Hewlett Packard
Enterprise

PG_TDE 検証

Noriyoshi Shinoda

March 27, 2025

SPEAKER

Featured in
The Redgate 100

- ✓篠田 典良（しのだ のりよし）
- ✓所属
  - ✓日本ヒューレット・パッカード合同会社
- ✓現在の業務など
  - ✓PostgreSQLをはじめ Oracle Database, Microsoft SQL Server, V
    RDBMS 全般に関するシステムの設計、移行、チューニング、コンサルティング
  - ✓「篠田の虎の巻」作成
  - ✓PostgreSQL 開発（PostgreSQL 10～17, 18 dev）
  - ✓Oracle ACE Pro (2009～)
- ✓関連する URL
  - ✓「PostgreSQL 虎の巻」シリーズ
    https://github.com/nori-shinoda/documents/blob/main/README.md
  - ✓Redgate 100 in 2022 (Most influential in the database community 2022)
    https://www.red-gate.com/hub/redgate-100/
  - ✓Oracle ACE Profile
    https://ace.oracle.com/apex/ace/profile/nshino483

はじめに
TDE とは？

- ✓Transparent Data Encryption = 透過的データ暗号化
  - ✓RDBMS に格納されるデータの暗号化機能
  - ✓認証を通過したユーザーには暗号化を意識させない
  - ✓ファイルシステム層や物理バックアップからの情報流出を防ぐ
- ✓TDE 実装
  - ✓pg_tde エクステンション
  - ✓EDB Postgres Advanced Server
  - ✓PowerGres Plus
  - ✓MySQL Enterprise Edition
  - ✓Oracle Database Enterprise Edition
  - ✓Microsoft SQL Server Enterprise Edition
  - ✓IBM Db2

はじめに
pg_tde とは？

- ✓TDE 機能を実現する PostgreSQL エクステンション
- ✓Percona が作成
  - ✓最新は 1.0.0 Beta 2
  - ✓Source https://github.com/percona/pg_tde
  - ✓Document https://percona.github.io/pg_tde/main/
- ✓サポートする暗号化鍵の管理方法
  - ✓Keyring ファイル
  - ✓HashiCorp Vault
  - ✓KMIP Server
- ✓ライセンス
  - ✓Percona Server for PostgreSQL
  - ✓Community version

インストールとセットアップ

インストールとセットアップ
Red Hat Enterprise Linux のパッケージ

- ✓Percona が提供する YUM リポジトリを利用
  - ✓ソースコードからのビルドを断念
  - ✓pg_tde エクステンションは percona-postgresql17-contrib パッケージに含まれる
- ✓インストールに使用したコマンド

```
# yum install https://repo.percona.com/yum/percona-release-latest.noarch.rpm
# percona-release enable-only ppg-17
# yum install percona-postgresql17
# yum install percona-postgresql17-server
# yum install percona-postgresql17-contrib
```

インストールとセットアップ
初期設定

✓インスタンス起動時（shared_preload_libraries = pg_tde 設定時）

```
$ pg_ctl -D data start
waiting for server to start....
LOG:  Initializing TDE principal key info
LOG:  initializing TDE key provider info
LOG:  registered custom resource manager "test_tdeheap_custom_rmgr" with ID 140
LOG:  tde_shmem_request: requested 5017672 bytes
LOG:  initializing shared state for principal key
LOG:  creating DSA area of size 819200
LOG:  initializing dsa area objects for principal key
LOG:  setting no limit to DSA area of size 819200
LOG:  redirecting log output to logging collector process
HINT:  Future log output will appear in directory "log".
 done
server started
```

インストールとセットアップ
初期設定

✓必要な初期設定
- ✓GUC shared_preload_libraries に pg_tde を設定
- ✓CREATE EXTENSION pg_tde 文の実行（データベース単位）

✓バージョン確認

```
postgres=# SELECT pg_tde_version() ;
   pg_tde_version
--------------------
 pg_tde 1.0.0-beta2
(1 row)
```

インストールとセットアップ
初期設定

✓暗号化鍵プロバイダの設定

```
postgres=# SELECT pg_tde_add_key_provider_file('keyfile1', '/key/tde1.key') ;
 pg_tde_add_key_provider_file
------------------------------
                            1
(1 row)
```

✓暗号化鍵プロバイダは複数作成できる

鍵管理方法	関数名	追加情報
ファイル	pg_tde_add_key_provider_file	ファイル名
HashiCorp Vault	pg_tde_add_key_provider_vault_v2	トークン、URL など
KMIP Server	pg_tde_add_key_provider_kmip	IP アドレス、PEM ファイルなど

インストールとセットアップ
初期設定

✓プリンシパル・キーの設定

```
postgres=# SELECT pg_tde_set_principal_key('principal1', 'keyfile1') ;
 pg_tde_set_principal_key
--------------------------
 t
(1 row)
```

✓暗号化鍵ローテーション

```
postgres=# SELECT pg_tde_rotate_principal_key() ;
 pg_tde_rotate_principal_key
-----------------------------
 t
(1 row)
```

インストールとセットアップ
初期設定

✓キーの保存
- ✓$PGDATA/pg_tde ディレクトリが作成される
- ✓以下のファイルが作成される

ファイル名	用途	備考
pg_tde_{DBOID}_map	鍵プロバイダ情報？	
pg_tde_{DBOID}_data	暗号化鍵情報？	
pg_tde_{DBOID}_keyring	Keyring ファイルのパス	

機能確認

機能確認
暗号化テーブルの作成

✓テーブル・アクセスメソッドが追加される
- ✓tde_heap = Percona Server for PostgreSQL
- ✓tde_heap_basic = Community Version

```
postgres=> SELECT * FROM pg_am WHERE amname LIKE '%tde%' ;
  oid  |      amname      |         amhandler          | amtype
-------+------------------+----------------------------+--------
 16418 | tde_heap_basic   | pg_tdeam_basic_handler     | t
 16420 | tde_heap         | pg_tdeam_handler           | t
(2 rows)
```

機能検証
暗号化テーブルの作成

✓暗号化対象テーブルは pg_tde 用のアクセスメソッドを使用

```
postgres=> CREATE TABLE tde1(c1 INT PRIMARY KEY, c2 TEXT) USING tde_heap ;
CREATE TABLE
Postgres=> ¥d+ tde1
                                          Table "public.tde1"
 Column |  Type   | Collation | Nullable | Default | Storage  | Compression | …
--------+---------+-----------+----------+---------+----------+-------------+-…
 c1     | integer |           | not null |         | plain    |             | …
 c2     | text    |           |          |         | extended |             | …
Indexes:
    "tde1_pkey" PRIMARY KEY, btree (c1)
Access method: tde_heap
```

機能検証
暗号化テーブルの作成

✓暗号化されているかを確認

✓pg_tde_is_encrypted 関数は SUPERUSER のみ実行可能（他のロールに GRANT 可能）

```
postgres=# SELECT pg_tde_is_encrypted('tde1') ;
 pg_tde_is_encrypted
---------------------
 t
(1 row)
```

機能検証
アクセスメソッド比較

✓アクセスメソッドの違い

✓暗号化範囲の違い

比較	tde_heap	tde_heap_basic	備考
TABLE	○	○	
INDEX	○	×	
MATERIALIZED VIEW	○	○	USING 句で指定可能
一時データ	○	○	
WAL	全データ	暗号化テーブルのみ	

✓URL: https://docs.percona.com/pg-tde/features.html

機能検証
アクセスメソッド比較

✓実行計画の比較
- ✓インデックス一致検索、範囲検索も実行可能

```
postgres=> EXPLAIN SELECT * FROM tde1 WHERE c1=1000 ;
                                QUERY PLAN
-------------------------------------------------------------------------
 Index Scan using tde_pkey on tde1  (cost=0.43..8.45 rows=1 width=25)
   Index Cond: (c1 = 1000)
(2 rows)

postgres=> EXPLAIN SELECT * FROM tde1 WHERE c2 BETWEEN '1000' AND '2000' ;
                                    QUERY PLAN
-----------------------------------------------------------------------------------
 Index Scan using idx1_tde1 on tde1  (cost=0.43..143892.38 rows=1124749 width=25)
   Index Cond: (((c2)::text >= '1000'::text) AND ((c2)::text <= '2000'::text))
(2 rows)
```

機能検証
キーアクセス障害

- ✓pg_tde ディレクトリ障害
  - ✓インスタンス起動不可

```
PANIC:  could not locate a valid checkpoint record at 14/43000028
LOG:  startup process (PID 6351) was terminated by signal 6: Aborted
LOG:  aborting startup due to startup process failure
LOG:  database system is shut down
```

機能検証
キーアクセス障害

- ✓暗号化鍵プロバイダに対するアクセス
  - ✓初回テーブルにアクセスした時に暗号化鍵をオープン
  - ✓（おそらく）共有メモリーにキャッシュ
- ✓暗号化鍵にアクセスできない場合
  - ✓インスタンスは起動できる
  - ✓暗号化テーブルに対するアクセスはエラーになる

```
postgres=> SELECT * FROM tde1 ;
ERROR:  invalid page in block 0 of relation base/5/17643
```

機能検証
GUC

✓WAL 暗号化はオプション（Tech Preview）
  ✓GUC は pg_tde.wal_encrypt（デフォルト off）のみ

```
postgres=# SHOW pg_tde.wal_encrypt ;
 pg_tde.wal_encrypt
--------------------
 on
(1 row)
```

パフォーマンスとデータ量

機能検証
アクセスメソッド比較

✓データのサイズ
- ✓4列（INTEGER (PK), TEXT, TEXT, TEXT）+ 1インデックス
- ✓1,000万件の INSERT 文実行により比較

比較	heap	tde_heap	tde_heap_basic	備考
テーブル	574 MB	574 MB	574 MB	ファイル・サイズ
インデックス	214 MB	214 MB	214 MB	
WAL	2,256 MB	2,240 MB	2,240 MB	TDE 環境のみ WAL 暗号化有効化

- ✓pg_stats ビューの内容は同じ
- ✓I/O 量に明確な差は無かった

機能検証
アクセスメソッド比較

- ✓パフォーマンス比較
  - ✓pgbench / HammerDB で検証したが、有意な差は見られなかった。
  - ✓環境の問題で実行毎の差が大きすぎた

実装比較

比較
他の TDE 実装との比較

✓機能比較

✓EDB Postgres Advanced Server / Oracle Database との比較

比較	pg_tde	EDB Postgres	Oracle Database
暗号化対象	テーブル、インデックス	クラスタ全体	列、テーブル、表領域
暗号化方法の指定	アクセス・メソッド	-	オブジェクト属性
キーストア	pg_tde/pg_tde_*	pg_encryption/key.bin	encryption_wallet_location で指定
暗号化鍵管理	File / HashiCorp Vault / KMS Server	data_encryption_key_unwrap_command で指定する任意コマンド	PKCS#11 HSM / Wallet File
暗号化メソッド（Data）	AES-CBC	AES-XTS	AES-CBC, 3DES
暗号化メソッド（WAL）	ASE-CTR	AES-CTR	AES-CFB, 3DES
鍵のサイズ（ビット）	128	128 / 256	128 / 192 / 256
ハードウェア最適化	-	-	Intel AES-NI

まとめ

まとめ

- ✓機能の検証
  - ✓テーブル作成時以外は暗号化を意識する必要が無い ⇒ default_table_access_method 指定でより簡単に
  - ✓実行計画は既存の heap と変わらない
  - ✓鍵管理システムへのアクセス情報が永続化される ⇒ 物理バックアップを取得すると自動復号される？
- ✓マニュアルが貧弱
  - ✓pg_tde を含む RPM パッケージの情報が無い
  - ✓ドキュメントが無い FUNCTION 多数
    - ✓pg_tde_list_all_key_providers
    - ✓pg_tde_principal_key_info
    - ✓pg_tde_version
    - ✓Etc, …
- ✓パフォーマンスの検証
  - ✓有意な差は確認できなかった

THANK YOU

Mail : noriyoshi.shinoda@hpe.com
X(Twitter) : @nori_shinoda
Qiita : @plusultra