FUJITSU

Fujitsu
Enterprise Postgres 15 SP2

セキュリティ運用ガイド

Linux

J2UL-2841-03ZJZ0(00)
2024年1月

まえがき

本書の目的

本書は、Fujitsu Software Enterprise Postgresを使ってシステムを構築/運用する際のセキュリティについて説明しています。

本書の読者

本書は、以下のような読者を対象に書かれています。

- Fujitsu Enterprise Postgresの導入を検討している方
- Fujitsu Enterprise Postgresでデータベースセキュリティ運用環境の設計、構築、運用を行う方
- Fujitsu Enterprise Postgresで構築されたデータベースシステムへアクセスする方

本書を読むためには、以下の知識が必要です。

- 業務に関する知識
- Fujitsu Enterprise Postgresに関する知識
- Linuxに関する一般的な知識

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 セキュリティの概要

セキュリティシステムの概要やFujitsu Enterprise Postgresが提供するセキュリティ機能について説明します。

第2章 セキュリティ運用の概要

セキュリティ運用の概要について説明します。

第3章 責任者の作業

責任者が実施するセキュリティ対策の作業について説明します。

第4章 管理者の作業

管理者が実施するセキュリティ対策の作業について説明します。

第5章 利用者の作業

利用者が実施するセキュリティ対策の作業について説明します。

第6章 監査ログ機能

Fujitsu Enterprise Postgresが提供する監査ログ機能について説明します。

第7章 機密管理支援機能

Fujitsu Enterprise Postgresが提供する機密管理支援機能について説明します。

付録A 機密管理支援機能が利用するテーブル

機密管理支援機能が利用するテーブルについて説明します。

付録B 機密管理支援機能が利用するシステム管理関数

機密管理支援機能が利用するシステム管理関数について説明します。

本書の出典

本ドキュメントは、以下のドキュメントの一部を引用しています。

・“データベースセキュリティガイドライン(2.0版)”
(データベース・セキュリティ・コンソーシアム(DBSC))

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

2024年　1月　第3版
2023年 10月　第2版
2023年　4月　　初版

著作権

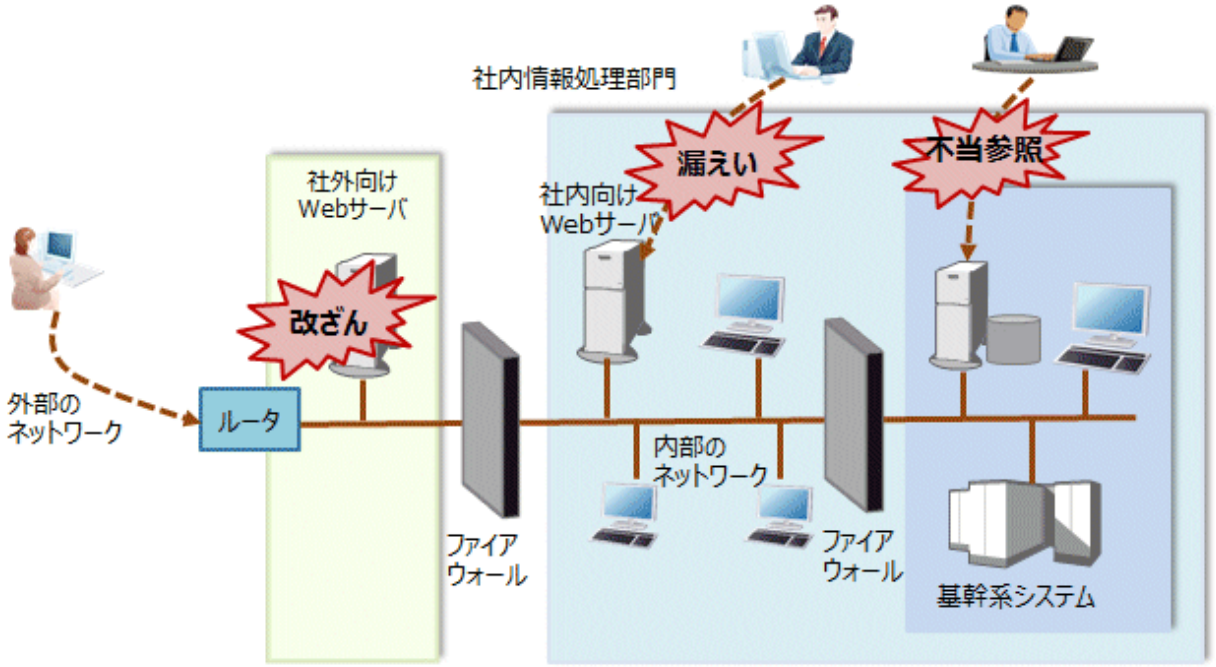
目　次

第1章 セキュリティの概要

1.1 セキュリティとは

セキュリティとは、情報の漏えいや改ざん、外部からの攻撃、進入、盗聴といった危険を排除すること、あるいは情報サービスへの不当な妨害を阻止することです。情報システムが社会基盤としての信頼を得るためには、漏えいや改ざんなどによるセキュリティの脅威を事前に防止するためのセキュリティ対策が必須です。

図1.1 セキュリティの脅威



情報システムにおけるセキュリティ対策は、以下に分類して考えることができます。

- ネットワーク
- Web
- アプリケーション
- データベース
- 端末

本書では、Fujitsu Enterprise Postgres利用時のデータベースセキュリティ対策を中心に説明します。

1.2 セキュリティの要件

情報システムに必要なセキュリティ要件を以下に示します。

セキュリティポリシーの維持

セキュリティポリシーとは、企業が情報資産に対してどのように取り組み、従業員がどのように行動すべきかといった方針を明文化した規範のことです。

セキュリティポリシーを維持しながら情報システムのセキュリティを遂行していく必要があります。

統合的なセキュリティ管理

セキュリティには、以下の側面があります。これらの側面から情報を統合的に管理する必要があります。

機密性

情報のアクセスを限定し、外部への情報漏えいを防止すること

対策例:情報漏えい防止、アクセス権の設定

完全性

情報が破壊されたり改ざんされたりせずに、完全であることを保証すること

対策例:改ざん防止や検出

可用性

障害を防ぎ、正常な運用を維持し、利用したいときに情報が利用できること

対策例:電源対策、システムの二重化

1.3 セキュリティの脅威

セキュリティの脅威とは、情報資産に対し、“1.2 セキュリティの要件”で示した「機密性」、「完全性」、「可用性」を脅かすものと定義します。データベースへのアクセスなど技術的な脅威を対象とし、物理的な破壊は含みません。

脅威は、脅威の発生元である登場人物、守るべき情報資産、手口、不正アクションの組合せで考えます。たとえば、「一般利用者が、データベース管理情報を、データベースの脆弱性を悪用して情報を入手し、情報を改ざんする」などが脅威となります。

セキュリティ対策を考える上で、まず、どのような脅威があるかを明確にする必要があります。ここでは、想定される脅威の一覧を以下に示します。登場人物、守るべき情報資産の定義については、それぞれ、“登場人物について”および“情報資産について”を参照してください。

想定される脅威

登場人物	情報資産	手口	不正アクション
一般利用者 社内利用者 システム責任者 システム開発者 システム管理者 システム運用者	データベース管理情報	パケットの盗聴	情報の不正入手(参照) 情報の不正改ざん、破壊(更新)
		パスワードの辞書攻撃	
		ソーシャルエンジニアリングによるID/パスワードの不正入手	
		設定を悪用した情報の不正入手	
		データベースの脆弱性を悪用した情報の不正入手	
		不正ルートで入手	
一般利用者 社内利用者	データベース一般情報	通常ルートで入手	入手できる情報の悪用(持ち出し)
		業務妨害を狙ったSQL発行	業務妨害(リソース枯渇)
一般利用者 社内利用者	データベース一般情報	パケットの盗聴	情報の不正改ざん、破壊(更新)
		パスワードの辞書攻撃	
		ソーシャルエンジニアリングによるID/パスワードの不正入手	
		設定ミスを悪用した情報の不正入手	
		データベースの脆弱性を悪用した情報の不正入手	
		不正ルートで入手	
システム責任者 システム開発者 システム管理者	データベース一般情報	パケットの盗聴	情報の不正入手(参照) 情報の不正改ざん、破壊(更新)
		パスワードの辞書攻撃	

登場人物	情報資産	手口	不正アクション
システム運用者		ソーシャルエンジニアリングによるID/パスワードの不正入手	
		設定ミスを悪用した情報の不正入手	
		データベースの脆弱性を悪用した情報の不正入手	
		不正ルートで入手	
システム開発者	データベース管理情報	バックドアの作成	情報の不正入手(参照) 情報の不正改ざん、破壊(更新)
	データベース一般情報		
システム責任者 システム管理者	データベース管理情報	不正なデータベース管理者アカウントの作成による情報の不正入手	情報の不正入手(参照) 情報の不正改ざん、破壊(更新)
	データベース一般情報		
システム責任者 システム運用者	データベース管理情報	データベース関連ファイル(定義ファイル、物理ファイルなど)改ざんによる情報の不正入手	情報の不正入手(参照) 情報の不正改ざん、破壊(更新)
	データベース一般情報		
データベース管理者	データベース管理情報	通常ルートで入手後、情報の悪用（持ち出し)	入手できる情報の悪用(持ち出し)
		管理情報からID/パスワードの不正利用	入手できる情報の改ざん、破壊
		管理情報の改ざんによる情報の不正入手	
		業務妨害を狙ったSQL発行	業務妨害(リソース枯渇)
	データベース一般情報	パケットの盗聴	情報の不正入手(参照) 情報の不正改ざん、破壊(更新)
		不正ルートで入手後、情報の悪用(持ち出し)	
データベース運用者	データベース管理情報	パケットの盗聴	情報の不正入手(参照) 情報の不正改ざん、破壊(更新)
		パスワードの辞書攻撃	
		ソーシャルエンジニアリングによるID/パスワードの不正入手	
		設定ミスを悪用した情報の不正入手	
		データベースの脆弱性を悪用した情報の不正入手	
		不正ルートで入手	
	データベース一般情報	通常ルートで入手	入手できる情報の悪用(持ち出し)
		業務妨害を狙ったSQL発行	業務妨害(リソース枯渇)

登場人物について

データベースセキュリティでは、データベースに関わる人物とその役割を以下のように定義します。

登場人物	役割
システム責任者	開発者、管理者、運用者の管理
システム開発者	データベース周辺のネットワーク構築 データベースサーバ構築
システム管理者	データベース周辺のネットワーク機器の運用 データベースサーバの運用
システム運用者	データベース周辺のネットワーク運用
データベース管理者	データベースシステムの構築 データベースシステムの運用
データベース運用者	業務の運用
社内利用者	社内のエンドユーザー
一般利用者	社外のエンドユーザー

情報資産について

データベースセキュリティでは、データベースサーバに格納する情報資産を保護する必要があります。

保護すべき情報資産を以下のように定義します。

データベース管理情報

- データベース構成情報(システムカタログ、ユーザーID/パスワードなど)
- データベースログ(アクセスログなど)

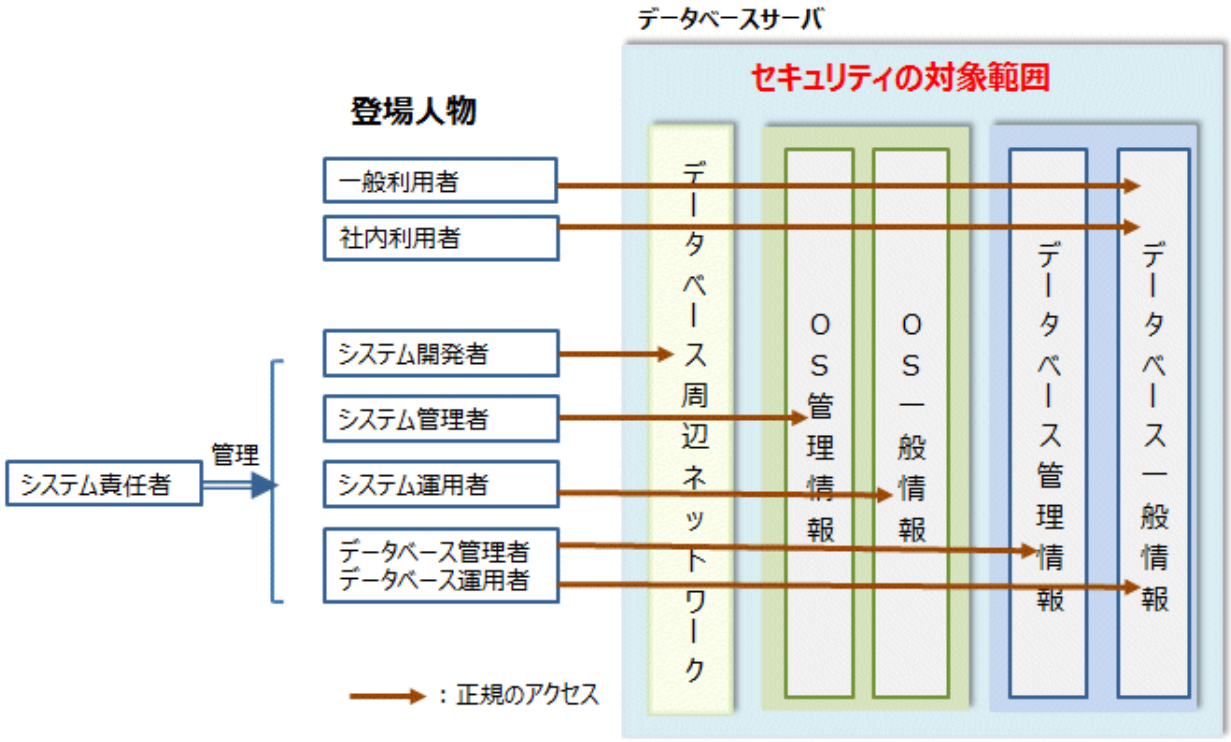
データベース一般情報

- 業務データ
- アプリケーション

1.4 セキュリティの対象範囲

データベースシステムでは、データベースサーバおよびデータベース周辺ネットワークをセキュリティの対象範囲とします。登場人物とそれぞれの登場人物がどの範囲で関わるかを明確にして、登場人物ごとにセキュリティ対策を考える必要があります。

以下に登場人物とセキュリティの対象範囲の関係を示します。



1.5 Fujitsu Enterprise Postgresが提供するセキュリティ

Fujitsu Enterprise Postgresでは、“1.2 セキュリティの要件”で示したセキュリティ要件を満たすセキュリティ機能が提供されています。

Fujitsu Enterprise Postgresが提供するセキュリティについて説明します。

1.5.1 セキュリティの対象者

Fujitsu Enterprise Postgresで構築するシステムでは、セキュリティに関する対象者を“責任者”、“管理者”、“利用者”とします。堅牢なセキュリティシステムを構築するために、それぞれの立場でセキュリティ対策を講じる必要があります。

セキュリティの対象者と“1.3 セキュリティの脅威”で示した登場人物についてのマッピングを下表に示します。

セキュリティの対象者	登場人物
責任者	システム責任者
管理者	システム開発者
	システム管理者
	システム運用者
	データベース管理者
	データベース運用者
利用者	一般利用者
	社内利用者

責任者

セキュリティポリシーを定め、組織全体の運用方針を決定します。

責任者の作業については、“第3章 責任者の作業”を参照してください。

管理者

システムを設計、構築および運用します。その際、管理者は、責任者が定めたセキュリティポリシーに従い、セキュリティ対策を実施する必要があります。

管理者の作業については、“第4章 管理者の作業”を参照してください。

利用者

データベースをアクセスする不特定多数の人であり、管理者以外の人です。データベースシステムに登録されている必要があり、データベースをアクセスする権限によりアクセスを制限されます。

利用者の作業については、“第5章 利用者の作業”を参照してください。

1.5.2 セキュリティ機能

Fujitsu Enterprise Postgresでは、以下のセキュリティ機能が提供されています。

- 認証
- アクセスコントロール
- 暗号化
- 監査ログ
- データ秘匿化

各機能について説明します。

認証

データベースにアクセスするデータベースユーザーの認証を行うことで、アクセス可能なデータベースを制限することができます。また、サーバを認証して、データベースサーバのなりすましを防止することができます。

認証の詳細については、“PostgreSQL Documentation”の“Server Administration”の“Client Authentication”を参照してください。

サーバの認証については、“PostgreSQL Documentation”の“Server Setup and Operation”の“Secure TCP/IP Connections with SSL”を参照してください。

Fujitsu Enterprise Postgresでは、データベースに接続する際のクライアント認証としてパスワード認証を利用する場合、データベース管理者は、事前に定義されたセキュリティポリシーに基づいたパスワードの運用をデータベースの利用者に強制できます。詳細については、“Fujitsu Enterprise Postgres 運用ガイド”の“ポリシーに基づいたパスワードの運用”を参照してください。

アクセスコントロール

データベースのオブジェクトは、初期状態ではオブジェクトの作成者、またはオブジェクト作成時に所有者として指定されたデータベースユーザー（以降、両者を所有者と呼びます）、またはスーパーユーザーのみが使用できます。オブジェクトの所有者、またはスーパーユーザーは、データベースユーザーに対するアクセス権限を制御することで、データベースに接続したデータベースユーザーがどのような表にアクセスできるか、どのような操作を行うことができるかを制御することができます。

Fujitsu Enterprise Postgresでは、アクセス制御の設計や運用を支援する機密管理支援機能を提供します。詳細については、“第7章 機密管理支援機能”を参照してください。

オブジェクトのアクセス制御の詳細については、“PostgreSQL Documentation”の“The SQL Language”の“Privileges”を参照してください。

暗号化

Fujitsu Enterprise Postgresでは、以下の要求事項を満たす透過的データ暗号化機能を提供します。

- 機密情報を判別不可能な状態にできること
- 暗号化キーとデータを分離して管理すること
- 定期的に暗号化キーの交換を行うこと

また、機密レベルの高いデータが暗号化されずに運用されるべきではありません。Fujitsu Enterprise Postgresでは、これを防止することを支援する機密管理支援機能を提供します。詳細については、“第7章 機密管理支援機能”を参照してください。

PostgreSQLでは、pgcryptoという暗号化機能が提供されており、Fujitsu Enterprise Postgresでも使用することができますが、アプリケーションに暗号化を意識した修正が必要となるため透過的データ暗号化機能の使用を推奨します。透過的データ暗号化については、“Fujitsu Enterprise Postgres 運用ガイド”の“透過的データ暗号化による格納データの保護”を参照してください。

また、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し、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必要があります。

通信データの暗号化については、“Fujitsu Enterprise Postgres 運用ガイド”の“Secure Sockets Layerによる安全な通信の構成”を参照してください。

監査ログ

管理者の権限乱用、利用者のデータベースへの不正アクセスなどの脅威に対抗するための機能です。管理者や利用者の処理を追跡するための情報を監査ログとして取得、保持します。

管理者は、監査ログを定期的に参照・監視することによって、システムが何らかの影響を受けている、あるいは利用者が誤ったオペレーションを行っているためにシステムの資源が枯渇しかかっている、といった事象を検知し、適切な対応を行うことによって、情報漏えいやシステムダウンなどを未然に防ぐことができます。

監査ログについては、“第6章 監査ログ機能”を参照してください。

データ秘匿化

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参照させることができます。

たとえば、従業員データの問合せに対して、実際のデータを表示せずに、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。

具体的には、データ秘匿化機能で改訂したデータをテスト用データベースに移行し、テストや開発を行うユーザーに参照させる利用方法があります。テスト時には本番環境データベースで利用するデータを用いてテストを行うことが求められます。しかし、機密データが流出するリスクが高まるため、本番環境データベースのデータをそのまま使用するべきではありません。本機能を利用することで、本番環境データベースにより近いデータをテストや開発環境でも安全に利用することができます。

データ秘匿化については、“Fujitsu Enterprise Postgres 運用ガイド”の“データ秘匿化”を参照してください。

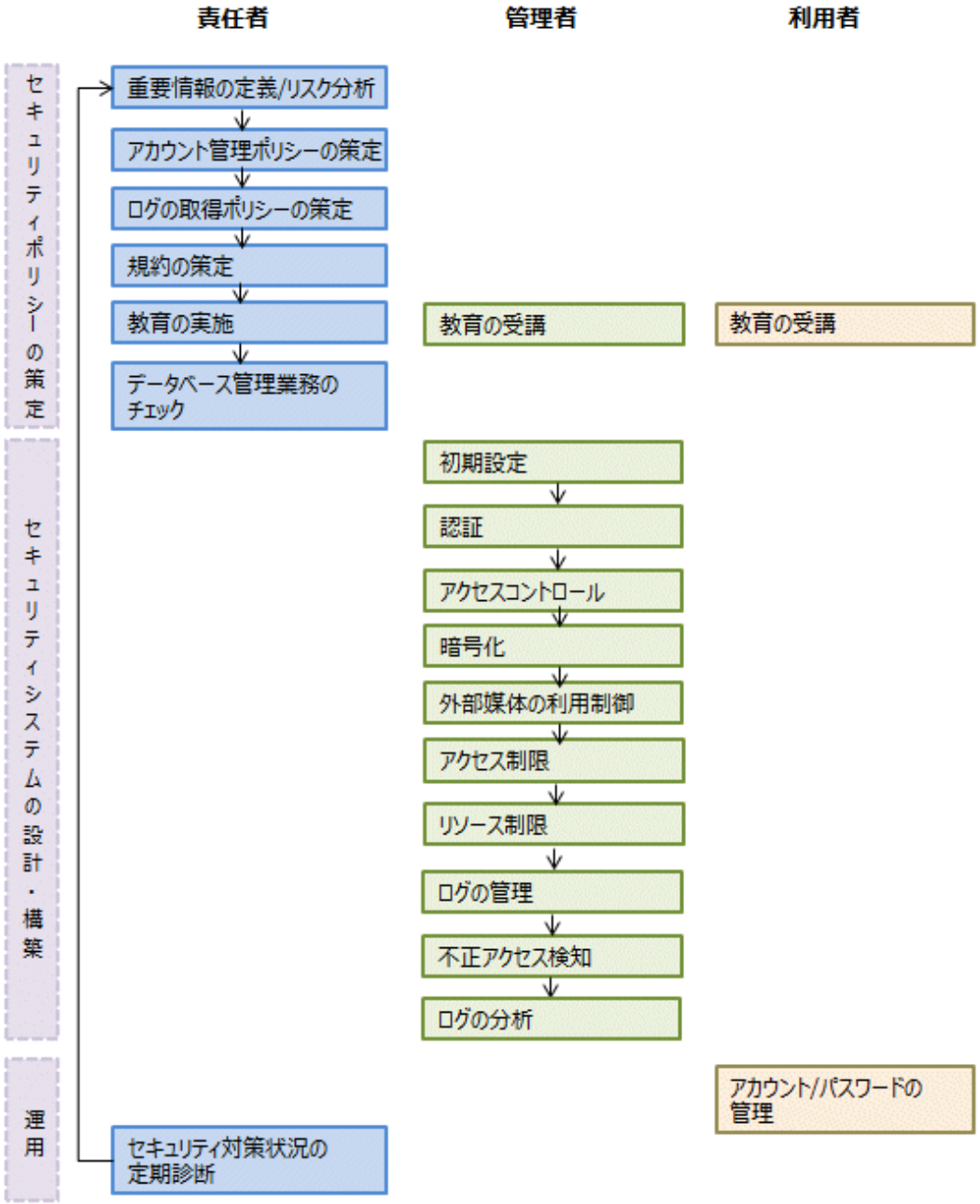
第2章 セキュリティ運用の概要

2.1 セキュリティ運用の流れ

Fujitsu Enterprise Postgresでセキュリティ環境を構築し、セキュリティ運用を行う場合の作業の流れを示します。

セキュリティ運用を行う場合、システムにセキュリティ機能を装備することによりセキュリティ脅威に対応する技術的な運用作業と、セキュリティの指針、教育制度および利用規則の制定などを実施する人的な作業があります。

図2.1 セキュリティ運用の流れ



第3章 責任者の作業

責任者は、セキュリティ対策の指針となるセキュリティポリシーを策定します。

3.1 重要情報の定義/リスク分析

セキュリティポリシーを策定する前に、重要情報の定義とリスク分析を行います。情報の重要度やリスク分析の結果に基づいて、どのようなセキュリティ対策を講じるのかを決定します。

重要情報の定義では、セキュリティ対策を有効に実施するため、守るべき情報を洗い出し、情報を重要度により分類します。守るべき情報としては、“情報資産について”に示したように、“データベース管理情報”、“データベース一般情報”があります。洗い出した情報は、“個人情報”、“機密情報”といった、情報の重要度により分類します。

リスク分析では、“想定される脅威”を参考に、起こりうる脅威を洗い出して、脅威に対するリスクを分析します。

また、年に一度を目安に、リスク分析を行うことで、業務に悪影響を及ぼす可能性がある脅威や関連する脆弱性を識別することができます。

3.2 アカウント管理ポリシーの策定

アカウント管理ポリシーの策定では、以下を実施し、策定したポリシーを文書化します。

システム利用者と役割の整理

“登場人物について”にもとづいて、対象とするシステムで必要な役割を洗い出します。また、役割ごとに要員を整理します。

アカウントの整理

役割ごとに適切な権限を持つようにアカウントを整理し、アカウントのポリシーを決定します。

- データベース管理者アカウント
  - データベース管理者とデータベース運用者でアカウントを分ける
  - データベース管理者アカウントは、特定の人しか利用できないようにする
  - データベース管理者権限を必要としない作業は、データベース管理者権限なしの別アカウントで実施する
- 一般アカウント

  アプリケーションの用途ごとに一般利用者のアカウントを作成する

アカウント管理ポリシーの見直し

セキュリティ対策を有効に実施するため、アカウントに関して、以下の見直しを実施します。

- 整理したアカウント、権限を定期的にチェックし、状況に適しているかを判断する
- システム変更や運用変更があった場合は、アカウント、権限を見直す
- 不適切なアカウント、権限を発見した場合は、再度、アカウントを整理する

3.3 ログの取得ポリシーの策定

ログの取得ポリシーの策定では、以下を実施し、策定したポリシーを文書化します。

ログ取得の目的を整理

何のためにログを取得するのかを明確にするため、取得するログの目的を整理します。

たとえば、目的の例として、「不正アクセス時の調査に利用する」、「何らかの問題が発生した場合に、捜査機関への証拠として提出する」などがあります。

取得するログの種類を決定

適切なログを取得するため、対象とするシステムにおいて取得できるログの種類を整理し、取得するログを決定します。

たとえば、ログの種類として、「OSログ」、「アプリケーションの実行ログ」、「データベースの監査ログ」などがあります。

ログ取得対象アクセスの整理

ログ取得の対象とするアクセスを決定するため、どのようなアクセスがあるかを整理します。

たとえば、以下のアクセスが考えられます。

- 重要情報に関するアクセス
  - 個人情報、機密情報、データベース管理情報へのアクセス
  - 業務時間外のアクセス
  - ログイン
  - 特定のSQL
- 不正が疑われるアクセス
  - 大量検索アクセス
  - 異なる場所からのアクセス
  - 業務時間外のアクセス

ログ取得内容の決定

取得したログを有効利用するため、ログとして必要な内容を整理し、取得内容を決定します。

たとえば、以下の出力内容が考えられます。

- いつ (時間)
- 誰が (データベースアカウント、アプリケーションユーザー)
- 何を (オブジェクトID、テーブル名)
- どこから (マシン名、IPアドレス)
- どうやって (SQL種別、SQL文)
- 実行結果 (成功/失敗)

ログの保全方針の策定

ログを正当な状態で利用するため、ログの保全指針を整理します。

たとえば、「保管場所」、「保管媒体」、「保管期間」、「アクセス制御」などがあります。

3.4 規約の策定

対象とするシステムのセキュリティ対策の基準となる規約を策定します。また、セキュリティ違反時の罰則を規定します。たとえば、以下のような規約や罰則を策定します。

- 規約の策定
  - セキュリティパッチや更新プログラムの適用
  - 無許可でのデータベースからの情報入手の禁止
  - 入手した情報について、許可された媒体以外への保存を禁止
- 不正利用による罰則の規定
  - 就業規則での罰則の規定
  - 罰則金の設定

3.5 教育の実施

情報セキュリティの重要性や必要性を認識させ、作業漏れや誤操作による不正アクセスを防ぐため、管理者および利用者に対して、セキュリティに関する教育を実施および推進します。

たとえば、「セキュリティポリシーの広報」、「教育スケジュールの策定」、「教育資料の策定」などを実施します。

3.6 データベース管理業務のチェック

管理者の作業ミスや不正行為を防止するため、以下の対策を実施します。

- データベースに関するセキュリティの事故や脆弱性の最新情報を常に収集する
- 管理業務は、事前申請を行ったうえで実施する
- 管理業務の記録を残す

3.7 セキュリティ対策状況の定期診断

セキュリティ対策が有効であるかを確認するため、セキュリティの脅威にもとづいたセキュリティ対策が適切に講じられているかを定期的に診断します。

また、脅威や脆弱性などに対し、現在のセキュリティ対策やセキュリティポリシーが有効であるかを評価し、問題がある場合は、セキュリティポリシーやセキュリティ対策を見直します。

第4章 管理者の作業

管理者は、責任者が策定したセキュリティポリシーに従い、システムの設計/構築/運用を行う際に、セキュリティ対策として以下を実施します。

事前準備

- 教育の実施

不正行為を防御するための対策

- 初期設定
- 認証
- アクセスコントロール
- 暗号化
- 外部媒体の利用制御
- アクセス制御
- リソース制限

不正行為を検知/追跡するための対策

- ログの管理
- 不正アクセス検知
- ログの分析

4.1 教育の受講

管理者は、責任者が策定した教育スケジュールに従って、セキュリティに関する教育を受講します。また、利用者に対し、教育の受講を指示します。

4.2 初期設定

データベースの脆弱性および不正アクセスの可能性が最小限となるように、システム構築の初期段階で以下のセキュリティ対策を実施します。また、データベースサーバでは、主として、データベースシステムのみを稼働させるような構成とします。

サーバの堅牢化

データベースサーバへの侵入や破壊などを防ぎ、セキュアなサーバでシステムが稼働されるようにOSやネットワークを設定します。

- 使用するOSにおいて、不要な機能やサービスを削除する
- 必要なプロトコルのみを有効にし、不要なプロトコルは無効にする
- ファイル共有、FTPなどセキュリティのレベルが比較的低いとされるサービス、プロトコル、デーモンにセキュリテイ機能を実装する

最新バージョンの導入

最新のセキュリティ対策を反映するため、常に最新のパッチを入手し、適用します。

必要最低限の機能の導入

システムの不正利用を防ぐため、インストール時に、必要な機能のみを導入します。

また、使用しない機能やサービスは、削除または無効にします。

ポートの変更

システムの不正利用を防ぐため、インストール時に作成されるデフォルトのポートを変更します。

ポイント

ポートは、Fujitsu Enterprise Postgresのセットアップで指定します。詳細は、“Fujitsu Enterprise Postgres 導入ガイド(サーバ編)”を参照してください。

通信機能のアクセス制限

通信機能を利用したシステムの不正利用を防ぐため、通信機能のアクセス制限を実施します。

データベース構成ファイルへのアクセス経路禁止の設定

データベース破壊を防ぐため、以下の対策を実施します。

- データベース構成ファイルへのアクセス許可者を制限し、定期的に権限見直しを実施する
- 表や定義スクリプトに対しては、管理者だけがアクセス可能とする

データベースへのアクセス経路の制限

データベースの不正利用や操作ミスを防止するため、データベースをアクセスするためのアプリケーションの配布範囲は、アクセス許可者が使用する機器のみに限定します。

不正プログラムの対処

アプリケーションのプログラムソースコードの改ざんなど、バックドアによるシステムへの不正侵入を防ぐため、稼働するプログラムは作成者を明文化し、改ざんなどがされないように、チェックおよびテストを実施します。また、安全なコーディング手法を採用し、一般的なコーディングの脆弱性の問題に対応できるようにしておくことも必要です。

システムのセキュリティ設定

システムのセキュリティ設定において、セキュリティに影響を与えることが明らかである場合、セキュリティパラメータを適切に設定するなどして、初期設定の段階で、確実にセキュリティ設定を実施します。

4.3 認証

悪意を持った利用者のなりすましによる情報漏えいや改ざんを防ぐため、データベースへのアクセス時には、必ず認証を行う必要があります。

パスワード認証は、データベースにログインする際に使用しますが、管理者は、認証で使用するアカウントやパスワードを厳重に管理します。

また、許可されたユーザーだけがデータベースにアクセスできるように、クライアントからデータベースへの接続に対する認証も確実に実施します。

4.3.1 アカウントの管理

アカウントの管理では、以下を実施します。

必要なアカウントの作成

なりすましなどのアカウントの不正利用を防ぐため、アカウントの作成時に、以下の対策を実施します。

- 必要なアカウントを選定する
- 利用者の権限を規定する
- データベース管理者アカウントと一般アカウントは、それぞれの権限に応じて、別々に作成する

ポイント

アカウントは、CREATE ROLE文で作成します。詳細は、“PostgreSQL Documentation”の“CREATE ROLE”を参照してください。

不要なアカウントの削除

「未使用のアカウント」や「製品インストール時にデフォルトで作成される業務で不要なアカウント」など、日常的に使用されないアカウントを削除します。

ポイント

アカウントは、DROP ROLE文で削除します。詳細は、“PostgreSQL Documentation”の“DROP ROLE”を参照してください。

アカウントのロックアウトの設定

定期的にアカウントの使用頻度をチェックし、長期間使用されないアカウントがある場合は、アカウントをロックします。また、ログインの失敗回数の許容回数を設定し、この回数を超えてログインに失敗した場合は、アカウントをロックします。また、アカウントがロックされた場合に、アカウントが再有効化されるまでの期間を設定します。

ポイント

LDAP認証を使用することで、アカウントのロックアウトを設定することができます。詳細は、“PostgreSQL Documentation”の“LDAP Authentication”を参照してください。

データベース管理者アカウントの管理

責任者が策定したアカウント管理ポリシーに従い、データベース管理者アカウントを管理します。

開発環境と運用環境のアカウント

開発環境で利用したアカウントの不正利用を防ぐため、運用環境での運用が開始される前に、開発環境で使用したアカウントを削除します。やむを得ず、開発環境で利用したアカウントを運用環境で利用する場合は、それぞれの環境において異なるパスワードを使用します。

一時利用アカウントの設定

一時的な利用者がシステムを利用する場合、共有アカウントに対し、利用ごとに一時的なパスワードを与えるか、または、一時的なアカウントを作成します。

4.3.2 パスワードの管理

パスワードの管理では、以下を実施します。

パスワードの複雑化

アカウントのパスワードは、他者に容易に推察されないように、「IDと同じパスワード」や「インストール時のデフォルトパスワード」の使用を禁止します。また、パスワードには、複雑性と強度を設定します。

パスワードの定期的な変更

パスワードを不正入手された場合に備え、パスワードの定期的な変更を設定します。また、初回利用時に使用したパスワードの変更を促すため、初回利用時のパスワードの強制的な変更を設定します。

パスワードの有効期限の設定

パスワードの定期的な変更を促進するため、パスワードに有効期限を設定します。

ポイント

パスワードの設定/変更は、CERATE ROLE文またはALTER ROLE文で指定します。詳細は、“PostgreSQL Documentation”の“CREATE ROLE”および“ALTER ROLE”を参照してください。

また、passwordcheckやLDAP認証を使用することで、以下を実施することができます。

- インストール時のデフォルトパスワードの変更
- パスワードの有効期間の設定
- パスワードの文字数や文字種のチェック

passwordcheckについては、“PostgreSQL Documentation”の“passwordcheck”を参照してください。また、LDAP認証については、“PostgreSQL Documentation”の“LDAP Authentication”を参照してください。

4.3.3 接続と認証の設定

利用者がデータベースを利用する場合に、許可された利用者だけがデータベースをアクセスできるように接続と認証を設定します。

ポイント

pg_hba.confでクライアント認証を設定します。詳細は、“PostgreSQL Documentation”の“Client Authentication”を参照してください。

4.4 アクセスコントロール

管理者および利用者に対して適切なアクセス権限が設定されていない場合、意図しない人物による情報アクセスが発生し、情報漏えいなどのセキュリティ事故が発生してしまう可能性があります。これらを防御するため、アクセス権限において、以下のセキュリティ対策を実施し、ルールに基づいたアクセスコントロールを行う必要があります。

ポイント

アクセス権限設定における留意点

- 全ユーザーに権限を付与できる特殊なアカウントの作成は禁止する
- 業務データなどの一般情報にアクセスできる一般アカウントの作成は禁止する

データベースへのアクセス要件の洗い出し

データベースの利用目的ごとに適切なアクセス権限を設定するため、以下の手順でアクセス要件を洗い出します。

1. 「データベース管理用」、「オブジェクト管理用」、「データアクセス用」など、アカウントの利用目的を分類します。
2. 「機能別」、「オブジェクト別」などアカウントの利用目的ごとに必要な権限を分類します。
3. それぞれの権限に基づいて、アカウントを分割します。
4. 分割したアカウントのそれぞれについて、アクセスする必要がある必要最小範囲のデータと必要最低限のアクセス内容(参照、更新、作成、削除)を洗い出し、データベースへのアクセス要件を決定します。

アクセス権限の設定

分割された各アカウントについて、データベースへのアクセス要件に基づいて、必要最低限の権限を付与します。また、管理者権限は、アカウントを限定して付与します。

アクセス権限の見直し

システムに対するアクセス要件の変化に対応するため、アクセス権限を定期的に見直し、不要なアクセス権限がないかを確認します。不要なアクセス権限が設定されている場合は、速やかにアクセス権限を修正します。

ポイント

アクセス権限は、GRANT文やREVOKE文で設定します。詳細は、“PostgreSQL Documentation”の“GRANT”および“REVOKE”を参照してください。

4.5 暗号化

データの盗難や通信の盗聴など、情報漏えい発生時のデータの不正利用を防ぐため、以下の暗号化対策を実施します。

通信の暗号化

データベースサーバとクライアント間におけるネットワーク上の盗聴に備え、暗号化機能を使用し、通信の暗号化を実施します。

通信の暗号化については、“Fujitsu Enterprise Postgres 運用ガイド”の“Secure Sockets Layerによる安全な通信の構成”を参照してください。

データの暗号化

データの盗難に備え、暗号化機能を使用し、データの暗号化を実施します。暗号化は、以下のデータを対象とします。

- データベースへの格納データ
- バックアップデータ
- データファイル

データの暗号化については、“Fujitsu Enterprise Postgres 運用ガイド”の“透過的データ暗号化による格納データの保護”を参照してください。

暗号鍵の管理

暗号化で使用する暗号鍵に対しては、アクセス可能な人物を最小限のデータベース管理者に限定します。

また、暗号化された情報が容易に復号されないように、ライフサイクル(生成、配布、保存、廃棄など)ごとに暗号鍵を適切に管理する仕組みを作り、暗号鍵を厳重に管理します。

暗号鍵の管理については、“Fujitsu Enterprise Postgres 運用ガイド”の“Secure Sockets Layerによる安全な通信の構成”および“透過的データ暗号化による格納データの保護”を参照してください。

4.6 外部媒体の利用制御

データベースに接続される外部媒体(CD/DVD、USBメモリ、外付けハードディスクなど)や端末の利用を制御し、データベースからのデータの持ち出しを制限することで、情報漏えいを未然に防ぐことができます。

外部媒体の接続制限

業務で使用しない外部媒体やプリンタを取り外し、情報の書き出し先となる外部媒体の接続を制限します。

外部媒体の利用制限

外部媒体やプリンタに対する接続制限を実施し、外部媒体への情報の書き出しを抑止します。

接続端末の利用制御

データベースに接続する端末に対して、以下の対策を実施することで、端末からの情報流出を防御します。

- 端末に対する外部媒体の接続制限を実施する
- 端末に対し、堅牢化のためのセキュリティ対策を実施する
- 端末アクセスの個人認証を実施する
- インストールされているソフトウェアを管理し、ソフトウェアの利用状況を監視する
- プリンタへの接続制限を実施する

4.7 サーバ/アプリケーションに対するセキュリティ対策

データベースに対するセキュリティ対策に加え、サーバやアプリケーションなどに対するセキュリティを強化することで、より強固なセキュリティシステムを実現できます。サーバやアプリケーションに対して、以下のセキュリティ対策を実施します。

アクセス制限

以下の対策を実施し、データベースサーバへのアクセスを制限します。

- データベースサーバは、ファイアウォールより内側に設置し、不特定多数の端末から直接データベースサーバにアクセスできないようにする。
- ローカルネットワークにおいては、ルータによりIPアドレスを制限するなどの対策を実施し、直接データベースサーバにアクセスできる端末やセグメントを限定する。

リソース制限

サービスの妨害や大量データの抽出を制限するため、一般利用者における、想定を超えるCPUリソースの使用を制限します。

4.8 ログの管理

ログは、管理者の権限乱用、利用者のデータベースへの不正アクセスなどの脅威に対抗するための機能です。情報漏えいや不正アクセスが発生した場合に、原因を特定するため、データベースに対する処理や操作を調査/追跡するための情報をログとして取得し管理します。

Fujitsu Enterprise Postgresでは、ログを取得、管理するための機能として、監査ログ機能が提供されています。監査ログ機能については、“第6章 監査ログ機能”を参照してください。

ここでは、情報漏えいや不正アクセスへの対策として、ログとして取得すべき情報とログの保全方法を説明します。

4.8.1 ログの取得

責任者が策定したログの取得ポリシーに従い、以下の監査ログを取得します。

ログイン情報

ログインおよびログアウト時のログを取得します。

データベースへのアクセス情報(参照/更新)

以下の情報に関するすべてのアクセスを取得します。

- データベースの一般情報(業務で使用する個人情報や機密情報など)
- データベースの管理情報(システムカタログ、ユーザーID/パスワードなど)

データベースオブジェクトの変更情報

データベースアカウント、テーブルなどのデータベースオブジェクトの作成、変更、削除に関するログを取得します。

監査ログに対する操作ログ

取得した監査ログの隠ぺいを防ぐため、監査ログの初期化や監査ログ機能の停止をログとして取得します。

4.8.2 ログの保全

責任者が策定したログの取得ポリシーに従い、ログの保全を実施します。

ログの保管

取得したログが他者に更新されないように、以下を実施し、ログを安全に保管します。

- ログを外部記憶媒体に保存し、外部記憶媒体は、施錠できる場所など、安全な場所に保管する
- ログの参照は、管理者のみに限定し、更新権を付与しないなど、ログにアクセス制限を設定する
- 問題発覚時期から遡った調査が必要な場合を考慮し、ログの保管期間を決定する

ログの改ざん防止

ログを多重化する、書き換え不可能なストレージを使用するなど、ログが改ざんされないような対策を実施します。

ログの暗号化

ログが容易に参照されないように、ログを暗号化します。

4.9 不正アクセス検知

不正アクセスに対処するため、データベースへの不正アクセスを検知する仕組みを設けて、アクセスを監視する必要があります。

不正アクセスの通知

ログイン失敗の許容回数を超えたアカウントロックが発生した場合、責任者および管理者に通知するなど検知した不正アクセスを通知する仕組みを作ります。

アクセス時間のチェック

以下の情報に対し、正規のアクセス時間帯以外の疑わしいアクセスがチェックできるような仕組みを作るとともに、それぞれの対策を実施します。

データベース管理情報へのアクセス検知

- ログを監視し、申請されていない時間帯のアクセスを検知する
- 正規のアクセス時間帯以外のアクセス許可を申請された場合、申請された内容と作業結果に相違のないことをログを見て確認する

データベース一般情報へのアクセス検知

- 一般アカウントごとに、データベースへのアクセスを許可する時間帯を決める
- セション情報のログから、正規のアクセス時間帯でないアクセスを検知する

アクセス不可の接続元のチェック

許可されていない接続元からのアクセスを検知するため、アクセス可能な接続元を定義し、許可されていない接続元からのアクセスを検知します。

データベース管理者アカウントおよび一般アカウントによるアクセスのパターン(接続元/OSユーザー/アカウント)を定義し、このパターン以外でのアクセスをチェックします。

4.10 ログの分析

情報漏えいや不正アクセスなどが疑われた場合、ログを分析する仕組みを作り、取得したログを分析することで、不正行為を検出します。以下の分析を実施します。

定期的なセション情報の分析

不正なログインを検出するため、以下のような観点でログのセション情報を分析します。

ー ログイン失敗回数が多いセションの傾向

ー 長時間ログインしているセションの傾向

ー 大量のリソースを消費するセションの傾向

定期的なデータベースアクセス情報の分析

不正なデータベースへのアクセスを検出するため、以下のような観点でSQL文を分析します。

ー 長時間に渡り実行されているSQLの傾向

ー 大量のリソースを消費するSQLの傾向

第5章 利用者の作業

利用者は、システムを利用する際に、セキュリティ対策として以下を実施します。

5.1 教育の受講

利用者は、責任者または管理者の指示するセキュリティに関する教育を受講して、セキュリティについての知識を習得する必要があります。利用者間でセキュリティに関する共通の意識を持つことで、より強固なセキュリティシステムの確立が実現できます。

5.2 アカウント/パスワードの管理

利用者は、管理者から提示されたアカウントおよびパスワードを使用することで、データベースシステムを利用できます。その際、アカウントおよびパスワードを他人に悪用されないように、以下の対策を実施します。

- ログイン時にアカウントロックがかからないように、IDおよびパスワードは責任を持って管理する
- 定期的にパスワードを変更する
- パスワードに設定されている有効期限を遵守し、期限が近づいたら速やかにパスワードを変更する

第6章 監査ログ機能

PostgreSQLでは、ログ出力機能を利用することで、サーバログとして出力されたログを監査ログとして利用することができますが、SQL実行時のログとして、スキーマ名が出力されないなど、適切な分析ができないログが存在します。また、出力条件について、詳細な指定ができないため、ログ量が多く、性能劣化につながる可能性があります。

Fujitsu Enterprise Postgresの監査ログ機能は、pgauditに機能を拡張することで、データベースアクセスに関する詳細な情報を監査ログとして取得することができます。また、監査ログは、専用ログファイルまたはサーバログに出力できます。これにより効率的かつ正確なログ監視が可能になります。

注意

PostgreSQLがシングルユーザーモードで起動している場合には、監査ログ機能を利用することはできません。

6.1 監査ログの出力モード

pgauditでは、以下の2種類の監査ログを出力することができます。

Session Audit Logging

Session Audit Loggingは、バックエンドプロセス(クライアントから接続要求を受けたときに生成されるプロセス)で実行されたSQLに関する情報、データベースの起動や接続に関する情報、エラーに関する情報をログとして出力します。Session Audit Loggingでは、ログ出力のための条件を指定し、出力するログを絞り込むことで、大量のログ出力に対する性能劣化を防ぐことができます。

Session Audit Loggingについては、“6.4 Session Audit Logging”を参照してください。

Object Audit Logging

Object Audit Loggingは、特定のオブジェクト(テーブルや列)に対して、SELECT、INSERT、UPDATE、DELETEが実行されたときに、ログとして出力します。TRUNCATEには対応していません。Object Audit Loggingでは、指定されたロールに対して、権限が付与されているオブジェクト操作をログとして出力します。Session Audit Loggingよりもさらに細かい粒度でログ出力を制御することができます。

Object Audit Loggingについては、“6.5 Object Audit Logging”を参照してください。

参考

アプリケーションやコマンドによっては、Fujitsu Enterprise Postgresが内部でSQLを実行し、その監査ログが取得される場合があります。

また、同じステートメントIDを持つ複数のSQLの監査ログが取得される場合があります。これは、利用者がSQLを実行する場合に、Fujitsu Enterprise Postgresによって、内部で別のSQLが実行されるためです。

6.2 セットアップ

pgauditのセットアップ方法を説明します。

1. pgauditのファイルコピー

   スーパーユーザーで以下のコマンドを実行します。“<*x*>”は、製品のバージョンを示します。

   ```
   $ su -
   Password:******
   # cp -r /opt/fsepv<x>server64/OSS/pgaudit/* /opt/fsepv<x>server64
   ```

2. pgaudit設定ファイルの作成

   pgauditの動作に必要な情報を記述するpgaudit設定ファイルを任意のファイル名で作成します。pgaudit設定ファイルは、データベースの符号化方式と同じエンコードで作成してください。

   また、監査ログに関するポリシーを意図しないユーザーに参照されないようにするために、pgaudit設定ファイルにはデータベース管理者にのみ読み込み権限を設定してください。

   pgaudit設定ファイルの詳細については、“6.3 pgaudit設定ファイル”を参照してください。

注意

この時点では、pgaudit設定ファイルのruleセクションは定義しないでください。

pgaudit設定ファイルの例

```
[output]
logger = 'auditlog'
```

3. postgresql.confの設定

   postgresql.confに、監査ログを利用するための以下のパラメータを設定します。

   shared_preload_libraries

   “pgaudit”を指定します。

   pgaudit.config_file

   pgaudit設定ファイルの配置先パス名を指定します。

   相対パスで指定した場合は、データ格納先のディレクトリからの相対パスとなります。

   log_replication_commands

   “on”を指定します。

   log_min_messages

   ERROR以上が指定されていることを確認します。

   監査ログをサーバログに出力する場合(pgaudit設定ファイルのloggerパラメータに“serverlog”を指定)は、サーバログに関する以下のパラメータを確認します。

   logging_collector

   “on”が指定されていることを確認します。

   log_destination

   “stderr”が指定されていることを確認します。

   log_file_mode

   許可された人物だけにサーバログへのアクセスを許可するために、サーバログのパーミッションが適切であることを確認します。

   ### 参考

   log_file_modeパラメータのデフォルトは0600であり、データベース管理者のみアクセス可能です。

   たとえば、データベース管理者のグループにも参照を許可する場合は、log_file_modeに0640を指定します。

   例

   ```
   log_file_mode = 0640
   ```

   0000を指定することで、データベース管理者も参照不可にすることができます。ただし、ログ出力を行うため、書き込み権限は付与された状態となります。

   監査ログを専用ログファイルに出力する場合(pgaudit設定ファイルのloggerパラメータに“auditlog”を指定)は、以下のパラメータを確認します。

   max_worker_processes

   max_worker_processesパラメータを設定している場合は、指定されている値に1を追加します。

参照

サーバログの詳細は“PostgreSQL Documentation”の“Error Reporting and Logging”を参照してください。

データベース多重化運用を使用する場合は、“6.6 データベース多重化運用の場合”を参照してください。

postgresql.confの例

下記の例では、監査ログ機能利用時に設定が必要なパラメータのみ記載しています。

```
shared_preload_libraries = 'pgaudit'
pgaudit.config_file = 'pgaudit.conf'
log_replication_commands = on
log_min_messages = WARNING
```

4. インスタンスの起動

   インスタンスを起動し、以下のメッセージが出力されることを確認してください。

```
LOG: pgaudit extension initialized
```

5. pgauditエクステンションの作成

   CREATE EXTENSIONを使用し、pgauditエクステンションを作成します。

```
$ psql
=# CREATE EXTENSION pgaudit;
=# ¥dx
                 List of installed extensions
Name    | Version |   Schema   |     Description
--------+---------+------------+--------------------------------
pgaudit | 1.0     | public     | provides auditing functionality
plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
(2 rows)
```

6. pgaudit設定ファイルのパラメータ設定

   pgaudit設定ファイルのパラメータを追加/変更します。

   pgaudit設定ファイルの詳細は“6.3 pgaudit設定ファイル”を参照してください。

7. インスタンスの再起動

   pgaudit設定ファイルの変更を適用するため、インスタンスを再起動します。再起動後、変更内容が正しく反映されていることを確認してください。

```
LOG:  log_catalog = 1
LOG:  log_level_string =
LOG:  log_level = 15
LOG:  log_parameter = 0
LOG:  log_statement_once = 0
LOG:  role =
LOG:  logger = auditlog
LOG:  log_directory = pgaudit_log
LOG:  log_filename = pgaudit-%Y-%m-%d_%H%M%S.log
LOG:  log_file_mode = 0600
LOG:  log_rotation_age = 1440
LOG:  log_rotation_size = 10240
LOG:  log_truncate_on_rotation = 0
LOG:  fifo_directory = /tmp
LOG:  Rule 0
LOG:  pgaudit extension initialized
```

6.3 pgaudit設定ファイル

pgaudit設定ファイルには、pgauditの動作に必要な情報を指定します。pgaudit設定ファイルは、“outputセクション”、“optionセクション”、“ruleセクション”の3つのセクションから構成されています。

outputセクション

outputセクションは、以下の形式で指定します。

- パラメータ名 = '値'

outputセクションで指定可能なパラメータを下表に示します。

パラメータ名	説明	備考
logger	監査ログの出力先について、専用ログファイル(auditlog)/サーバログ(serverlog)を指定します。 デフォルトは、“auditlog”(専用ログファイル)です。	専用ログファイルは、データベースの符号化方式と同じエンコードで出力されます。
log_directory	監査ログを作成するディレクトリを指定します。ディレクトリは、絶対パスまたはデータ格納先のディレクトリからの相対パスで指定します。 デフォルトは、“pgaudit_log”です。	loggerパラメータに“auditlog”を指定した場合のみ有効
log_filename	監査ログのファイル名を指定します。postgresql.confファイルのlog_filenameと同様に、時刻によって変動するファイル名を指定します。 デフォルトは、“pgaudit-%Y-%m-%d_%H%M%S.log”です。	loggerパラメータに“auditlog”を指定した場合のみ有効
log_file_mode	許可された人物だけに監査ログファイルへのアクセスを許可するために、監査ログファイルのパーミッションを指定します。パラメータの値はchmod、およびumaskシステムコールで許容されるフォーマットで指定される数値モードです。デフォルトは“0600”です。 監査ログファイルのパーミッションについては、“6.2 セットアップ”の“log_file_mode”を参照してください。	loggerパラメータに“auditlog”を指定した場合のみ有効
log_rotation_age	監査ログファイルの最大寿命を指定します。ここで指定した時間(分単位)が経過すると、新しい監査ログファイルが生成されます。“0”を設定することで、時間に基づいた新しいログファイルの生成は無効になります。 指定可能な単位はmin(分)、h(時間)、d(日数)です。単位を省略した場合はmin(分)となります。 デフォルトは、“1d”(1日)です。	loggerパラメータに“auditlog”を指定した場合のみ有効
log_rotation_size	監査ログファイルの最大容量を指定します。ここで指定したキロバイト分のログがログファイルに出力された後、新しいログファイルが生成されます。“0”を設定することで、サイズに基づいた新しいログファイルの生成は無効になります。 指定可能な単位はkB(キロバイト)、MB(メガバイト)、GB(ギガバイト)です。単位を省略した場合はkB(キロバイト)となります。 デフォルトは、“10MB”です。	loggerパラメータに“auditlog”を指定した場合のみ有効
log_truncate_on_rotation	監査ログファイルを時間を基にしてローテーションする場合に、監査ログを既存の同名のファイルに上書きする(on)/上書きしない(off)を指定します。例えば、“on”を指定し、log_filenameに“pgaudit-%H.log”を指定すると、24個の時別なログファイルが生成され、それらは周期的に上書きされることになります。	loggerパラメータに“auditlog”を指定した場合のみ有効

パラメータ名	説明	備考
	デフォルトは、“off”です。“off”を指定すると既存の監査ログファイルに追記します。	
fifo_directory	監査ログファイルを出力するデーモンプロセスとバックエンドプロセスの間で使用するFIFO(名前付きパイプ)のディレクトリを指定します。p.PGAUDIT.nnnnという名前のFIFO(nnnnはpostmasterのPID)がfifo_directoriesディレクトリの中に作成されます。いずれのファイルも手作業で削除することはできません。 デフォルトは、“/tmp”です。	loggerパラメータに“auditlog”を指定した場合のみ有効
audit_log_disconnection	Mirroring Controllerを利用する場合、切断ログの出力を有効(on)/無効(off)を指定します。デフォルトは、“off”(無効)です。本パラメータはpostgresql.confのlog_disconnectionsがoffの場合に有効となります。	Session Audit Loggingのみで使用するパラメータ

参考

loggerパラメータに“serverlog”を指定した場合、監査ログはサーバログにログメッセージとして出力されるため、監査ログの先頭にpostgresql.confファイルのlog_line_prefixパラメータによるステータス情報とメッセージ深刻度レベルが出力されます。

loggerパラメータを省略、または“auditlog”を指定した場合、監査ログは専用ログとして専用ログファイルに出力されるため、postgresql.confファイルのlog_line_prefixパラメータによるステータス情報やメッセージ深刻度レベルは出力されません。

監査ログの出力形式については、“6.4 Session Audit Logging”の“出力形式”または“6.5 Object Audit Logging”の“出力形式”を参照してください。

ポイント

log_file_modeパラメータは、postgresql.confのlog_file_modeパラメータの設定およびinitdbコマンドの--allow-group-accessオプションによる影響は受けません。また、監査ログを専用ログファイルに出力する場合、log_directoryパラメータで指定するディレクトリのデフォルト値はデータ格納先のディレクトリ配下になります。そのため、log_file_modeパラメータで指定したパーミッションが、initdbコマンドの--allow-group-accessオプションの対象となるデータ格納先のパーミッションと競合する場合には、log_directoryパラメータで指定するディレクトリはデータ格納先以外を指定してください。

optionセクション

optionセクションは、以下の形式で指定します。

- パラメータ名 = '値'

optionセクションで指定可能なパラメータを下表に示します。

パラメータ名	説明	備考
role	Object Audit Loggingで使用するroleの名前を指定します。 大文字、キーワード、マルチバイト文字およびカンマを指定する場合は、二重引用符で囲んでください。	Object Audit Loggingのみで使用するパラメータ
log_catalog	pg_catalogに対するログ出力の有効(on)/無効(off)を指定します。 pgAdminなど、pg_catalogにアクセスする監査ログを取得しない場合に“off”を指定します。 デフォルトは、“on”(有効)です。	

パラメータ名	説明	備考
log_parameter	SQL実行でパラメータで渡した値の出力について、有効(on)/無効(off)を指定します。 デフォルトは、“off”(無効)です。	
log_statement_once	同じSQLがログの出力対象となる場合、2回目以降のSQLの出力について、抑止する(on)/抑止しない(off)を指定します。 デフォルトは、“off”(抑止しない)です。	
log_level	監査ログのログレベルを指定します。 有効な値は、“DEBUG5”、“DEBUG4”、“DEBUG3”、“DEBUG2”、“DEBUG1”、“INFO”、“NOTICE”、“WARNING”、および“LOG”です。 デフォルトは、“LOG"です。	loggerパラメータに“serverlog”を指定した場合のみ有効

ruleセクション

ruleセクションは、Session Audit Loggingで使用されるセクションです。“6.4 Session Audit Logging”を参照してください。

注意

optionセクションにroleパラメータを指定した場合は、ruleセクションを指定しないでください。ruleセクションを指定すると、Object Audit LoggingとSession Audit Loggingの監査ログが混在して出力されるため、CSV形式で参照できなくなるためです。

6.4 Session Audit Logging

Session Audit Loggingでは、pgaudit設定ファイル内のruleセクションで、出力するログを絞り込むためのルールを指定します。

ルールは、以下の形式で指定します。値はカンマ区切りで複数指定できます。

- パラメータ名 = '値'
- パラメータ名 != '値'

ruleセクションに[rule]のみ記載し、パラメータを指定しない場合、すべての監査ログが出力されます。

例

```
[output]
logger = 'auditlog'
[rule]
```

また、ruleセクションを記載しない([rule]を記載しない)場合、監査ログは出力されません。

例

```
[output]
logger = 'auditlog'
```

ruleセクションで指定可能なパラメータを下表に示します。

パラメータ名	説明	指定例
timestamp	タイムスタンプの範囲。 指定形式については、“timestamp”を参照してください。	timestamp = '09:00:00 - 10:00:00, 18:00:00 - 18:30:00'

パラメータ名	説明	指定例
database	データベース名。 空の値を指定する場合は、二重引用符を2つ続けて指定("")してください。大文字、キーワード、マルチバイト文字およびカンマを指定する場合は、二重引用符で囲んでください。	database = 'product_db'
audit_role	ロール名。 空の値を指定する場合は、二重引用符を2つ続けて指定("")してください。大文字、キーワード、マルチバイト文字およびカンマを指定する場合は、二重引用符で囲んでください。	audit_role = 'appuser1'
class	操作のクラス。 以下から選択します。複数指定が可能です。各クラスの意味については、“class”を参照。 ・ BACKUP ・ CONNECT ・ DDL ・ ERROR ・ FUNCTION ・ MISC ・ READ ・ ROLE ・ WRITE ・ SYSTEM	class = 'READ, WRITE'
object_type	オブジェクトの種類。 classパラメータが“READ”および“WRITE”の場合に有効になります。 以下から選択します。複数指定が可能です。 ・ TABLE ・ INDEX ・ SEQUENCE ・ TOAST_VALUE ・ VIEW ・ MATERIALIZED_VIEW ・ COMPOSITE_TYPE ・ FOREIGN_TABLE ・ FUNCTION	object_type = 'TABLE, INDEX'
object_name	オブジェクト名。 classパラメータが“READ”および“WRITE”の場合に有効になります。 テーブルなどスキーマ修飾できるオブジェクトは、スキーマ名で修飾してください。	object_name = 'myschema.tbl1' object_name = '"myschema.テーブル1"' object_name = 'myschema.tbl1, "マイスキーマ.TABLE"'

パラメータ名	説明	指定例
	大文字、キーワード、マルチバイト文字およびカンマを指定する場合は、二重引用符で囲んでください。“スキーマ名.テーブル名”といったオブジェクト名を指定する場合は、スキーマ名修飾したオブジェクト名全体を二重引用符で囲んでください。 空の値を指定する場合は、二重引用符を2つ続けて指定("")してください。	
application_name	アプリケーション名。 空の値を指定する場合は、二重引用符を2つ続けて指定("")してください。	application_name = 'myapp'
remote_host	接続先のホスト名またはIPアドレス。 postgresql.confファイルのlog_hostnameパラメータに“on”を指定している場合は、ホスト名を指定してください。それ以外の場合は、IPアドレスを指定してください。なお、ローカルホストの場合は“[local]”を指定してください。 空の値を指定する場合は、二重引用符を2つ続けて指定("")してください。	remote_host = 'ap_server'

timestamp

ログ出力の対象を、「開始時刻」から「終了時刻」までのタイムスタンプの範囲で指定します。指定形式は、'hh:mm:dd-hh:mm:dd'(hhは24時間表記、また、hh、mm、ddは2桁で表記)です。

開始のタイムスタンプは、終了のタイムスタンプよりも小さい値を指定する必要があります。複数の範囲を指定する場合はタイムスタンプの開始と終了をコンマで区切ります。

終了時刻は、終了時刻のタイムスタンプのミリ秒までが対象となります。たとえば、'11:00:00 - 11:59:59'と指定した場合は、「11:00:00:000」から「11:59:59:999」までを対象とします。

pgauditのruleセクション内の評価で使用されたタイムスタンプは、ログエントリで発行されたタイムスタンプとは異なります。ログエントリは、pgauditの評価後に出力され、その時点でタイムスタンプが生成されるためです。

class

classパラメータで指定するクラスの意味は、下記のとおりです。

- READ: SELECT、COPY FROM
- WRITE: INSERT、UPDATE、DELETE、TRUNCATE、COPY TO
- FUNCTION: 関数呼び出し、DO
- ROLE: GRANT、REVOKE、CREATE ROLE、ALTER ROLE、DROP ROLE
- DDL: ROLEクラスのDDL以外のすべてのDDL (CREATE...、ALTER...など)
- CONNECT: 接続に関するイベント(要求、認証、切断)
- SYSTEM: インスタンスの起動、プライマリサーバへの昇格
- BACKUP: pg_basebackup
- ERROR: エラーで終了したイベント(PostgreSQLのエラーコードが00以外)。postgresql.confのlog_min_messagesパラメータがERROR以下に設定されている場合に使用可能。
- MISC: その他のコマンド (DISCARD、FETCH、CHECKPOINT、VACUUMなど)

ruleセクションの評価

- ログイベントが発生すると、ruleセクション内のすべての式が一度に評価されます。ログエントリは、ruleセクション内のすべてのパラメータがtrue(真)と評価された場合にのみ出力されます。

  たとえば、以下のルールが設定されていた場合、'apserver'が'myschema.tbl1'に対して行った操作のうち、午前10時から午前11時の間のクラス'WRITE'以外に対応する操作が監査ログとして出力されます。

  ```
  [rule]
  timestamp = '10:00:00-11:00:00'
  remote_host = 'apserver'
  object_name = 'myschema.tbl1'
  class != 'WRITE'
  ```

- pgaudit設定ファイルでは、ruleセクションを複数定義することができます。ログイベントは、それぞれのruleセクションによって評価され、ruleセクションが一致するごとに監査ログが出力されます。

  たとえば、以下のルールが設定されていた場合、重複した監査ログが出力されます。

  ```
  [rule]
  object_name = 'myschema.tbl1'
  [rule]
  object_name = 'myschema.tbl1'
  ```

- 1つのruleセクション内に同一のパラメータが複数指定されていた場合、最後に指定されているパラメータが有効になります。

  たとえば、以下のルールが設定されていた場合、“object_name = 'myschema.tbl3'”が有効になります。

  ```
  [rule]
  object_name = 'myschema.tbl1'
  object_name = 'myschema.tbl2'
  object_name = 'myschema.tbl3'
  ```

出力形式

Session Audit Loggingでは、監査ログが以下の形式で出力されます。

```
AUDIT: SESSION,READ,2022-03-12 19:00:58 PDT,
   (1)          (2)      (3)
[local],19944,psql,appuser,postgres,2/8,  2,  1,SELECT,,TABLE,myschema.account,  ,
  (4)     (5)    (6)    (7)       (8)    (9)(10)(11)(12)(13)(14)     (15)          (16)
SELECT * FROM myschema.account;,<not logged>
       (17)                        (18)
```

No	出力内容
(1)	ログのヘッダー。 “AUDIT: SESSION”で固定です。
(2)	クラス
(3)	SQLの開始時間
(4)	リモートホスト名。 ローカルホストの場合は、[local]が出力されます。
(5)	バックエンドプロセスID
(6)	アプリケーション名。 アプリケーション名が指定されていない場合は、[unknown]が出力されます。
(7)	ユーザー名

No	出力内容
(8)	データベース名
(9)	仮想トランザクションID
(10)	ステートメントID
(11)	サブステートメントID
(12)	コマンドタグ
(13)	SQLSTATE
(14)	オブジェクトタイプ
(15)	オブジェクト名
(16)	エラーメッセージ
(17)	SQL。 CREATE ROLEなどSQLにパスワードが含まれている場合は、<REDACTED>に置換えられます。 また、pgaudit設定ファイルのoptionセクションのlog_statement_onceパラメータに“on”を指定している場合、2回目以降は<previously logged>が出力されます。
(18)	pgaudit設定ファイルのoptionセクションのlog_parameterパラメータの値により、以下の出力内容となります。 ・ log_parameter=onを指定 SQLにパラメータを指定している場合は、パラメータの値が空白区切りで連結されて出力されます。 SQLにパラメータを指定していない場合は、<none>が出力されます。 ・ log_parameter=off(デフォルト)を指定 <not logged>が出力されます。 また、pgaudit設定ファイルのoptionセクションのlog_statement_onceパラメータに“on”を指定している場合、2回目以降は<previously logged>が出力されます。

参考

以下の機能を利用した資源にアクセスする場合、(12)のコマンドタグが???として出力される場合があります。

- INSTEAD OFトリガ
- RULE
- VIEW
- 行単位のセキュリティポリシー
- テーブルの継承

使用例

Session Audit Loggingで、監査ログを取得する例を示します。

1.設定

pgaudit設定ファイルに、以下のruleセクションを指定します。

```
[rule]
class = 'READ, WRITE'
object_name = 'myschema.account'
```

2.ログを取得

以下のSQLをクライアントから実行します。

```
CREATE TABLE myschema.account
(
    id int,
    name text,
    password text,
    description text
);
INSERT INTO myschema.account (id, name, password, description) VALUES (1, 'user1', 'HASH1', 'blah, blah');
SELECT * FROM myschema.account;
```

以下の監査ログが取得できます。

classパラメータに'DDL'が定義されていないため、CREATE TABLEは監査ログとして出力されません。

```
AUDIT: SESSION,WRITE,2022-03-12 19:00:49 PDT,[local],19944,psql,appuser,postgres,
2/7,1,1,INSERT,,TABLE,myschema.account,,"INSERT INTO myschema.account (id, name, password, description) VALUES (1,
'user1', 'HASH1', 'blah, blah');",<not logged>
AUDIT: SESSION,READ,2022-03-12 19:00:58 PDT,[local],19944,psql,appuser,postgres,
2/8,2,1,SELECT,,TABLE,myschema.account,,SELECT * FROM myschema.account;,<not logged>
```

6.5 Object Audit Logging

Object Audit Loggingでは、ロールを使用することで、監査ログの取得を実現しています。

optionセクションのroleパラメータで、監査ログ取得のためのロールを指定します。ロールに実行されたコマンドの権限がある場合、また、別のロールから権限が継承されている場合、そのコマンドの操作を監査ログとして出力します。

たとえば、optionセクションのroleパラメータに“auditor”を設定後、accountテーブル対するSELECT権限とDELETE権限を“auditor”に付与します。この場合、accountテーブルに対し、SELECTまたはDELETEが実行されると、監査ログが出力されます。

出力形式

Object Audit Loggingでは、監査ログが以下の形式で出力されます。

```
AUDIT: OBJECT,1,1,READ,SELECT,TABLE,public.account,SELECT password FROM account;,<not logged>
     (1)     (2)(3)(4)   (5)     (6)         (7)                (8)                    (9)
```

No	出力内容
(1)	ログのヘッダー。 “AUDIT: OBJECT”で固定です。
(2)	ステートメントID
(3)	サブステートメントID
(4)	クラス名
(5)	コマンドタグ
(6)	オブジェクトタイプ
(7)	オブジェクト名
(8)	SQL。 pgaudit設定ファイルのoptionセクションのlog_statement_onceパラメータに“on”を指定している場合、2回目以降は<previously logged>が出力されます。

No	出力内容
(9)	pgaudit設定ファイルのoptionセクションのlog_parameterパラメータの値により、以下の出力内容となります。 ・ log_parameter=onを指定 SQLにパラメータを指定している場合は、パラメータの値がカンマ区切りで連結されて出力されます。 SQLにパラメータを指定していない場合は、<none>が出力されます。 ・ log_parameter=off(デフォルト)を指定 <not logged>が出力されます。 また、pgaudit設定ファイルのoptionセクションのlog_statement_onceパラメータに“on”を指定している場合、2回目以降は<previously logged>が出力されます。

参考

以下の機能を利用した資源にアクセスする場合、(5)のコマンドタグが???として出力される場合があります。

- INSTEAD OFトリガ
- RULE
- VIEW
- 行単位のセキュリティポリシー
- テーブルの継承

例

Object Audit Loggingでログを取得する例を示します。

ロールに対し、権限を付与する対象を細かく設定することで、ログの出力を制御することができます。

以下の例では、accountテーブルのログ取得は、列に付与された権限で制御されますが、account_role_mapテーブルのログ取得は、テーブルに付与された権限に対して制御されます。

1.設定

pgaudit設定ファイルのoptionセクションに、以下のroleパラメータを指定します。

```
[option]
role = 'auditor'
```

2.ロールの定義

Object Audit Loggingのためのロールを定義します。

```
CREATE USER auditor NOSUPERUSER LOGIN;
```

3.ログを取得

以下のSQLをクライアントから実行します。

```
CREATE TABLE account
(
    id int,
    name text,
```

```
    password text,
    description text
);
GRANT SELECT (password) ON public.account TO auditor;
SELECT id, name FROM account;
SELECT password FROM account;
GRANT UPDATE (name, password) ON public.account TO auditor;
UPDATE account SET description = 'yada, yada';
UPDATE account SET password = 'HASH2';
CREATE TABLE account_role_map
(
    account_id int,
    role_id int
);
GRANT SELECT ON public.account_role_map TO auditor;
SELECT account.password, account_role_map.role_id
  FROM account
  INNER JOIN account_role_map ON account.id = account_role_map.account_id;
```

以下の監査ログが取得できます。

acoountテーブルでは、権限が付与されている列に対する操作のみがログとして出力されます。

account_role_mapテーブルでは、テーブルに対して権限が付与されているため、テーブルに対する操作がログとして出力されます。

```
AUDIT: OBJECT,4,1,READ,SELECT,TABLE,public.account,SELECT password FROM account;,<not logged>
AUDIT: OBJECT,7,1,WRITE,UPDATE,TABLE,public.account,UPDATE account SET password = 'HASH2';,<not logged>
AUDIT: OBJECT,10,1,READ,SELECT,TABLE,public.account,"SELECT account.password, account_role_map.role_id
        FROM account
        INNER JOIN account_role_map ON account.id = account_role_map.account_id;",<not logged>
AUDIT: OBJECT,10,1,READ,SELECT,TABLE,public.account_role_map,"SELECT account.password, account_role_map.role_id
        FROM account
        INNER JOIN account_role_map ON account.id = account_role_map.account_id;",<not logged>
```

6.6 データベース多重化運用の場合

データベース多重化運用時の監査ログ取得について説明します。

6.6.1 セットアップ

構築済みのデータベース多重化環境に監査ログ機能をセットアップする場合、以下の手順でセットアップしてください。

1. pgauditのファイルコピー

   プライマリサーバおよびスタンバイサーバでpgauditのファイルをコピーします。

   pgauditのファイルのコピーについては、“6.2 セットアップ”の手順1を参照してください。

2. pgaudit設定ファイルの作成

   プライマリサーバでpgaudit設定ファイルを作成します。また、作成したpgaudit設定ファイルをスタンバイサーバにコピーします。

   pgaudit設定ファイルの作成については、“6.2 セットアップ”の手順2を参照してください。

3. postgresql.confの設定

   プライマリサーバおよびスタンバイサーバのpostgresql.confに、監査ログを利用するためのパラメータを設定します。パラメータには、同じ値を設定してください。

   設定するパラメータについては、“6.2 セットアップ”の手順3および“6.6.2 監査ログ取得の設定”を参照してください。

4. Mirroring Controllerのサーバ識別子.confファイルの設定

   プライマリサーバおよびスタンバイサーバのサーバ識別子.confファイルに、監査ログを利用するためのパラメータを設定します。

設定するパラメータについては、“6.6.2 監査ログ取得の設定”を参照してください。

5. インスタンスの起動

   プライマリサーバおよびスタンバイサーバのインスタンスを起動します。

6. pgauditエクステンションの作成

   プライマリサーバでCREATE EXTENSIONを使用し、pgauditエクステンションを作成します。

   pgauditエクステンションの作成については、“6.2 セットアップ”の手順5を参照してください。

7. pgaudit設定ファイルのパラメータ設定

   プライマリサーバでpgaudit設定ファイルのパラメータを追加/変更します。追加/変更したpgaudit設定ファイルは、スタンバイサーバにコピーします。

   設定するパラメータについては、“6.3 pgaudit設定ファイル”および“6.6.2 監査ログ取得の設定”を参照してください。

8. インスタンスの再起動

   プライマリサーバおよびスタンバイサーバのインスタンスを再起動します。

6.6.2 監査ログ取得の設定

データベース多重化運用では、多重化の状態確認や障害検知のため、Mirroring Controllerが定期的にデータベースにアクセスします。これにより、監査ログも定期的に取得され、ログファイルが枯渇してしまいます。このため、以下のパラメータを設定して、Mirroring Controllerによる監査ログを取得しないようにします。

postgresql.conf

log_connections

省略または“off”を指定します。

log_disconnections

省略または“off”を指定します。

Mirroring Controllerのサーバ識別子.confファイル

target_db

template1を指定します。

注意

新規にデータベースを作成する場合は、Mirroring Controllerを停止してから作成するか、テンプレートデータベースにtemplate1以外を指定して作成してください。

audit_log_disconnections

optionセクションにaudit_log_disconnections = “on”を指定します。

pgaudit設定ファイル

ruleセクションのdatabase

database != 'template1'を指定します。

6.7 SQLでの監査ログの参照

監査ログは、追加モジュールのfile_fdwを使用することで、SQLでアクセスすることができます。ここでは、専用ログファイルに出力したSession Audit Loggingを例に監査ログを参照する方法を説明します。

1. file_fdwのインストール

   CREATE EXTENSIONを使用し、file_fdwをエクステンションとしてインストールします。

```
$ psql
=# CREATE EXTENSION file_fdw;
=# ¥dx
                  List of installed extensions
Name     | Version |   Schema   |     Description
---------+---------+------------+-----------------------------------------
file_fdw | 1.0     | public     | foreign-data wrapper for flat file access
pgaudit  | 1.0     | public     | provides auditing functionality
plpgsql  | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

2. 外部サーバの作成

   CREATE SERVERを使用し、file_fdwで管理する外部サーバを作成します。

```
$ psql
=# CREATE SERVER auditlog FOREIGN DATA WRAPPER file_fdw;
```

3. 監査ログのテーブルの作成

   CREATE FOREIGN TABLEを使用し、監査ログのテーブルのカラム、CSVファイル名とフォーマットを定義します。

```
$ psql
=# CREATE FOREIGN TABLE auditlog (
header text,
class text,
sql_start_time timestamp with time zone,
remote_host_name text,
backend_process_id text,
application_name text,
session_user_name text,
database_name text,
virtual_transaction_id text,
statement_id text,
substatement_id text,
command_tag text,
sqlstate text,
object_type text,
object_name text,
error_message text,
sql text,
parameter text
) SERVER auditlog
OPTIONS ( filename '/database/inst1/pgaudit_log/pgaudit-2022-03-12.log', format 'csv' );
```

注意

監査ログファイルをローテーションさせて複数の監査ログファイルが存在する場合は、それぞれの監査ログファイルに対してテーブルを作成する必要があります。

4. 監査ログの参照

   SELECTを使用し、監査ログを参照します。

```
$ psql
=# SELECT * FROM auditlog;
     header     |  class  |     sql_start_time     | remote_host_name | backend_process_id ...
----------------+---------+------------------------+------------------+-------------------- ...
AUDIT: SESSION  | WRITE   | 2022-03-12 19:00:49+09 | [local]          | 19944              ...
AUDIT: SESSION  | READ    | 2022-03-12 19:00:58+09 | [local]          | 19944              ...
```

6.8 アンセットアップ

pgauditのアンセットアップ方法を説明します。

1. インスタンスの起動

   インスタンスを起動します。

2. pgauditエクステンションの削除

   DROP EXTENSION文を使用して、データベースからpgauditエクステンションを削除します。

```
$ psql -d <database name>
=# DROP EXTENSION pgaudit;
=# ¥q
```

3. postgresql.confの変更

   pgauditに関する以下のパラメータを削除します。

   - shared_preload_libraries
   - pgaudit.config_file

4. インスタンスの再起動

   インスタンスを再起動します。

5. pgauditのファイルの削除

   スーパーユーザーで以下のコマンドを実行します。“<*x*>”は、製品のバージョンを示します。

```
$ su -
Password:******
# rm -rf /opt/fsepv<x>server64/セットアップ時にコピーしたファイル
```

参考

セットアップ時にコピーしたファイルは以下で確認することができます。

```
# find /opt/fsepv<x>server64/OSS/pgaudit
```

第7章 機密管理支援機能

データベースに格納される種々のデータが不正に利用されることを防ぐには、データベースの利用者に対して、各データベース資源に対する適切な権限を設定することが必要です。しかし権限を与える利用者の数は複数あり、また、対象とするデータベース資源の数も多くあります。そのため、その設定を行うには大変な手間がかかります。機密管理支援機能では、そのための手間を軽減し適切な権限を設定・維持するための支援を行います。

本章では、機密管理支援機能の導入、設計および使用方法について説明します。また、機能の利用例については、“7.8 機密管理支援機能の利用例”を参照してください。

7.1 セットアップ

本機能は、PostgreSQLのEXTENSIONとして提供されます。名前は、pgx_confidential_management_supportです。以下のようにCREATE EXTENSION文を使って、拡張をデータベースクラスタに登録してください。

これはスーパーユーザーが実行しなければなりません。なぜならば、この拡張はPostgreSQLのイベントトリガを登録するからです。イベントトリガを登録することで、テーブルなどのデータベースオブジェクトが削除されたときに、機密管理支援機能が管理する情報の中から、関連する情報を削除します。

この拡張は、どのスキーマに対しても作成することができます。

```
CREATE EXTENSION pgx_confidential_management_support
```

注意

- 以降で説明する様々な定義は、この拡張に含まれるテーブルに登録されます。従って、この拡張をDROP EXTENSION文によって削除すると、これらの定義もすべて削除される点に注意してください。
- スーパーユーザーが機密管理ロールを兼ねる場合や、機密管理ロールを1つだけ用意する場合には、セットアップは完了です。機密管理ロールについては、“7.2.2 機密管理ロールを決める”を参照してください。しかし、もし複数の機密管理ロールが、それぞれ異なる機密マトリクスを管理するならば、互いの機密マトリクスを互いに操作できないようにしなければなりません。そのために、以下のように、スーパーユーザーが、製品が提供するスクリプトを実行してください。このスクリプトは、機密管理支援機能が提供するテーブルに、行レベルセキュリティ機能のポリシーを定義します。${install_dir}は、製品をインストールしたディレクトリを指します。

  ```
  psql -f ${install_dir}/share/extension/pgx_confidential_management_support_policy.sql
  ```

- CREATE EXTENSIONを実行すると、この拡張に含まれるテーブルについて、publicへSELECT権限が付与されています。publicからこの権限を剥奪しないでください。例えば、一般ユーザーがテーブルを削除したときに、機密管理支援機能の定義も変更するべきかをイベントトリガが確認します。この確認のために必要な権限です。また、pg_catalogがそうであるように、これらのテーブルの情報をすべてのユーザーが知ったとしても、問題はありません。

7.2 機密管理の設計

機密管理の設計について説明します。

7.2.1 機密マトリクスを設計する

機密マトリクスとは、機密レベルと機密グループのマトリクスです。機密マトリクスの要素は機密権限です。ここでは、これらを定義します。

機密レベルは同じ機密性の度合いを持つデータのグループであり、機密グループは機密性のあるデータに対して同じアクセス権限を持つロールのグループです。詳細は、以降で説明します。

最終的には、データベースオブジェクトを機密レベルに分類し、ロールを機密グループに分類します。しかし、最初のステップでは、具体的なテーブルやロールなどを意識せずに、機密マトリクスを抽象的に定義してください。そうすることによって、機密レベルと機密グループの関係を、異なるデータベースオブジェクト群を持つデータベースに適用することもできます。

複数の機密マトリクスを定義することができます。これらの機密マトリクスには名前を付けて識別しますが、その名前はデータベース(データベースクラスタではなく)の中で一意です。

機密マトリクスを管理するロールは、機密管理ロールと呼びます。後で詳細に説明します。

複数の機密マトリクスを作るときの考え方

様々な設計の方法がありますが、下記に例を示します。

- データ集合に対して1つの機密マトリクスを作る

  例えば、組織の1つのセクションに対して1つの機密マトリクスを作成します。

- 2つのセクションの機密の設計が同じであるならば、機密マトリクスをコピーする
- 複数の組織がデータ集合を共有するときには、共有データ集合に対して1つの機密マトリクスを作成する

  この場合の例を以下に示します。

  ― セクション1だけがアクセスするデータ集合のための機密マトリクスAを作成する。

  ― セクション2だけがアクセスするデータ集合のための機密マトリクスBを作成する。

  ― セクション1とセクション2が共有するデータ集合のための機密マトリクスSを作成する。

  ― セクション1の管理者M1が、機密マトリクスAの機密管理ロールを担当する。

  ― セクション2の管理者M2が、機密マトリクスBの機密管理ロールを担当する。

  ― M1とM2とが属するロールグループGが、機密マトリクスSの機密管理ロールを担当する。

7.2.1.1 機密レベルを定義する

機密レベルとは、どのくらい外部に漏洩してはならないデータの集合なのかを示す概念です。例えば、個人情報を含むテーブルと含まないテーブルを分類するような機密レベルを決めます。

機密レベルに分類するデータの集合は、テーブルのようなデータベースのオブジェクトです。このようなオブジェクトを機密オブジェクトと呼びます。機密オブジェクトについては、“機密オブジェクトとは”を参照してください。

機密レベルには属性を設定することができます。例えば、機密レベルに暗号化を要求する属性を設定したならば、その機密レベルに属するテーブルは、暗号化されることを要求されます。機密管理支援機能が、所属する機密オブジェクトの属性を自動的に変更する場合としない場合があります。また、属性ごとに、ターゲットとなる機密オブジェクトは異なります。

以下の表は、属性の説明、ターゲットとなる機密オブジェクト、および機密オブジェクトの属性を変更するかどうかを示しています。

機密レベルの属性	指定できる値	機密オブジェクトの型	自動変更	説明
encryption_algorithm	AES128 AES256 none	table column rowset	No	機密オブジェクトが、指定されたアルゴリズムで暗号化されていることを要求します。詳細は、“運用ガイド”の“透過的データ暗号化による格納データの保護”を参照してください。 noneが指定されると暗号化を要求しません。 デフォルトはnoneです。 暗号化と復号は、大規模なデータ更新を伴うので自動的に属性を変更しません。そのかわり、指定された暗号化強度よりも低い暗号化強度の機密オブジェクトを追加することはできません。また、暗号化強度を強くするように変更するときに、変更後の条件を満たさない機密オブジェクトが含まれていてはいけません。

機密オブジェクトとは

機密オブジェクトには次の表に示したような型があります。例えば、テーブルTに含まれる一部のデータを機密レベルL1に追加し、そのほかのデータを機密レベルL2に追加することもできます。

機密オブジェクトの型	説明
schema	スキーマに含まれるすべてのテーブルを意味しません。単にシステムカタログに存在するオブジェクトを意味します。GRANT文によるアクセス制御のターゲットとなるスキーマ自身を意味します。
table	テーブルだけでなく、ビュー、マテリアライズドビュー、パーティションを含みます。今は、外部テーブルをサポートしていません。
column	カラムを意味します。
rowset	指定された条件を満たすような行の集合です。 PostgreSQLのシステムカタログにはrowsetを意味するオブジェクトは存在しませんが、機密管理支援機能では、使い易さのために機密オブジェクトとして扱われます。 rowset型の機密オブジェクトのアクセス制御は、PostgreSQLの行レベルセキュリティの機能を内部的に使います。よって、どのようにアクセスを制御するかは行レベルセキュリティの仕様に従います。

以下のオブジェクトにも機密にするべき情報が含まれる可能性があります。しかし、現在の機密管理支援機能は機密オブジェクトとしてサポートしていません。これらを管理するときには、機密管理ロールとは異なるロールで管理することをお勧めします。詳細は、“7.5監視方法の提案”で説明しています。

- ファンクション
- プロシージャ
- 外部サーバー
- 外部データラッパー
- 外部テーブル

7.2.1.2 機密グループを定義する

機密グループとは、どの機密レベルにアクセスしてよいロールなのかを示すオブジェクトです。例えば、個人情報にアクセスしても良いロールのグループとそうではないグループを決めます。機密グループは、実際にはPostgreSQLのロール（ロールグループ）として自動的に定義されます。このロールを機密グループロールと呼びます。

機密グループロールとは

機密グループロールは、機密管理支援機能が内部で実行するGRANT文やREVOKE文のターゲットです。そして、機密グループにロールを追加すると、追加されたロールは機密グループロールのメンバーに加えられます。追加されたロールの権限は、機密管理支援機能が機密グループロールに与えた権限を継承する形で与えられます。

機密グループを作成する関数を介して、機密グループロールには以下の属性を設定することができます。属性の意味とデフォルトの値は、CREATE ROLE文の仕様と同じです。これらは、機密を管理するときに必要であろう属性に限定しています。ですから、機密管理支援機能を使って変更してください。機密管理支援機能が、機密グループに追加されたロールに対して、以下の属性を自動的に設定します。

- SUPERUSER
- CREATEDB
- CREATEROLE
- REPLICATION
- BYPASSRLS

これら以外の属性は、PostgreSQLのCREATE ROLE文のデフォルトに従います。もし、ALTER ROLE文を使って属性を変更しても、機密管理支援機能の動作は妨げられません。

機密グループロールの命名規則は以下のとおりです。

```
pgx_cgroup_role_${serial_number}
```

機密グループを削除するときに、機密グループロールを削除しないことを選択できます。そうしたときに、後から機密グループロールであったことを知る手段になります。

この命名規則にしたがうような文字列を使ったロールを定義しないでください。なぜならば、この命名規則にしたがうようなロールの${serial_number}の最大値の次の数値を新たな機密グループロールの名前に使うからです。例えば、もし、pgx_cgroup_role_9999999999999999999という名前のロールが存在すると、新しい機密グループロールを作成できなくなります。

7.2.1.3 機密権限を定義する

機密権限とは、ある機密グループが、ある機密レベルに登録された機密オブジェクトへ、どのようにアクセスできるかを示します。

例えば、機密グループRが、機密レベルLに属するtable型の機密オブジェクトへSELECT、INSERT、UPDATE、DELETE権限を持つように定義することができます。

機密オブジェクトやロールが、機密レベルや機密グループに追加されたならば、機密権限の定義にしたがって、本機能が自動的にGRANT文またはCREATE POLICY文を使用して、機密グループロールに実際のデータベースのオブジェクトへの権限を付与します。

rowset型の場合には、機密権限は、CREATE POLICY文で指定できる権限です。rowset以外の場合には、GRANT文で指定できる権限です。詳細な権限は、“PostgreSQL Documentation”のそれぞれのSQL文のリファレンスを参照してください。

この機能が内部的に作成したPOLICYの定義を変更しないでください。もし、変更しても、この機能の関数が、警告なく変更前の状態へ戻すことがあるからです。

機密マトリクスに登録されているロールに対して、ある機密オブジェクトへの権限が機密権限よりも広く与えられていたならば、機密権限に合わせてそれらの権限を剥奪します。

機密マトリクスに登録されて***いない***ロールまたはPUBLICに対して、機密マトリクスに登録されている機密オブジェクトへの権限が付与されていたならば、この機能は、そのようなロールから権限を剥奪せずに、機密オブジェクトやロールを追加するための関数が失敗します。この機能が管理することを許されていないからです。

以下のように、もう少し複雑な状況であっても、機密オブジェクトやロールを追加する関数が失敗することで、機密マトリクスに追加されたロールが機密権限を逸脱しないようにします。

- 機密マトリクスに登録されていないロールを機密マトリクスに登録されたロールが継承できるように設定されていて、かつ、結果的に機密権限よりも大きな権限を持っている。
- 機密マトリクスに登録されているロールに対して、この機能が作成していないPOLICYによって権限が与えられている。このケースでは、機密権限を逸脱しているかを厳密には検査せずに、このようなPOLICYが存在していたならば、関数が失敗します。

7.2.2 機密管理ロールを決める

機密管理ロールは、機密マトリクスに対するすべての操作を行います。そのため、機密管理ロールは、以下のいずれかの非常に強い権限を要求します。

1. 以下のすべての権限を持つ
   - スーパーユーザー権限
   - 機密レベルに属するデータベースオブジェクトの所有権

2. 以下のすべての権限を持つ

   ー CREATEROLE権限

   ー 拡張に含まれるすべてのテーブルへのSELECT権限、INSERT権限、UPDATE権限、および、DELETE権限
   CREATE EXTENSIONを実行すると、publicにSELECT権限が付与されています。

   ー 機密レベルに属するデータベースオブジェクトの所有権

注意

機密管理ロールが所有者になると、前の所有者はGRANT文などを実行できなくなる可能性があることに注意してください。なぜならば、データベースオブジェクトの所有権は、異なるロールグループに属する複数のロールで共有することができないからです。これはPostgreSQLの仕様です。

機密管理ロールは複数の機密マトリクスを作成し管理することができますが、権限を分散した方が安全です。そのために、以下の優先順位で機密管理ルールを決めることをお勧めします。

- 機密マトリクスと機密管理ロールを1対1で作成する。
- 複数の機密管理ロールをメンバーに持つロールグループを作成する、あるいは複数の機密マトリクスをひとつの機密管理ロールが管理する。
- スーパーユーザーが機密管理ロールを兼ねる。

注意

機密管理支援機能を使って管理するロールに、スーパーユーザーだけが付与できる属性を与える場合には、スーパーユーザーが機密管理ロールを兼ねる必要があります。例えば、REPLICATION属性が、そのような属性に該当します。詳細は、PostgreSQL Documentationのリファレンスを参照してください。

7.2.3 機密レベルの定義に従って、機密オブジェクトを分類する

7.2.3.1 機密オブジェクトを定義する

まず、管理対象にしたいスキーマ、テーブルなどの定義の一覧を参照して、機密オブジェクトを定義してください。前述したように、機密オブジェクトの型にはcolumn型やrowset型などの多様な型が用意されているので、データの内容に基づいて機密オブジェクトを定義することができます。

rowset型の機密オブジェクトのアクセス制御は、PostgreSQLの行レベルセキュリティの機能を内部的に使います。ですから、行の集合を指定する方法は、CREATE POLICY文のAS句、USING句、WITH CHECK句の仕様に従います。また、rowset型の機密オブジェクトの定義は、rowset型の機密オブジェクトを機密レベルに登録する関数の引数で指定します。この機能は、rowset型の機密オブジェクトが追加されたときに、POLICYを有効化するために、ENABLE ROW LEVEL SECURITY 句を指定してALTER TABLE文を実行します。

今は、table型の機密オブジェクトとして外部テーブルを登録することはできません。

ポイント

機密オブジェクトへの権限をPUBLICに付与しないことを推奨します。PUBLICへ権限を付与することは、機密マトリクスに登録されたすべてのロールへ権限を与えていることと同じです。機密オブジェクトにアクセスするすべてのロールを、この機能を使って管理するならば、このようなことは意味がありません。もしPUBLICに機密オブジェクトへの権限を付与しても、この機能に含まれる関数は、各ロールに許された権限を超えないように検査し、超えていたならば失敗します。

注意

- 機密オブジェクトがcolumn型のときには注意が必要です。なぜならば、table型を同時に設定すると、table型の権限が優先されるからです。

  例えば、あるテーブルTの特別なカラムCだけからSELECT権を剥奪したいならば、テーブルTへSELECT権限を付与せずに、カラムC以外のカラムを列挙して、それらにSELECT権を付与してください。このことは、PostgreSQLの列への権限付与の仕様に従っています。

  非常に多くのカラムを列挙しなければならないならば、特別なカラムを新しいテーブルに移動させ、既存のアプリケーションに両方のテーブルをJOINするようなVIEWを見せるのも良いアイディアかもしれません。

- 機密オブジェクトがrowset型のときには、rowset型に指定する権限と同じ権限をtable型にも設定してください。SQL文でデータにアクセスするときには、最初にテーブルへのアクセス権限があることがチェックされるからです。その後に、条件に合致した行であったときにrowsetの権限がチェックされます。このことは、PostgreSQLの行レベルセキュリティの仕様に従っています。

7.2.3.2 機密オブジェクトを分類する

機密オブジェクトを機密レベルの定義にしたがって分類してください。

異なる機密マトリクスの機密レベルであったとしても、ひとつの機密オブジェクトを複数の機密レベルに分類することはできません。

7.2.4 機密グループの定義に従ってロールを分類する

ロールは、複数の異なる機密マトリクスの機密グループに同時に所属することができます。しかし、1つのロールを、1つの機密マトリクスの中の複数の機密グループに分類することはできません。

注意

- 機密グループロールを、他の機密グループに分類しないことを強く勧めます。なぜならば、機密管理支援機能はこのような使い方を禁止しませんが、おそらく機密管理が複雑になるだけだからです。また、ロールのグルーピングが循環することをPostgreSQLが許可していないことにも注意してください。

- 機密グループにロールを追加した後に、このようなロールを機密マトリクスで管理されていないロールグループのメンバーにしないことを強く勧めます。なぜならば、機密管理支援機能はこのような状況を禁止しませんが、例えば、そのようなロールグループの権限を不当に強くすることが、機密管理支援機能で管理されたロールの抜け道になるかもしれないからです。
  なお、この機能は、このような抜け道が設定されたロールを機密グループに追加することを許可しません。

7.3 機密管理支援機能の使用方法（定義）

機密管理支援機能は、関数とテーブルを提供します。

一部の関数は、機密マトリクスなどを定義、変更、あるいは削除します。これらの関数は、それ自身が実行されたことを示す監査ログを出力するので、不正な操作が行われたことを後から確認することができます。

また、この機能が提供するテーブルを直接的に参照したり、参照を助ける関数を使って定義した内容を確認したりすることができます。

また、一部の関数は、PostgreSQLのシステムカタログに定義されている機密オブジェクトの属性と、その機密オブジェクトに設定されているべき属性を出力します。これらを使用することで属性を比較できます。

詳細は、“付録B 機密管理支援機能が利用するシステム管理関数”を参照してください。

pgx_confidential_management_support拡張が含むテーブル

テーブル名	説明
pgx_confidential_matrix	機密マトリクスの一覧です。 登録された機密マトリクスの属性、更新時刻、あるいは、機密レベルや機密グループを登録または削除した時刻などを参照できます。
pgx_confidential_level	機密レベルの一覧です。

テーブル名	説明
	登録された機密レベルの属性、更新時刻、あるいは、機密レベルに機密オブジェクトを登録または削除した時刻などを参照できます。
pgx_confidential_group	機密グループの一覧です。 登録された機密グループの属性、更新時刻、あるいは、機密グループにロールを登録または削除した時刻などを参照できます。
pgx_confidential_privilege	機密権限の一覧です。 機密オブジェクトごとに設定された機密権限、あるいは、更新時刻などを参照できます。
pgx_confidential_object	機密オブジェクトの一覧です。 オブジェクトの属性、あるいは、更新時刻などを参照できます。
pgx_confidential_role	機密グループに登録されたロールの一覧です。 ロールの属性、あるいは、更新時刻などを参照できます。
pgx_confidential_policy	rowset型の機密オブジェクトに権限を設定するために作成したポリシーの一覧です。作成したポリシーの名前、設定している権限を参照できます。 このテーブルの行は、rowset型の機密オブジェクトを追加したときに挿入されます。

定義を追加、削除または更新するための関数を実行すると、各テーブルの行が追加、削除または更新されます。

また、対象の機密オブジェクトがDROP TABLE文などにより削除された場合にも、以下のテーブルから削除されます。

- pgx_confidential_object
- pgx_confidential_policy

定義を追加、削除または更新するための関数

これらの関数は、それ自身が実行されたことを示す監査ログを出力します。

関数名	説明
pgx_create_confidential_matrix	機密マトリクスを作成します。
pgx_alter_confidential_matrix	機密マトリクスの属性を変更します。
pgx_drop_confidential_matrix	機密マトリクスを削除します。
pgx_copy_confidential_matrix	機密マトリクスを複製します。
pgx_create_confidential_level	機密レベルを作成し、機密マトリクスに登録します。
pgx_alter_confidential_level	機密レベルの属性を変更します。
pgx_drop_confidential_level	機密マトリクスから機密レベルを除去し、機密レベルを削除します。
pgx_create_confidential_group	機密グループを作成し、機密マトリクスに登録します。
pgx_alter_confidential_group	機密グループの属性を変更します。
pgx_drop_confidential_group	機密マトリクスから機密グループを除去し、機密グループを削除します。
pgx_grant_confidential_privilege	機密権限を付与します。
pgx_revoke_confidential_privilege	機密権限を剥奪します。
pgx_add_object_to_confidential_level	機密レベルに機密オブジェクトを追加します。
pgx_remove_object_from_confidential_level	機密レベルから機密オブジェクトを除去します。

関数名	説明
pgx_add_role_to_confidential_group	機密グループにロールを追加します。
pgx_remove_role_from_confidential_group	機密グループからロールを除去します。

定義の参照およびシステムカタログとの比較を支援する関数

関数名	説明
pgx_get_attribute_of_objects	指定した機密マトリクスに登録されているすべての機密オブジェクトに関して、機密マトリクスで規定されている属性と、実際にデータベース上で設定されている属性を表示します。
pgx_get_attribute_of_roles	指定した機密マトリクスに登録されているすべてのロールについて、機密マトリクスにて規定されている属性と、実際にシステムカタログで設定されている属性とを表示します。
pgx_get_privileges_on_level_and_group	指定された機密レベルに登録されている機密オブジェクトと、指定された機密グループに登録されているロールの組み合わせについて、以下を比較できるような一覧を表示します。 ・ 指定されたロールに付与されるべき、機密権限の設定値によって規定された権限 ・ 実際のシステムカタログで付与されている権限
pgx_get_privileges_on_object	指定された機密オブジェクトについて、以下を比較できるような一覧を表示します。 ・ すべてのロールに付与されるべき、機密権限の設定値によって規定された権限 ・ 実際のシステムカタログで付与されている権限
pgx_get_privileges_on_role	すべての機密オブジェクトについて、以下を比較できるような一覧を表示します。 ・ 指定されたロールに付与されるべき、機密権限の設定値によって規定された権限 ・ 実際のシステムカタログで付与されている権限
pgx_get_privileges_on_matrix	指定された機密マトリックスに登録されたすべてのオブジェクトについて、以下を比較できるような一覧を表示します。 ・ 機密マトリクスに登録されたすべてのロールに付与されるべき、機密権限の設定値によって規定された権限 ・ 実際のシステムカタログで付与されている権限

定義の手順を説明します。

7.3.1 機密管理ロールを作成する

CREATE ROLE文を使って機密管理ロールを作成するか、既存のロールを機密管理ロールとして利用してください。機密管理ロールには、“7.2.2 機密管理ロールを決める”で示した権限や属性を設定してください。

注意

- 機密管理ロールの名前を変更するときには注意が必要です。もし、そのようにしたいならば、その機密管理ロールが管理する機密マトリクスをすべて削除し、機密管理ロールの名前を変更し、その後でもう一度同じ機密マトリクスを定義してください。そうしなければ、機密

マトリクスを操作できなくなります。例えば、機密権限を変更したり、機密オブジェクトを機密レベルから削除したりすることができません。もし、誤って、先に名前を変更してしまったならば、もう一度名前を元に戻してから、前に述べたように操作してください。将来は、機密管理ロールの変更後の名前を、単に機密マトリクスに設定できるようにします。

- もし、機密管理ロールを削除する場合には、機密マトリクスを削除した後に、機密管理ロールを削除してください。そうしないと、残された機密マトリクスを削除できないので、同じ名前の機密マトリクスを作成できません。もし、誤って、機密マトリクスを削除する前に、機密管理ロールを削除してしまった場合には、同じ名前の機密管理ロールをもう一度作成して機密マトリクスを削除するか、SUPERUSER権限を持ったロールで機密マトリクスを削除してください。

7.3.2 機密マトリクスを作成する

以下のように、機密マトリクスを説明するコメントとともに機密マトリクスを作成します。このコメントは、pgx_confidential_matrixテーブルに保存されます。

```
select pgx_create_confidential_matrix('matrix_foo', 'This matrix is defined for foo.')
```

ポイント

この関数を実行したロールを、機密管理ロールとみなします。もし、SET ROLEした後にこの関数を実行したならば、SET ROLEを実行したロールではなく、SET ROLEに指定されたロールを機密管理ロールとみなします。

以下のように、既に作成されている機密マトリクス'matrix_src'を複製して、新しく機密マトリクス'matrix_dest'を作成することもできます。機密マトリクスを複製して作成すると、コメントも複製されます。pgx_alter_confidential_matrix関数を使って、コメントを変更することができます。

```
select pgx_copy_confidential_matrix('matrix_dest', 'matrix_src')
```

また、pgx_confidential_matrixテーブルを参照すると、作成したマトリクスを確認することができます。

ポイント

複製元は、同じデータベース内でなければなりません。もし、異なるデータベース、あるいは、異なるデータベースインスタンスから複製するときには、以下のいずれかの方法を選択してください。ただし、COPYを使用した方法は、以下に示す注意を確認後、実行してください。

- 複製元と同じように関数を使って定義する。
- 以下の手順を実行する

  手順6以外は、SUPERUSER権限を持つロールで実行してください。

  1. ターゲットのデータベースにおいて、拡張をセットアップします。
  2. 以下の表から、複製元の機密マトリクスのIDをWHERE句に指定して、COPY TO文を使ってデータを抽出します。
     - pgx_confidential_matrix
     - pgx_confidential_level
     - pgx_confidential_group
     - pgx_confidential_privilege
  3. pg_sequencesシステムビューを参照して、以下のSEQUENCEのlast_valueカラムの値を、ファイルなどに保存します。
     - pgx_confidential_matrix_cmatid_seq
     - pgx_confidential_level_clevid_seq
     - pgx_confidential_group_cgroid_seq
     - pgx_confidential_privilege_cpriid_seq
  4. 手順2で抽出したデータを、ターゲットのデータベースで、COPY FROM文を使用して上記のテーブルにロードします。この操作は、いずれかの機密マトリクスの機密管理ロールであれば実行できます。

5. 複製先の各SEQUENCEに対して、手順3で保存した値を指定して、pg_catalog.setval関数を使ってlast_valueを設定します。以下に例を示します。

```
SELECT pg_catalog.setval('pgx_confidential_matrix_cmatid_seq', 5)
```

6. pgx_copy_confidential_matrix関数を使って、手順4でロードした機密マトリクスを複製元に指定して、新しい機密マトリクスを作成します。この操作は、新しく作成する機密マトリクスの機密管理ロールで実行します。

7. 手順4でロードした機密マトリクスを、drop_roleにfalseを指定して、pgx_drop_confidential_matrix関数を使って削除します。

注意

- drop_roleにtrueを指定すると、複製元の機密マトリクスによる管理を続けられなくなります。理由は以下のとおりです。

  手順6が完了すると、以下の状態になっています。

  a) 複製元のデータベースの機密マトリクス

  b) 手順4でロードした機密マトリクス

  c) 手順6で複製した機密マトリクス

  b)とc)の機密マトリクスは、異なる機密グループロールを使用しています。これは、pgx_copy _confidential_matrix関数の効果です。しかし、a)とb)の機密マトリクスは、同じ機密グループロールを共有しています。これは一時的ですが、良くない状況です。そのために、b)の機密マトリクスを手順7で削除します。このときに、drop_roleにtrueを指定すると、a)で使用している機密グループロールが削除されます。よって、a)の機密マトリクスによる管理を続けられません。

- コピー先では、この拡張をCREATE EXTENSIONでインストールした直後でなければなりません。この拡張に含まれる表のデータの中には、機密マトリクスや機密レベルを表すIDが含まれていて、このIDは拡張に含まれるSEQUENCEによって生成されるからです。例えば、もし機密マトリクスをひとつでも生成していたならば、その機密マトリクスにはIDの1がナンバリングされますが、そのマトリクスはコピー元におけるIDの1がナンバリングされたオブジェクトではありません。

7.3.3 機密レベルを機密マトリクスに追加する

以下のように、機密レベルを作成して機密マトリクスに追加します。以下の例では、JSON形式の第3引数によって、'level1'に属する機密オブジェクトに対して、暗号化されていることを要求しています。何も要求することがなければ、NULLを指定してください。要求の内容については、“B.2 機密レベル操作関数”を参照してください。

```
select pgx_create_confidential_level('matrix_foo', 'level1', '{"encryption_algorithm":"AES256"}', 'The strongest
encryption is required for this level.')
```

また、pgx_confidential_levelテーブルを参照すると、追加した機密レベルを確認することができます。

7.3.4 機密グループを機密マトリクスに追加する

以下のように、機密グループを作成して機密マトリクスに追加します。JSON形式の第3引数によって、機密グループに属性を付与します。付与できる属性については、“B.3 機密グループ操作関数”を参照してください。

以下の例では、この機密グループに属するロールは、CREATEDB権限を付与します。

```
select pgx_create_confidential_group('matrix_foo', 'group1', '{"CREATEDB":true}', 'Roles belonging to this confidential
role are permitted to create db.')
```

また、pgx_confidential_groupテーブルを参照すると、追加した機密グループを確認することができます。

7.3.5 機密権限を機密グループに付与する

以下の例では、group1に属するロールには、level1に属するテーブルに対するSELECT権限、INSERT権限およびDELETE権限を付与しています。JSON形式の第4引数によって、機密オブジェクトの型ごとに権限を指定することができます。付与できる権限については、“B.4 機密権限操作関数”を参照してください。

```
select pgx_grant_confidential_privilege('matrix_foo', 'level1', 'group1',  '{"schema":["ALL"], "table":
["SELECT","INSERT","DELETE"]}')
```

機密レベルに対象の種別の機密オブジェクトが登録されていたならば、この関数を実行したときに、機密管理支援機能が内部的にGRANT文を使って、機密レベルに登録されている機密オブジェクトへのアクセス権限を機密グループロールに付与します。GRANT文が作用した結果は、以下のように確認することができます。

```
select pgx_get_privileges_on_level_and_group('matrix_foo', 'level1', '["role1","role2"]')
```

この関数のような、権限情報を出力する関数が返却する表の形式は、“B.7 定義の参照およびシステムカタログとの比較を支援する関数”を参照してください。

7.3.6 機密レベルに機密オブジェクトを追加する

機密レベルの設計にしたがって、対象としているデータベースオブジェクトを適切な機密レベルに分類してください。以下の例では、schema1のtable1とschema1のtable2を同時にlevel1に追加しています。JSON形式の第3引数に、機密オブジェクトの型と機密オブジェクトの名前を指定します。機密オブジェクトの型ごとに登録したい複数のオブジェクトを列挙できます。この例では、table型だけを登録していますが、同時にschema型なども登録することができます。

```
select pgx_add_object_to_confidential_level ('matrix_foo', 'level1',
'[{
    "type":"table",
    "object":[
        {
          "schema":"schema1",
          "table":["table1","table2"]
        }
    ]
}]')
```

rowsetを指定するときは、どのような行の集合であるかをrowset_expressionキーの値において宣言しなければなりません。指定方法の詳細は、“B.5 機密オブジェクト操作関数”を参照してください。

この関数を実行すると、機密管理支援機能が内部的にGRANT文またはCREATE POLICY文を使って、指定された機密オブジェクトへの権限を、機密マトリクスに設定されているすべての機密グループに付与します。GRANT文が作用した結果は、以下のように確認することができます。この例では、schema1のtable1とschema1のtable2に付与された権限を確認しています。

```
select pgx_get_privileges_on_object('matrix_foo',
'[{
    "type":"table",
    "object":[
          "schema":"schema1",
          "table":["table1","table2"]
    ]
}]')
```

7.3.7 機密グループにロールを追加する

機密グループの設計にしたがって、対象としているロールを適切な機密グループに分類してください。複数のロールを同時に追加することができます。以下の例では、機密グループgroup1に、ロールrole1とロールrole2を追加しています。

```
select pgx_add_role_to_confidential_group('matrix_foo', 'group1', '["role1","role2"]')
```

機密管理支援機能は、指定されたロールを、機密グループロールのメンバーに加えます。

このときには機密オブジェクトに対するGRANT文を実行しません。既に機密グループロールと機密オブジェクトとの間のGRANT文は実行済みであり、個々のロールと機密オブジェクトとの間のGRANT文は実行しないからです。

以下の関数を使用すると、ロール側のINHERIT属性によって機密グループロールの権限を継承したり、SET ROLE文で機密グループロールに変更したりした後に、指定したロールが行使できる権限を確認することができます。つまり、実際に登録されたロールが行使す

ることができるアクセス権限を確認することができます。出力フォーマットの詳細は、“B.7 定義の参照およびシステムカタログとの比較を支援する関数”を参照してください。以下の例では、ロール'role1'とロール'role2'の権限を確認しています。

```
select pgx_get_privileges_on_role('matrix_foo', '["role1","role2"]')
```

7.4 機密管理支援機能の使用方法（変更と削除）

7.4.1 機密オブジェクトの名前を変更する

現在の機密管理支援機能は、機密オブジェクトの名前が変更されたことをフォローできません。したがって、名前を変更した場合には、古い名前を指定したpgx_remove_object_from_confidential_level関数で古い名前の機密オブジェクトを除外し、新しい名前を指定したpgx_add_object_to_confidential_level関数で新しい名前の機密オブジェクトを登録してください。

7.4.2 ロールの名前を変更する

現在の機密管理支援機能は、ロールの名前が変更されたことをフォローできません。したがって、名前を変更するときは、古い名前を指定したpgx_remove_role_from_confidential_group関数で古い名前のロールを除外し、ロールの名前を変更後、新しい名前を指定したpgx_add_role_to_confidential_group関数で新しい名前のロールを登録してください。

7.4.3 ロールを削除する

現在の機密管理支援機能は、ロールが削除されたことをフォローできません。したがって、ロールを削除したならば、pgx_remove_role_from_confidential_group関数を使って、そのロールを機密グループから除外してください。

7.4.4 機密マトリクスを変更する

以下のように、機密マトリクスの名前や属性を変更します。名前や属性の変更後の値をJSON形式で指定します。名前や複数の属性を1回で変更することができます。ここで指定されなかった属性は変更されません。

```
select pgx_alter_confidential_matrix('matrix_foo', '{"name": "matrix_bar", "comment":"This matrix is defined for bar."}')
```

7.4.5 機密マトリクスを削除する

以下のように、機密マトリクスを削除します。

```
select pgx_drop_confidential_matrix('matrix_foo', false, false)
```

第2引数や第3引数を指定すると、機密マトリクスに追加されたすべての機密レベルと機密グループを削除したり、機密グループロールを削除したりすることを選択できます。削除しなかった機密グループロールは、前に述べた命名規則を使って特定することができます。どのような選択をしても機密グループに登録されたロールは削除しません。上記の例では、単純に機密マトリクスだけを削除します。

7.4.6 機密レベルを変更する

以下のように、機密マトリクス'matrix_foo'に追加していた機密レベルの名前や属性を変更します。名前や属性の変更後の値をJSON形式で指定します。名前や複数の属性を1回で変更することができます。ここで指定されなかった属性は変更されません。

```
select pgx_alter_confidential_level('matrix_foo', 'level1', '{"name": "levelX", "comment":"This level required the
highest confidential clearance."}')
```

すでに機密レベルに機密オブジェクトが登録されていた場合には、機密オブジェクトの属性も自動的に変更されます。ただし、“7.2.1.1 機密レベルを定義する”で述べたように、自動的に機密オブジェクトの属性を変更できないこともあります。そのときには、この関数は、登録されている機密オブジェクトの属性が、変更後の機密レベルの属性よりも厳しいことをチェックします。チェックが失敗したならば、この関数も失敗します。例えば、encryption_algorithmをAES128からAES256に変更しようとしたときに、AES128で暗号化されたテーブルが機密オブジェクトとして登録されていたならば、この関数は失敗します。このようなときには、テーブルをAES256で暗号化した後に、本関数もう一度実行してください。

7.4.7 機密レベルを削除する

以下のように、機密レベルを削除します。

```
select pgx_drop_confidential_level('matrix_foo', 'level1', false)
```

第3引数を指定すると、指定された機密レベルに依存するものを削除することを選択できます。当然ですが、機密レベルに登録されていた機密オブジェクトは削除されません。

7.4.8 機密グループを変更する

以下のように、機密マトリクス'matrix_foo'に追加していた機密グループを変更します。名前や属性の変更後の値をJSON形式で指定します。名前や複数の属性を1回で変更することができます。ここで指定されなかった属性は変更されません。

```
select pgx_alter_confidential_group('matrix_foo', 'group1', '{"name": "groupX", "comment":"Members of this group have the highest confidential clearance."}')
```

7.4.9 機密グループを削除する

以下のように、機密グループを削除します。

```
select pgx_drop_confidential_group('matrix_foo', 'group1', true, true)
```

第2引数や第3引数を指定すると、指定された機密グループに依存するものを削除したり、機密グループロールを削除したりすることを選択できます。どのような選択をしても機密グループに属するロールは削除されません。

注意

機密グループロールを残しても、この関数は、機密グループロールから権限を剥奪します。剥奪する権限は、機密権限に定義された権限です。ですから、機密グループに登録されたロールは、機密グループロールを継承することで付与されていた様々な権限を剥奪されることに注意してください。

7.4.10 機密権限を剥奪する

以下のように、機密グループから機密権限を剥奪します。JSON形式で機密オブジェクトと機密グループと剥奪する権限を指定してください。以下の例では、'group1'から'level1'に属するtable型の機密オブジェクトに対するINSERT権とDELETE権を剥奪しています。

```
select pgx_revoke_confidential_privilege('matrix_foo','level1', 'group1',  '{"table":["INSERT","DELETE"]}')
```

7.4.11 機密レベルから機密オブジェクトを除外する

以下のように、機密レベルから機密オブジェクトを除外します。JSON形式で除外する機密オブジェクトの型と名前を指定してください。以下の例では、機密レベル'level1'から、テーブル型の機密オブジェクトであるschema1.table1とschema1.table2を除去しています。

```
select pgx_remove_object_from_confidential_level('matrix_foo', 'level1',
'[{
    "type":"table",
    "object":[
        {
          "schema":"schema1",
          "table":["table1","table2"]
        }
    ]
}]')
```

7.4.12 機密グループからロールを除去する

以下のように機密グループからロールを除去します。JSON形式で除去するロールを指定してください。以下の例では、機密グループ'group1'から、ロール'role1'と'role2'を除去しています。

```
select pgx_remove_role_from_confidential_group('matrix_foo', 'group1', '["role1","role2"]')
```

7.5 監視方法の提案

機密管理者は、機密管理の定義を実行した後も、意図したとおりに安全にデータベースが運用されていることを保証しなければなりません。機密管理支援機能だけを使用しているならば、そのような心配をする必要はありません。しかし、機密管理支援機能は、この機能を使わずにテーブルやロールの定義を変更することを***禁止しません***。

ですから、そのような行為が行われたことを検知しなければなりません。

しかし、検知したとしても対処を忘れるかもしれません。ですから、定期的に、機密レベルや機密グループと、実際の機密オブジェクトやロールの定義が一致していることの差分を確認しなければなりません。もちろん、仮に不一致だったとしても、機密オブジェクトやロールの方が、より厳しい属性や権限を設定されていたのならば問題ないでしょう。

ここで示す手順は、一致することを目標とします。

7.5.1 機密管理支援機能を使用しない権限の変更を検知するための方法

機密管理支援機能を介さずに、不正に、機密オブジェクトやロールの属性を変更したり、権限を変更したりしたことを、監査ログを使って検出してください。

検出するための基本的な方法は、機密管理ロール以外のロールによる操作を検出することです。しかし、以下のような例外があります。

正当な理由のために、機密管理支援機能を介さない操作を行う場合

例えば、機密管理支援機能が機密オブジェクトとして扱わない関数の権限を変更する場合です。これを見分けるためには、機密管理支援機能を介さないような変更を行うロールを決めておくことを推奨します。多様なロールがこれを行うと、運用ルールを逸脱した監査ログを検出することが難しくなるからです。

機密管理ロールの活動を監視したい場合

例えば、決められた時間や、決められた端末からだけアクセスするようなルールを作り、かつ、そのルールに違反した活動を監査ログから検出すると良いでしょう。違反したことを隠蔽できないようなルールを定めることが重要です。例えば、application_nameは簡単に偽装することができるので適切ではありません。

7.5.2 機密オブジェクトやロールを確認する方法

機密マトリクスに登録された内容と、PostgreSQLのシステムカタログで管理される機密オブジェクトやロールの状態とが一致していることを確認する方法を示します。ここで示す方法は、ひとつの例にすぎません。

確認者

確認者は、pg_confidential_management_support拡張が持つ表へのSELECT権限と、システム表における、対象の機密マトリクスに追加された機密オブジェクトとロールの情報へのSELECT権限を付与されていなければなりません。少なくとも、確認対象の機密マトリクスの機密管理ロールとスーパーユーザーは、確実にそのような権限を持っています。

手順

確認するための手順は、以下の3つのフェーズに分かれます。不一致を検出したならば、監査ログを利用して、不一致を発生させた原因を調査してください。そして、一致するように修正してください。

- 正しい状態に戻すと、様々なオブジェクトが連鎖的に変更(修正)される場合があります。そうすると、不正な操作の痕跡が消されてしまうかもしれません。ですから、設計と一致していない箇所をすべて調査し、調査に必要な監査ログを確保してから、修正することをお勧めします。

・ ここで紹介するpgx_get_privileges_on_matrix()関数は、機密オブジェクトやロールの数が大きいと、非常に大きな表を出力する可能性があります。この表のサイズがPostgreSQLのwork_memパラメータの値をこえると、PostgreSQLの仕様にしたがってI/Oが発生し、遅くなります。ですから、この関数を実行するセッションの中で、できる限り大きな値をwork_memに指定することをお勧めします。

(1) 機密マトリクスの状態が、設計と同じであることを確認する

不一致なものを検出した場合には、この拡張が提供するpgx_alter_confidential_level()などの関数を実行して正しい状態に戻してください。

― 機密マトリクスの属性

pgx_confidential_matrixテーブルを使って確認してください。

― 登録されている機密レベルの数と各機密レベルの属性

pgx_confidential_levelテーブルの中から、確認対象のclevmatidと一致する行を参照して確認してください。機密マトリクスの識別子を知るためには、pgx_confidential_matrixテーブルのcmatidを参照してください。

― 登録されている機密グループの数と各機密グループの属性

pgx_confidential_roleテーブルの中から、ccolmatidが確認対象の機密マトリクスの識別子と一致する行を参照して確認してください。

― 機密レベルに設定されている機密グループの権限

pgx_confidential_privilegesテーブルを参照してください。ただし、このテーブルでは、機密レベルや機密グループは識別子として表現されています。機密レベルの名前や機密グループの名前で確認したいならば、pgx_confidential_levelテーブルやpgx_confidential_roleテーブルと結合してください。

(2) 登録されている機密オブジェクトとロールが正しいかを確認する

機密管理支援機能の定義と、機密オブジェクトやロールの定義とが一致していることを確認します。不一致なものを検出した場合には、この拡張が提供するpgx_alter_confidential_level()などの関数を実行して正しい状態に戻してください。

― 機密レベルに登録されている機密オブジェクトの数が設計と同じか

― 各機密オブジェクトの属性が、その機密オブジェクトが属する機密レベルの属性と一致しているか

pgx_get_attribute_of_objects()を使って、これらを確認することをお勧めします。なぜならば、この関数は、機密オブジェクトの状態をPostgreSQLのシステムカタログから得て、定義と比較できるような表を出力するからです。表の形式は、“B.7 定義の参照およびシステムカタログとの比較を支援する関数”を参照してください。

― 機密グループに登録されているロールの数が設計と同じか。

― 各ロールの属性が、そのロールが属する機密グループの属性と一致しているか。

pgx_get_attribute_of_roles()を使って、これらを確認することをお勧めします。この関数が返却する表の形式は、“B.7 定義の参照およびシステムカタログとの比較を支援する関数”を参照してください。

(3) 機密オブジェクトへのアクセス権限を確認する

機密オブジェクトへのアクセス権限を付与されているロールと、権限の内容が機密マトリクスの定義と一致していることを確認します。pgx_get_privileges_on_matrix()関数を使用すると便利です。出力フォーマットの詳細は、“7.3.5 機密権限を機密グループに付与する”を参照してください。

もし、いずれかの機密レベルやロールに着目して調べたい場合は、以下の関数を使用してください。

― pgx_get_privileges_on_level_and_group()

― pgx_get_privileges_on_object()

― pgx_get_privileges_on_role()

ポイント

様々なSQL文で分析するために、大きな表を何度も出力させることは非効率です。以下のように、この関数を実行するクエリをINSERT文に指定すると効率的に分析を行うことができます。

```
INSERT INTO temp_table_for_analysis SELECT pgx_get_privileges_on_matrix('matrix_foo')
```

7.6 バックアップ/リストア

以下の4つの方法があります。

- pg_basebackupを使用する
- pg_dumpallを使用して、データベースクラスタ全体をスクリプトファイルとしてバックアップする
- pg_dumpとpg_restoreを使用して、一部のデータベースだけをバックアップおよびリストアする
- pg_dumpとpg_restoreを使用して、一部のテーブルやスキーマだけをバックアップおよびリストアする

このうち、pg_basebackupを使用する方法を採用するならば、特別に注意するべきことはありません。他の方法を採用するときには、以下に示す特別な注意が必要です。

pg_dumpallを使用して、データベースクラスタ全体をスクリプトファイルとしてバックアップする

以下に注意してください。

- pg_dumpallコマンドの-Oオプションを使用しないでください。機密オブジェクトの所有者が変わると、この機能が正しく動作しないからです。
- pg_dumpallコマンドの-xオプションを使用しないでください。当然ですが、厳密に管理されていた権限が変更されてしまうからです。
- スーパーユーザーが、バックアップされたスクリプトファイルを実行してください。バックアップされたスクリプトファイルには、この拡張のCREATE EXTENSION文が含まれています。このCREATE EXTENSION文を、スーパーユーザーで実行しなければならないからです。

pg_dumpとpg_restoreを使用して、一部のデータベースだけをバックアップおよびリストアする

以下に注意してください。

- pg_dumpallコマンドの-rオプションを使って、ロールの情報もバックアップおよびリストアしてください。当然ですが、機密管理にはロールが不可欠だからです。
- pg_dumpコマンドの-Oオプションを使用しないでください。機密オブジェクトの所有者が変わると、この機能は正しく動作しないからです。
- pg_dumpコマンドの-xオプションを使用しないでください。当然ですが、厳密に管理されていた権限が変更されてしまうからです。
- スーパーユーザーが、リストアしてください。この拡張を作成しなければならないからです。

pg_dumpのpg_restoreを使用して、一部のテーブルやスキーマだけをバックアップおよびリストアする

以下に注意してください。

- pg_dumpコマンドの-eオプションを使用して、この拡張もバックアップおよびリストアしてください。この拡張に含まれるテーブルに、機密管理に必要な情報が格納されているからです。
- pg_dumpallコマンドの-rオプションを使って、ロールの情報もバックアップおよびリストアしてください。当然ですが、機密管理にはロールが不可欠だからです。
- pg_dumpコマンドの-Oオプションを使用しないでください。機密オブジェクトの所有者が変わると、この機能は正しく動作しないからです。
- pg_dumpコマンドの-xオプションを使用しないでください。当然ですが、厳密に管理されていた権限が変更されてしまうからです。
- スーパーユーザーが、リストアしてください。この拡張を作成しなければならないからです。

7.7 アンセットアップ

この機能を利用しなくなった後も、それまでと同じようにデータベースを使い続ける場合には、単にこの拡張を削除（DROP EXTENSION）することをお勧めします。なぜならば、もしこの拡張を使って作成した機密マトリクスや機密レベルなどの定義を、この拡張が提供する削除のための関数を使って削除すると、この拡張が自動的に作成したロールが削除されたり、この拡張が付与したロールの権限が剥奪されたりするからです。単にこの拡張を削除するだけならば、この拡張が含むテーブルが削除されるだけであり、これらのことは行われません。

7.8 機密管理支援機能の利用例

機密レベルと機密グループの考え方の例を説明します。

ここでは、顧客の購入情報を取り扱う簡単な業務を想定します。

まずは、これらの情報の機密管理を行うための機密マトリクスを作成しておきます。

```
SELECT pgx_create_confidential_matrix('購入情報管理', '顧客の購入情報の機密管理');
```

取り扱うデータの中には個人が特定できる情報も含まれていることがあります。そのようなデータに関しては、アクセスできる人物を限定し、情報が漏洩するリスクを最小限に抑える必要があります。

個人が特定できる情報を含む顧客の購入情報は次のようなテーブルで管理しています。

```
CREATE TABLE 購入管理.顧客情報(      -- 顧客の情報
    顧客id         integer,          -- 顧客のID
    氏名           text,
    住所           text,
    電話番号       char(12),
    顧客ランク     integer           -- 顧客のサービスランク
);

CREATE TABLE 購入管理.購入履歴(      -- 顧客が商品を購入した履歴
    顧客id        integer,           -- 購入した顧客のID
    購入日        date,              -- 商品を購入した日付
    商品コード    char(12),          -- 購入した商品のコード
    数量          integer,           -- 購入した数量
    金額          integer            -- 購入した商品の金額
);
```

顧客情報の中で、氏名、住所、電話番号はそれらを合わせると個人が特定できてしまうため、個人情報となります。このような情報を適切に取り扱うために、適切な教育を受けた特定のグループに属した従業員だけが取り扱えるようにしました。

そこで、機密度の高い個人情報を意味する「個人情報」と、それ以外の情報を意味する「顧客情報」の2つの機密レベルを用意します。

```
SELECT pgx_create_confidential_level('購入情報管理', '個人情報',
                                     NULL, '個人を特定できる情報');
SELECT pgx_create_confidential_level('購入情報管理', '顧客情報',
                                     NULL, '個人を特定できない情報');
```

さらに、個人情報の取り扱いに関して教育を受け、適切に個人情報を取り扱いうことができる「有資格者」と、それ以外の「一般従業員」の2つの機密グループを用意します。

```
SELECT pgx_create_confidential_group('購入情報管理', '有資格者',
                                     NULL, '個人情報取り扱い有資格者');
SELECT pgx_create_confidential_group ('購入情報管理', '一般従業員',
                                     NULL, '一般従業員');
```

取り扱うデータに関して、もう少し詳しく見ていきます。

顧客情報テーブルは個人情報を含むので個人情報に該当します。しかし、顧客情報テーブルに含まれる顧客idと顧客ランクは個人を特定できる情報ではないため、個人情報ではありません。また、この顧客idと顧客ランクは経営分析に必要となる情報でもあるため、それらの情報を個人情報の取り扱い有資格者しかアクセスできないのは不便です。

そこで、顧客情報テーブルはカラムを用いた機密管理を行うことにします。顧客情報テーブル全体は個人情報として保護を行い、個人情報とはならないカラムを一般の顧客情報とすることでアクセスできる範囲を広げます。

この方針に従い、機密グループに対する機密レベルの権限を設定します。

```
SELECT pgx_grant_confidential_privilege('購入情報管理', '個人情報', '有資格者',
                                     '{"table":["ALL"]}');
```

```
SELECT pgx_grant_confidential_privilege('購入情報管理', '顧客情報', '有資格者',
                                        '{"table":["ALL"]}');
SELECT pgx_grant_confidential_privilege('購入情報管理', '顧客情報', '一般従業員',
                                        '{"table":["ALL"], "column":["SELECT"]}');
```

「個人情報」を取り扱えるのは「有資格者」のみです。「顧客情報」は「有資格者」も「一般従業員」も扱うことができます。「個人情報」を扱うテーブルの一部のカラムは「顧客情報」として参照することを許可します。

これで機密マトリクスでの権限設定が完了しました。

続いて購入情報を取り扱うデータベースオブジェクトを機密マトリクスに登録していきます。

```
SELECT pgx_add_object_to_confidential_level('購入情報管理', '個人情報',
                                    '[{
                                        "type":"table",
                                        "object":[{
                                                    "schema": "購入管理",
                                                    "table": ["顧客情報"]
                                         }]
                                    }]');
SELECT pgx_add_object_to_confidential_level('購入情報管理', '顧客情報',
                                    '[{
                                        "type":"column",
                                        "object":[{
                                                     "schema": "購入管理",
                                                     "table": "顧客情報",
                                                     "column": ["顧客id", "顧客ランク"]
                                         }]
                                    }]');
SELECT pgx_add_object_to_confidential_level('購入情報管理', '顧客情報',
                                    '[{
                                        "type":"table",
                                        "object":[{
                                                     "schema": "購入管理",
                                                     "table": ["購入履歴"]
                                    }]
                                    }]');
```

顧客情報テーブル全体は「個人情報」、顧客情報テーブルの顧客idカラムと顧客ランクカラムは「顧客情報」、購入履歴テーブル全体も「顧客情報」となります。

最後に、従業員を機密グループに登録します。「相田」さんと「飯田」さんは個人情報管理の教育を受けた「有資格者」、「上田」さんと「江田」さんはまだ個人情報管理の教育を受けてはいないため「一般従業員」です。

```
SELECT pgx_add_role_to_confidential_group('購入情報管理', '有資格者',
                                          '["相田", "飯田"]');
SELECT pgx_add_role_to_confidential_group('購入情報管理', '一般従業員',
                                          '["上田", "江田"]');
```

付録A　機密管理支援機能が利用するテーブル

機密管理支援機能が利用するテーブルを説明します。

A.1　pgx_confidential_matrix

機密マトリクスの一覧です。登録された機密マトリクスの属性、更新時刻、機密レベルや機密グループを登録または削除した時刻などを参照できます。

カラム名	型	制約	説明
cmatid	bigint	primary key generated always as identity	機密マトリクスの識別子。
cmatname	varchar(63)	unique not null	機密マトリクスの名前。
cmatowner	name	not null	機密マトリクスの所有者。機密マトリクスを作成したロールが所有者となる。
cmatcomment	text		コメント。
cmatupdatetime	timestamp with time zone	not null	機密マトリクス自体の更新時刻。
cmatoperationtime	timestamp with time zone		機密レベル、機密グループを追加・削除した時刻。

A.2　pgx_confidential_level

機密レベルの一覧です。登録された機密レベルの属性、更新時刻、機密レベルに機密オブジェクトを登録または削除した時刻などを参照できます。

カラム名	型	制約	説明
clevid	bigint	primary key generated always as identity	機密レベルの識別子。
clevname	varchar(63)	not null	機密レベルの名前。
clevmatid	bigint	not null references pgx_confidential_matrix(cmatid)	機密レベルが属する機密マトリクスの識別子。
clevcomment	text		コメント。
clevupdatetime	timestamp with time zone	not null	機密レベル自体の更新時刻。
clevoperationtime	timestamp with time zone		機密オブジェクトを追加、削除した時刻。
clevencalgorithm	text	not null	暗号化手法。暗号化を利用しないときにはnone。AES128、AES256が設定可能。

A.3　pgx_confidential_group

機密グループの一覧です。登録された機密グループの属性、更新時刻、機密グループにロールを登録または削除した時刻などを参照できます。

カラム名	型	制約	説明
cgroid	bigint	primary key generated always as identity	機密グループの識別子。
cgroname	varchar(63)	not null	機密グループの名前。
cgromatid	bigint	not null references pgx_confidential_matrix(cmatid)	機密グループが属する機密マトリクスの識別子。
cgrocomment	text		コメント。
cgroupdatetime	timestamp with time zone	not null	機密グループ自体の更新時刻。
cgrooperationtime	timestamp with time zone		ロールを追加、削除した時刻。
cgrorolename	name	not null	機密グループロールの名前。
cgrosuperuser	bool	not null	機密グループのロールがSUPERUSER権限を持つ場合にtrue。
cgrocreatedb	bool	not null	機密グループのロールがCREATEDB権限を持つ場合にtrue。
cgrocreaterole	bool	not null	機密グループのロールがCREATEROLE権限を持つ場合にtrue。
cgroreplication	bool	not null	機密グループのロールがREPLICATION権限を持つ場合にtrue。
cgrobypassrls	bool	not null	機密グループのロールがBYPASSRLS権限を持つ場合にtrue。

A.4 pgx_confidential_privilege

機密権限の一覧です。機密オブジェクトごとに設定された機密権限や更新時刻などを参照できます。

カラム名	型	制約	説明
cpriid	bigint	primary key generated always as identity	権限の識別子。
cprimatid	bigint	not null references pgx_confidential_matrix(cmatid)	権限が属する機密マトリクスの識別子。
cprilevid	bigint	not null references pgx_confidential_level(clevid)	権限の設定対象となる機密レベルの識別子。
cprigroid	bigint	not null references pgx_confidential_group(cgroid)	権限の設定対象となる機密グループの識別子。
cpritype	text	not null	権限の設定対象となる機密オブジェクトの種別。
cpriupdatetime	timestamp with time zone	not null	権限の設定・変更が行われた時刻。
cpriacl	text[]	not null	設定されているアクセス権限(注1)。

注1: 権限を示す文字列は、cpriaclのtext型の配列の中で、以下の順に現れます。

ALL,SELECT,INSERT,UPDATE,DELETE,TRUNCATE,REFERENCES,TRIGGER,CREATE,CONNECT,TEMPORARY,EXECUTE,USAGE

もし、該当する権限が存在しなければ、単に文字列が現れません。例えば、INSERT権限とTRUNCATE権限だけを持っているならば、{'INSERT','TRUNCATE'}です。

A.5 pgx_confidential_object

機密オブジェクトの一覧です。オブジェクトの属性や更新時刻などを参照できます。

カラム名	型	制約	説明
cobjid	bigint	primary key generated always as identity	機密オブジェクトの識別子。
cobjmatid	bigint	not null references pgx_confidential_matrix(cmatid)	機密オブジェクトが属する機密マトリクスの識別子。
cobjlevid	bigint	not null references pgx_confidential_level(clevid)	機密オブジェクトが属する機密レベルの識別子。
cobjtype	text	not null	機密オブジェクトの種別。
cobjschema	name	not null	機密オブジェクトのスキーマ名。
cobjtable	name	not null	機密オブジェクトのテーブル名。
cobjname	text	not null	機密オブジェクトの名前。
cobjupdate	timestamp with time zone	not null	機密オブジェクトの登録時刻。
cobjpolicy	jsonb		種別がrowsetの場合に、行集合の範囲を決める条件。 POLICYでの設定内容で表現する。

A.6 pgx_confidential_role

機密グループに登録されたロールの一覧です。ロールの属性や更新時刻などを参照できます。

カラム名	型	制約	説明
crolid	bigint	primary key generated always as identity	ロールの識別子。
crolmatid	bigint	not null references pgx_confidential_matrix(cmatid)	ロールが属する機密マトリクスの識別子。
crolgroid	bigint	not null references pgx_confidential_group(cgroid)	ロールが属する機密グループの識別子。
crolname	name	not null	ロールの名前。
crolupdate	timestamp with time zone	not null	ロールの登録時刻。

A.7 pgx_confidential_policy

rowset型の機密オブジェクトに権限を設定するために作成したポリシーの一覧です。作成したポリシーの名前、設定している権限を参照できます。このテーブルの行は、rowset型の機密オブジェクトを追加したときに挿入されます。

カラム名	型	制約	説明
cpolid	bigint	primary key generated always as identity	ポリシーの識別子。
cpolmatid	bigint	not null references pgx_confidential_matrix(cmatid)	ポリシーが属する機密マトリクスの識別子。
cpollevid	bigint	not null references pgx_confidential_level(clevid)	ポリシーが属する機密レベルの識別子。
cpolgroid	bigint	not null references pgx_confidential_group(cgroid)	ポリシーが属する機密グループの識別子。
cpolobjid	bigint	not null references pgx_confidential_object(cobjid)	このポリシーを利用している行セットオブジェクトの識別子。
cpolprivilege	text	not null	このポリシーが持つ権限(SELECT, INSERT, UPDATE, DELETE, ALL)。
cpolname	name	not null	ポリシーの名前。
cpolexpression	jsonb	not null	ポリシーの条件。

付録B　機密管理支援機能が利用するシステム管理関数

機密管理支援機能が利用するシステム管理関数を説明します。すべての関数は、失敗した場合にトランザクションをアボートします。

注意

- 機密グループの削除を伴うような操作をするときには、注意が必要です。

  機密グループを機密グループロールとともに削除するならば、単に機密オブジェクトにアクセスできるロールがなくなるだけです。しかし、機密グループロールを残すときには、機密グループロールから権限を剥奪します。剥奪する権限は、機密権限に定義された権限です。

- pgx_get_privileges_on_matrix()関数は、機密オブジェクトやロールの数が大きいと、非常に大きな表を出力する可能性があります。この表のサイズがPostgreSQLのwork_memパラメータの値を超えると、PostgreSQLの仕様にしたがってI/Oが発生し、遅くなります。これを抑止するために、この関数を実行するセッションの中で、できる限り大きな値をwork_memに指定することをお勧めします。

B.1　マトリクス操作関数

関数名	戻り値	説明
pgx_create_confidential_matrix(confidential_matrix_name varchar, comment text)	void	指定された名前の機密マトリクスを作成します。 作成された機密マトリクスは、commentとともにpgx_confidential_matrixテーブルに登録されます。 機密管理ロールに要求される資格を持つロールだけが、この関数を実行できます。この関数を実行したロールが、機密マトリクスの機密管理ロールです。機密マトリクス名を指定して実行するような関数は、実行したロールが、指定された機密マトリクスの機密管理ロールであることを要求します。詳細は、“7.2.2 機密管理ロールを決める”を参照してください。 confidential_matrix_nameの長さは、64文字未満でなければなりません。単位がバイトではないことに注意してください。 confidential_matrix_nameに使える文字に制限はありません。 他の関数に機密マトリクスの名前を指定するときには、この関数に指定した文字列と同じ文字列を指定しなければなりません。 多くのCREATE文とは違って、機密マトリクスの名前は大文字と小文字を区別することに注意してください。
pgx_copy_confidential_matrix(confidential_matrix_name varchar, source_confidential_matrix_name varchar)	void	source_confidential_matrix_nameで指定された複製元の機密マトリクスを、confidential_matrix_nameと名付けられた機密マトリクスに複製します。機密マトリクス、機密レベル、機密グループ、および、機密権限の詳細が複製されます。しかし、機密レベルに登録されている機密オブジェクト、機密グループに登録されているロールの情報は複製されません。 いずれかの機密マトリクスの機密管理ロールであれば、この関数を実行できます。 複製された機密マトリクスの所有者は、この関数を実行したロールです。 confidential_matrix_nameの文字列としての制限は、pgx_create_confidential_matrix関数と同じです。 コメントも複製されます。
pgx_alter_confidential_matrix(confidential_matrix_name varchar, alter_object json)	void	confidential_matrix_nameで指定された名前の機密マトリクスの属性を変更します。

関数名	戻り値	説明
		指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。 alter_objectには以下のように、変更したい属性と変更後の値をkey-value形式で指定します。指定しなかった属性は変更されません。 '{ "name":"matrix_foo", "comment":"This matrix is defined for foo.") }' name: 変更後の機密マトリクスの名前を指定してください。nullを指定できません。すでに存在する機密マトリクスの名前を指定すると関数は失敗します。 comment: 変更後のコメントを指定してください。nullを指定できます。
pgx_drop_confidential_matrix(confidential_matrix_name varchar, cascade bool, drop_role bool)	void	指定された名前の機密マトリクスを削除します。 指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。 cascadeにtrueを指定すると、この機密マトリクスに依存したオブジェクトを再帰的チェックして削除します。例えば、この機密マトリクスに登録されている機密グループと機密レベルを削除します。そして、その機密レベルに関連する機密権限を削除します。機密レベルを削除するときは、cascadeの値を指定してpgx_drop_confidential_level()を実行します。機密グループを削除するときは、cascadeとdrop_roleの値を指定してpgx_drop_confidential_group()を実行します。これらの関数の説明も参照してください。特に、機密オブジェクトの権限がどのように変更されるのかは重要です。 cascadeにfalseを指定すると、単に機密マトリクスだけを削除します。もし、この機密マトリクスに依存するオブジェクトが存在するならば、関数は失敗します。 drop_roleにtrueを指定すると、この機密マトリクスに登録された機密グループロールを削除します。当然ですが、cascadeがtrueのときにだけ意味があります。

B.2 機密レベル操作関数

関数名	戻り値	説明
pgx_create_confidential_level(confidential_matrix_name varchar, confidential_level_name varchar, options json, comment text)	void	機密レベルを作成し、指定された機密マトリクスに登録し、指定されたcommentやoptionsに指定された属性とともにpgx_confidential_levelテーブルに追加します。 指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。 confidential_level_nameの長さは、64文字未満でなければなりません。 単位がバイトではないことに注意してください。 confidential_level_name名に使える文字に制限はありません。 他の関数に機密レベルの名前を指定するときには、この関数に指定した文字列と同じ文字列を指定しなければなりません

関数名	戻り値	説明
		多くのCREATE文とは違って、機密レベルの名前は大文字と小文字を区別することに注意してください。 commentには、コメントを指定してください。 optionsには以下のように、機密レベルの属性を指定します。もしNULLを指定したならば、各属性のデフォルト値が設定されます。 '{ "encryption_algorithm":"AES256" }' encryption_algorithm: 暗号化アルゴリズムを指定してください。指定できるアルゴリズムとデフォルト値は、Fujitsu Enterprise Postgresの透過的データ暗号化のtablespace_encryption_algorithmパラメータと同じです。nullを指定してはなりません。
pgx_alter_confidential_level(confidential_matrix_name varchar, confidential_level_name varchar, alter_object json)	void	機密レベルの属性を変更します。 指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。 alter_objectには以下のように、変更したい属性と変更後の値をkey-value形式で指定します。 '{ "name":"level_new", "comment":"This level is the highest confidential level.", "encryption_algorithm":"AES256" }' name: 変更後の機密レベルの名前を指定します。nullを指定できません。 comment: 変更後のコメントを指定してください。nullを指定できます。 その他の属性は、pgx_create_confidential_level()のoptionsと同じです。指定されなかった属性は変更されません。 機密性の度合いを上げるときには注意が必要です。例えば、暗号化の強度を強くするときに、変更後の強度よりも低い機密オブジェクトがあったならば、この関数は失敗します。
pgx_drop_confidential_level(confidential_matrix_name varchar, confidential_level_name varchar, cascade bool)	void	機密マトリクスから機密レベルを除去し、機密レベルを削除します。 指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。 cascadeにtrueを指定すると、この機密レベルに機密オブジェクトが登録されていても機密レベルを削除することができます。 cascadeにfalseを指定すると、機密オブジェクトが登録されているような機密レベルを削除することができません。

B.3 機密グループ操作関数

関数名	戻り値	説明
pgx_create_confidential_group(confidential_matrix_name varchar, confidential_group_name varchar, options json, comment text)	void	機密グループを作成し、機密マトリクスに登録し、指定されたcommentやoptionsに指定された属性とともにpgx_confidential_groupテーブルに追加します。

関数名	戻り値	説明
		指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。ただし、後に述べるように、一部の属性を設定するときにはスーパーユーザー権限が必要です。したがって、必然的に、それらの属性を設定するならば、機密管理ロールはスーパーユーザーでなければなりません。 この関数は、内部的にCREATE ROLE文を使って機密グループロールを作成します。 confidential_group_nameの長さは、64文字未満でなければなりません。単位がバイトではないことに注意してください。 confidential_group_name名に使える文字に制限はありません。 他の関数に機密グループの名前を指定するときには、この関数に指定した文字列と同じ文字列を指定しなければなりません。 多くのCREATE文とは違って、機密グループの名前は大文字と小文字を区別することに注意してください。 optionsには以下のように、機密グループの属性を指定します。もしNULLを指定したならば、各属性のデフォルト値が設定されます。 '{ "SUPERUSER":false, "CREATEDB":true, "CREATEROLE":false, "REPLICATION":false, "BYPASSRLS":false }' 指定できる属性は、データへのアクセス権限に関係している、SUPERUSER属性、CREATEDB属性、CREATEROLE属性、REPLICATION属性およびBYPASSRLS属性だけです。 これらの属性は、CREATE ROLE文に指定できるロールの属性の一部です。属性の意味とデフォルト値は、CREATE ROLE文の仕様と同じです。 CREATE ROLE文の説明で述べられているように、スーパーユーザー以外が、SUPERUSER属性、REPLICATION属性、および、BYPASSRLS属性にtrue指定すると、この関数は失敗します。
pgx_alter_confidential_group(confidential_matrix_name varchar, confidential_group_name varchar, alter_object json);	void	機密グループの属性を変更します。 指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。pgx_create_confidential_group()で説明したように、一部の属性を設定するときにはスーパーユーザーでなければなりません。 この関数は、内部的にALTER ROLE文を使って機密グループロールの属性を変更します。 alter_objectには以下のように、変更したい属性と変更後の値をkey-value形式で指定します。 '{ "name":"group_new", "comment":"Members of this group have the highest confidential clearance.", "CREATEDB":false }' name: 変更後の機密グループの名前を指定します。nullを指定できません。

関数名	戻り値	説明
		comment: 変更後のコメントを指定してください。nullを指定できます。 その他の属性は、pgx_create_confidential_group()のoptionsと同じです。指定されなかった属性は変更されません。 例えば、CREATEDBをfalseへと変更するように、より弱い属性へと変更した場合には、機密グループに登録されたロールの属性も同じように弱めます。
pgx_drop_confidential_group(confidential_matrix_name varchar, confidential_group_name varchar, cascade bool, drop_role bool)	void	機密マトリクスから機密グループを除去し、機密グループを削除します。 指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。 cascadeにtrueを指定すると、この機密グループにロールが登録されていても機密グループを削除することができます。 cascadeにfalseを指定すると、ロールが登録されているような機密グループを削除できません。 drop_roleにtrueを指定すると、機密グループロールを削除します。機密グループに登録されたロールは、そのまま残ります。drop_roleにfalseを指定すると、機密グループロールを削除しません。 機密グループロールを残すときには、機密グループロールから権限を剥奪します。剥奪する権限は、機密権限に定義された権限です。

B.4 機密権限操作関数

関数名	戻り値	説明
pgx_grant_confidential_privilege(confidential_matrix_name varchar, confidential_level_name varchar, confidential_group_name varchar, privilege json)	void	機密権限を付与します。 指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。 confidential_level_nameで指定された機密レベルへのアクセス権限を、confidential_group_nameで指定された機密グループに付与します。 繰り返してこの関数を実行したときには、単に付与する権限を追加します。同じ権限を重複して付与してもエラーにはなりません。 付与する権限は、privilegeで指定します。privilegeは、以下のように、キーに機密オブジェクトの型を指定し、値に権限の配列を指定します。 '{ "table":["SELECT", "INSERT", "UPDATE", "DELETE"], "schema":["CREATE", "USAGE"], "rowset":["ALL"] }' 指定できる権限は、機密オブジェクトの型に依存して、異なります。 rowset型は、CREATE POLICY文のFOR句に指定できる権限です。 rowset型以外では、機密オブジェクトの型に応じた、GRANT文で付与できる権限です。

関数名	戻り値	説明
		ALLを指定した場合には、その型で指定できるすべての権限を列挙したとみなします。 つまり、pgx_confidential_privilegeテーブルのcpriaclカラムには、ALLが現れません。 GRANT文のWITH GRANT OPTION句と同じような指定はできません。なぜならば、機密管理ロールだけが、この機能を使って、機密オブジェクトへのアクセス権限を変更するべきだからです。 PUBLICへ権限を付与した機密オブジェクトをターゲットにするときには、注意が必要です。なぜならば、PUBLICへ権限を付与することは、機密マトリクスに登録されたすべてのロールへ権限を与えていることと同じだからです。もし、各ロールにPUBLICを利用して間接的に付与された権限が、そのロールに与えられるべきではないことが機密権限で定義されているならば、この関数は失敗します。同様に、機密マトリクスに登録されていないグループロールを利用して間接的に付与された権限も、この関数は検査します。このとき、継承のチェインを再帰的に検査します。
pgx_revoke_confidential_privilege(confidential_matrix_name varchar, confidential_level_name varchar, confidential_group_name varchar, privilege json)	void	機密権限を剥奪します。 指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。 confidential_level_nameで指定された機密レベルへのアクセス権限を、confidential_group_nameで指定された機密グループから剥奪します。 付与されていない権限を剥奪しても失敗しません。 剥奪する権限は、privilegeで指定します。指定方法は、pgx_grant_confidential_privilege()と同じです。 PUBLICへ権限を付与した機密オブジェクトをターゲットにするときには、注意が必要です。なぜならば、PUBLICへ権限を付与することは、機密マトリクスに登録されたすべてのロールへ権限を与えていることと同じだからです。もし、各ロールにPUBLICを利用して間接的に付与された権限が、そのロールに与えられるべきではないことが機密権限で定義されているならば、この関数は失敗します。同様に、機密マトリクスに登録されていないグループロールを利用して間接的に付与された権限も、この関数は検査します。このとき、継承のチェインを先祖まで検査します。

B.5 機密オブジェクト操作関数

関数名	戻り値	説明
pgx_add_object_to_confidential_level(confidential_matrix_name varchar, confidential_level_name varchar, object_name json)	void	指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。 confidential_level_nameで指定された機密レベルに、object_nameで指定された機密オブジェクトを追加します。 この関数は、内部的にGRANT文を使って、この機密レベルに関連付けされた機密権限にしたがって、機密グループロールに権限を付与します。 ただし、機密オブジェクトの型がrowsetの場合は、内部的にCREATE POLICY文を使って、この機密レベルに関連付けされた機密権限にしたがって、機密グループロールに権限を付与します。また、POLICYを有効化するために、ENABLE ROW LEVEL

関数名	戻り値	説明

		SECURITY 句を指定して、ターゲットのテーブルに対してALTER TABLE文を実行します。
		
		今は、table型の機密オブジェクトとして外部テーブルを登録することはできません。
		
		PUBLICへ権限を付与した機密オブジェクトをターゲットにするときには、注意が必要です。なぜならば、PUBLICへ権限を付与することは、機密マトリクスに登録されたすべてのロールへ権限を与えていることと同じだからです。もし、各ロールにPUBLICを利用して間接的に付与された権限が、そのロールに与えられるべきではないことが機密権限で定義されているならば、この関数は失敗します。同様に、機密マトリクスに登録されていないグループロールを利用して間接的に付与された権限も、この関数は検査します。このとき、継承のチェインを先祖まで検査します。
		
		rowset型の機密オブジェクトの場合には、ターゲットのテーブルに、この機能を使わずに作成されたPOLICYが定義されていたならば、付与された権限の内容にかかわらず、この関数は失敗します。
		
		このような検査は、機密グループロールや機密グループに登録されたロールだけを対象にします。そうではないロールをターゲットにしたPOLICYが存在していても、この関数は失敗しません。
		
		object_nameは、以下のように指定します。rowsetだけが少し異なります。以下の例では、複数の種類のオブジェクトを1回で登録しようとしています。
		
		'[{
		"type":"schema",
		"object":[
		{"schema":"schema1"},
		{"schema":"schema2"}
		]
		},
		
		{
		"type":"table",
		"object":[
		{
		"schema":"schema1",
		"table":["table1","table2"]
		},
		{
		"schema":"schema2",
		"table":["table8","table9"]
		}
		]
		},
		
		{
		"type":"column",
		"object":[
		{
		"schema":"schema1",
		"table":"table1",
		"column":["column1", "column2"]
		},
		{
		"schema":"schema1",
		"table":"table2",
		"column":["column8", "column9"]

関数名	戻り値	説明

		}
		]
		}]'
		
		rowset型の場合には、次に示す例のように、どのような行の集合であるかを定義し、名前を付けます。この名前は、pgx_remove_object_from_confidential_level関数で、rowset型の機密オブジェクトを除去するときに、その機密オブジェクトを識別するために使用します。
		
		この例では、以下のことを示しています。
		
		・ schema1.table1において、条件式(user = current_user OR manger = current_user)がtrueであるような行の集合(rowset)を表している。
		
		・ manager=current_userがtrueである場合には、INSERTするレコードまたはUPDATE後のレコードのcol1の値がゼロよりも大きくなければならない。
		
		'[{
		"type":"rowset",
		"object":[
		{
		"schema":"schema1",
		"table":"table1",
		"rowset_name":"rowset1",
		"rowset_expression": [
		{
		"as":"permissive",
		"using":"user = current_user"
		},
		{
		"as":"permissive",
		"using":"manager = current_user",
		"with check":"col1 > 0"
		}
		]
		}
		]
		}]'
		
		各キー(as、using、wich check)は、CREATE POLICY文の同じ名前の句と同じ意味です。
		
		このことからも分かるように、rowset_expressionに指定した配列の1つの要素は、CREATE POLICY文で作成する1つのPOLICYオブジェクトに対応します。
		
		実際に、この関数は内部的にCREATE POLICY文を配列の要素の数だけ実行します。このときのPOLICYの名前は、'pgx_cms_policy_${cpolid}'です。
		
		${cpolid}は、この拡張が自動的にナンバリングします。
		
		<注意>
		
		pgx_cms_policy_で始まるような名前のポリシーを作成しないでください。この関数が失敗する可能性があるためです。

pgx_remove_object_from_confidential_level(confidential_matrix_name varchar, confidential_level_name varchar, object_name json)	void	機密レベルから機密オブジェクトを除去します。 指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。

関数名	戻り値	説明
		confidential_level_nameで指定された機密レベルから、object_nameで指定された機密オブジェクトを除去します。 object_nameのフォーマットは、pgx_add_object_to_confidential_level関数のobject_nameと同じです。ただし、rowset_expressionキーは省略できます。また、指定されても単に無視します。これによって、単純にpgx_add_object_to_confidential_levelのobject_nameに指定したJSON形式の値を、修正せずに指定できます。 このときに、機密権限にもとづいて機密グループに付与されていた権限は、剥奪されます。 もし、機密オブジェクトがrowset型の場合には、内部的に作成していたポリシーを削除します。 このときに、テーブルに関連づけられたポリシーがない場合には、内部的にDISABLE ROW SECURITY句を付けたALTER TABLE文を実行して、行レベルセキュリティを無効にします。

B.6 ロール操作関数

関数名	戻り値	説明
pgx_add_role_to_confidential_group(confidential_matrix_name varchar, confidential_group_name varchar, role_name json)	void	機密グループにロールを追加します。 指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。 confidential_group_nameで指定された機密グループに、role_nameで指定されたロールを追加します。 追加されるロールに、機密権限よりも広い権限を付与されていたならば、機密権限に合わせて権限を剥奪します。 PUBLICへ権限を付与した機密オブジェクトをターゲットにするときには、注意が必要です。なぜならば、PUBLICへ権限を付与することは、機密マトリクスに登録されたすべてのロールへ権限を与えていることと同じだからです。もし、各ロールにPUBLICを利用して間接的に付与された権限が、そのロールに与えられるべきではないことが機密権限で定義されているならば、この関数は失敗します。同様に、機密マトリクスに登録されていないグループロールを利用して間接的に付与された権限も、この関数は検査します。このとき、継承のチェインを先祖まで検査します。 また、追加されるロールに、機密グループよりも強い属性が付与されていたならば、機密グループに合わせて属性を変更します。 機密マトリクスに登録されていないグループロールを利用して間接的に強い属性が付与されていたならば、この関数は失敗します。 role_nameは以下のように指定します。 '{["role1","role"]}'
pgx_remove_role_from_confidential_group(confidential_matrix_name varchar, confidential_group_name varchar, role_name json)	void	機密グループからロールを除去します。 指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。 confidential_group_nameで指定された機密グループから、role_nameで指定されたロールを除去します。

関数名	戻り値	説明
		この関数は、内部的にREVOKE文を実行して、ロールを機密グループロールから除去します。 除去されるロールの属性や、そのロールに付与されている権限などは変更しません。 権限を継承できないように、単にグループから追放するだけです。 role_nameの指定方法は、pgx_add_role_to_confidential_group()と同じです。

B.7 定義の参照およびシステムカタログとの比較を支援する関数

関数名	戻り値	説明
pgx_get_attribute_of_objects(confidential_matrix_name varchar)	setof record	指定した機密マトリクスに登録されているすべての機密オブジェクトに関して、機密マトリクスで規定されている属性と、実際にデータベース上で設定されている属性の表を返却します。表の形式は、“pgx_get_attribute_of_objectsが返却する表”を参照してください。 指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。
pgx_get_attribute_of_roles(confidential_matrix_name varchar)	setof record	指定した機密マトリクスに登録されているすべてのロールについて、機密マトリクスにて規定されている属性と、実際にシステムカタログで設定されている属性の表を返却します。表の形式は、“pgx_get_attribute_of_rolesが返却する表”を参照してください。 指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。
pgx_get_privileges_on_level_and_group(confidential_matrix_name varchar, confidential_level_name varchar, confidential_group_name varchar)	setof record	指定された機密レベルに登録されている機密オブジェクトと、指定された機密グループに登録されているロールの組み合わせについて、以下を比較できるような一覧の表を返却します。表の形式は、“pgx_get_privileges_on_level_and_groupが返却する表”を参照してください。 ・ 指定されたロールに付与されるべき、機密権限の設定値によって規定された権限 ・ 実際のシステムカタログで付与されている権限 指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。
pgx_get_privileges_on_object(confidential_matrix_name varchar, object_name json)	setof record	指定された機密オブジェクトについて、以下を比較できるような一覧の表を返却します。表の形式は、“pgx_get_privileges_on_objectが返却する表”を参照してください。 ・ すべてのロールに付与されるべき、機密権限の設定値によって規定された権限 ・ 実際のシステムカタログで付与されている権限 指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。
pgx_get_privileges_on_role(confidential_matrix_name varchar, role_name json)	setof record	すべての機密オブジェクトについて、以下を比較できるような一覧の表を返却します。表の形式は、“pgx_get_privileges_on_roleが返却する表”を参照してください。 ・ 指定されたロールに付与されるべき、機密権限の設定値によって規定された権限 ・ 実際のシステムカタログで付与されている権限

関数名	戻り値	説明
		指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。
pgx_get_privileges_on_matrix(confidential_matrix_name varchar)	setof record	指定された機密マトリックスに登録されたすべてのオブジェクトについて、以下を比較できるような一覧の表を返却します。表の形式は、“pgx_get_privileges_on_matrixが返却する表”を参照してください。 ・ 機密マトリクスに登録されたすべてのロールに付与されるべき、機密権限の設定値によって規定された権限 ・ 実際のシステムカタログで付与されている権限 指定された機密マトリクスの機密管理ロールだけが、この関数を実行できます。

pgx_get_attribute_of_objectsが返却する表

カラム名	型	意味
matrix_name	varchar(63)	機密マトリクス名
confidential_level_name	varchar(63)	機密レベル名
object_type	text	機密オブジェクトの種別
object_schema	name	機密オブジェクトのスキーマ名
object_table	name	機密オブジェクトのテーブル名
object_name	text	機密オブジェクトの名前
rowset_expression	json	機密オブジェクトが行の場合の条件式
encrypt_on_matrix	text	機密マトリクスで規定される暗号化手法と強度
encrypt_on_object	text	機密オブジェクトの実際の暗号化手法と強度

pgx_get_attribute_of_rolesが返却する表

カラム名	型	意味
matrix_name	varchar(63)	機密マトリクス名
confidential_group_name	varchar(63)	機密グループ名
role_name	name	ロールの名前
confidential_group_role	bool	機密グループロールであるか否かを示します。機密グループロールであるならば、trueです。
superuser_on_matrix	bool	機密マトリクスで規定されるSUPERUSER属性
superuser_on_role	bool	ロールの実際のSUPERUSER属性
createdb_on_matrix	bool	機密マトリクスで規定されるCREATEDB属性
createdb_on_role	bool	ロールの実際のCREATEDB属性
createrole_on_matrix	bool	機密マトリクスで規定されるCREATEROLE属性
createrole_on_role	bool	ロールの実際のCREATEROLE属性

カラム名	型	意味
replication_on_matrix	bool	機密マトリクスで規定されるREPLICATION属性
replication_on_role	bool	ロールの実際のREPLICATION属性
bypassrls_on_matrix	bool	機密マトリクスで規定されるBYPASSRLS属性
bypassrls_on_role	bool	ロールの実際のBYPASSRLS属性

pgx_get_privileges_on_level_and_group、pgx_get_privileges_on_object、pgx_get_privileges_on_roleおよびpgx_get_privileges_on_matrixが返却する表

カラム名	型	意味
matrix_name	varchar(63)	機密マトリクス名
confidential_level_name	varchar(63)	機密レベル名
confidential_group_name	varchar(63)	機密グループ名
object_type	text	機密オブジェクトの種別
object_scheme	name	機密オブジェクトのスキーマ名
object_table	name	機密オブジェクトのテーブル名
object_name	text	機密オブジェクトの名前
role_name	name	ロール名
privilege_list_on_matrix	text[]	機密マトリクスの設定により規定されている権限をカンマ区切りで出力する。
privilege_list_on_object	text[]	機密マトリクスの設定により規定されている権限をカンマ区切りで出力する。
policy_name	name	機密オブジェクトがrowset型ではない場合にはNULL。 rowset型の場合にはrowsetの名前(注1)
policy_setting_on_matrix	jsonb	機密オブジェクトがrowset型ではない場合にはNULL。 rowset型の場合には、機密マトリクスに設定されているrowsetのポリシー情報(注1)
policy_setting_on_policy	jsonb	機密オブジェクトがrowset型ではない場合にはNULL。 Rowset型の場合には、実際のポリシーに設定されている行のポリシー情報(注1)

注1: rowset型の機密オブジェクトを追加するときには、複数の権限を一度に設定できますが、この表では一つの行で表現されません。例えば、SELECT権限とDELETE権限を設定したならば、SELECT権限を表す行と、DELETE権限を表す行とが出力されます。なぜならば、rowset型のアクセス制御は、PostgreSQLの行レベルセキュリティのPOLICYを使っているからです。この仕様では、SELECT権限のためのPOLICYと、DELETE権限のためのPOLICYは異なります。