_state

pgx_loader_stateテーブルは、高速データロード機能が準備したトランザクションに関する情報を提供します。

列	型	説明
id	serial	一意な識別子です。 pgx_loader_state_id_seqシーケンスから割り当てられます。
gid	text	トランザクションに割り当てられたグローバルのトランザクション識別子です。
state	text	トランザクションの状態です。 以下のいずれかの値を示します。 ・ commit: 準備されたトランザクションはコミット済みです。 ・ rollback: 準備されたトランザクションはインダウト状態です。
leader_pid	integer	pgx_loader制御関数を実行したバックエンドプロセス(リーダープロセス)のプロセスIDです。
role_oid	integer	ロール識別子(OID)です。準備されたトランザクションを完了するには、元のトランザクションを実行したユーザーと同じユーザーか、スーパーユーザーでなければなりません。
relation_oid	integer	オブジェクト識別子(OID)です。

注意

pgx_loader_stateテーブルおよびpgx_loader_state_id_seqシーケンスは、高速データロード機能によって更新されます。これらのデータベースオブジェクトをSQL文により直接更新しないでください。

付録G　WebAdminのWebサーバ機能の起動と停止

WebAdminを使用して、Fujitsu Enterprise Postgresのインスタンスを作成・管理するには、Fujitsu Enterprise Postgresをインストールしたサーバで、あらかじめWebAdminのWebサーバ機能を起動する必要があります。

- WebAdminを単一サーバ構成で利用する場合

  対象となるサーバでWebサーバの起動が必要です。

- WebAdminを複数サーバ構成で利用する場合

  WebAdminをインストールしたすべてのサーバでWebサーバの起動が必要です。

WebAdminのWebサーバ機能の起動方法、停止方法について説明します。

なお、“<x>”は、製品のバージョンを示します。

参照

複数サーバ構成のインストールについては、“導入ガイド(サーバ編)”の“複数サーバ構成でのWebAdminのインストール”を参照してください。

G.1　WebAdminのWebサーバ機能の起動

以下の手順でWebAdminのWebサーバ機能を起動してください。

1. スーパーユーザーへの変更

   システム上でスーパーユーザーになります。

   例

   ```
   $ su -
   Password:******
   ```

2. WebAdminのWebサーバ機能の起動

   WebAdminStartコマンドを実行し、WebAdminのWebサーバ機能を起動します。

   例

   WebAdminが、"/opt/fsepv<*x*>webadmin"にインストールされている場合

   ```
   # cd /opt/fsepv<x>webadmin/sbin
   # ./WebAdminStart
   ```

G.2　WebAdminのWebサーバ機能の停止

WebAdminのWebサーバ機能の停止方法について説明します。

以下の手順でWebAdminのWebサーバ機能を停止してください。

1. スーパーユーザーへの変更

   システム上でスーパーユーザーになります。

   例

   ```
   $ su -
   Password:******
   ```

2. WebAdminのWebサーバ機能の停止

WebAdminStopコマンドを実行し、WebAdminのWebサーバ機能を停止します。

例

WebAdminが、"/opt/fsepv<*x*>webadmin"にインストールされている場合

```
# cd /opt/fsepv<x>webadmin/sbin
# ./WebAdminStop
```

付録H　WebAdminウォレット

WebAdminのウォレット機能の使い方について説明します。

リモートインスタンスまたはスタンバイインスタンスを作成する際、認証のためにユーザー名とパスワードをリモートマシンまたはデータベースインスタンスに提供する必要があります。

これらのクレデンシャルを作成し保存するために、WebAdminのウォレット機能を利用すると便利です。

一度作成すると、これらのクレデンシャルを複数のインスタンスで繰り返し使用することができます。

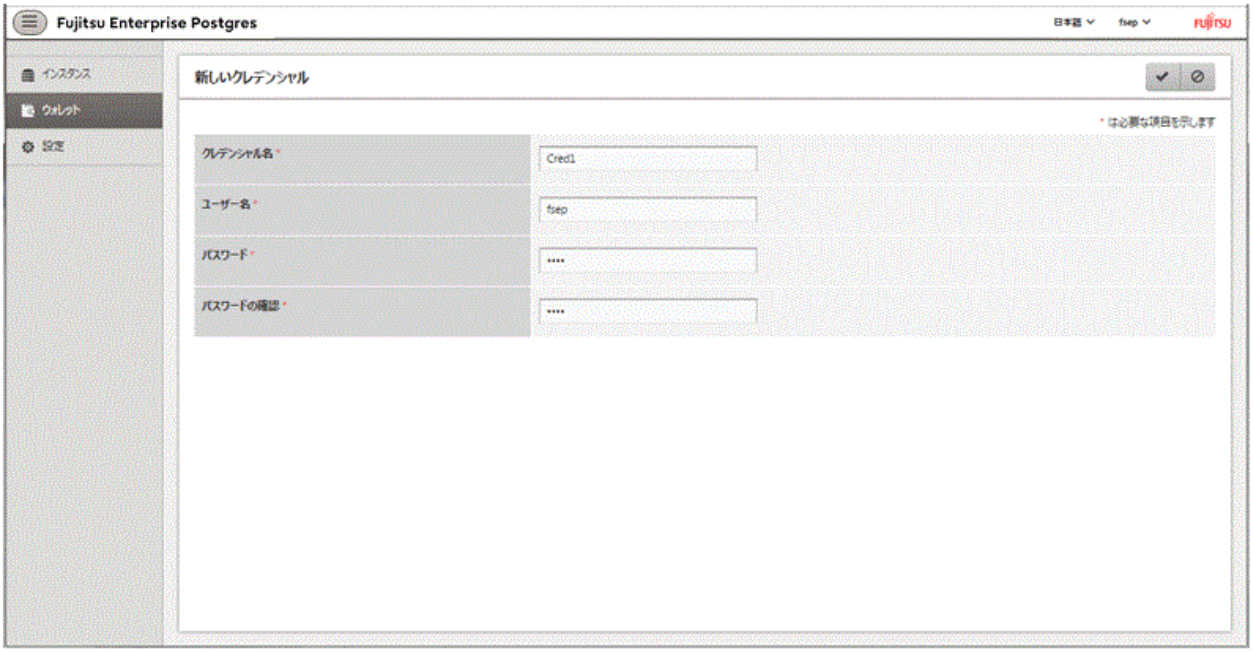
注意

ウォレットでクレデンシャルを作成することは必須ではありません。ウォレットでクレデンシャルを作成しなくても、リモートインスタンスやスタンバイインスタンスを作成することができます。事前にクレデンシャルが作成されていない場合、インスタンス作成ページでユーザー名とパスワードを入力できます。

“リモート”インスタンスを作成する際に、ウォレットに保存されているクレデンシャルを使わずにOSのクレデンシャルを入力すると、WebAdminは入力されたユーザー名とパスワードで自動的にクレデンシャルを作成し、その後使用できるようにユーザーのウォレットに保存します。

H.1　クレデンシャルの作成

1. [ウォレット]タブで、➕アイコンをクリックします。[新しいクレデンシャル]ページが表示されます。
2. クレデンシャル情報を入力します。



以下の項目を入力します。クレデンシャル名、ユーザー名およびパスワードに指定できない文字については、“付録I WebAdminで使用できない文字”を参照してください。

ー [クレデンシャル名]:クレデンシャル名

以下の条件に従って設定してください。

- 最大16文字
- 先頭文字はASCII英字
- 他の文字はASCII英数字

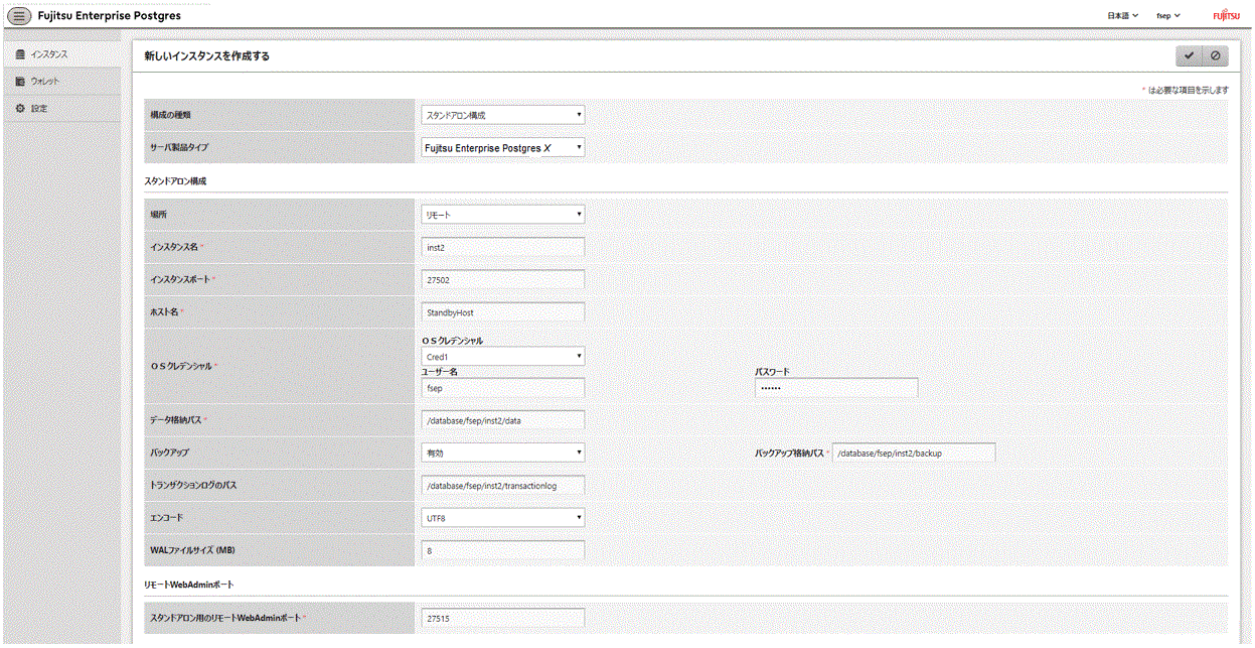
- [ユーザー名]: 後で使用するOSユーザー名またはデータベースインスタンスユーザー名
- [パスワード]: ユーザーのパスワード
- [パスワードの確認]: 再度パスワードを入力してください。

3. ✔をクリックしてクレデンシャルを保存します。

H.2 クレデンシャルの利用

ウォレットにクレデンシャルを作成すると、リモートインスタンスやスタンバイインスタンスの作成時に利用できます。

前節で作成されたクレデンシャルが以下のページで使われています。



[OSクレデンシャル]で'Cred1'を選択すると、ユーザー名とパスワードがクレデンシャルから自動的に入力されます。

付録I　WebAdminで使用できない文字

WebAdminで使用できない文字を以下に示します。

|	&	;	$
%	@	'	"
¥'	¥"	<>	()
+(CR、ASCII 0x0d)	LF(ASCII 0x0a)	,	¥

付録J　障害調査情報の採取

環境構築や運用中に発生したトラブルの原因が判明しない場合、初期調査のための情報を採取します。

初期調査のための情報の採取方法について説明します。

pgx_fjqssinfコマンドを使用して、初期調査のための情報を採取してください。

参照

pgx_fjqssinfコマンドについては、“リファレンス”の“pgx_fjqssinf”を参照してください。

索　引

[ま]

[ら]