FUJITSU

FUJITSU Enterprise Postgres
Dedicated Audit Log

Data drives today's information economy, and the protection and accountability of that data is not only expected, but increasingly dictated by law. Compliance with the requirements of government, financial, and various certification authorities is driving the need for database management systems to provide mechanisms that can track and analyse what data was accessed, when it was accessed, how it was accessed, and by whom.

Content

About FUJITSU Enterprise Postgres

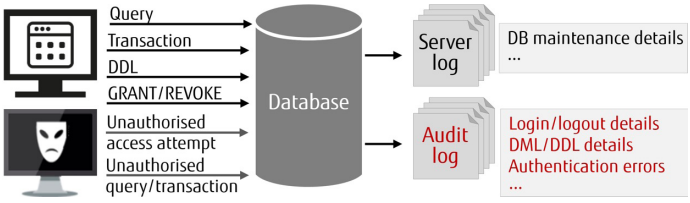
Founded on PostgreSQL, the world's most advanced open source relational database system, FUJITSU Enterprise Postgres extends PostgreSQL functionality with a number of enhanced enterprise features.

About the dedicated audit log

A feature unique to FUJITSU Enterprise Postgres is a dedicated audit log. This feature delivers in the key areas of data accountability, traceability, and auditability. The log is not only Payment Card Industry Data Security Standard (PCI DSS) compliant but provides a level of protection that satisfies clients' expectations and protects the brand and data asset value of an organisation.

Overview

There is an increasing expectation in society that organisations provide the necessary security required to prevent any abuse of personal information. Data and information in general is now seen as an asset with a value attached to it – any security issue associated with that data naturally leads not only to devaluation of that asset, but also to a corporation's brand. FUJITSU Enterprise Postgres provides a data platform that constantly monitors and secures data thereby helping to not only protect but grow the value of the data asset.



Security information is directed to a dedicated audit log

The audit log provided by FUJITSU Enterprise Postgres can be used to counter security threats such as spoofing, unauthorised access to the database, application manipulation, and misuse of privileges. Administrators can configure the types of operation that are logged, which allows for more efficient and accurate operation monitoring.

Details relating to database access are stored in a dedicated audit log by extending PostgreSQL Audit extension (pgaudit) functionality. PostgreSQL outputs audit log records to the server log only, which makes operation and analysis complex and time consuming (because output criteria cannot be specified, and since log volumes can be large, performance will deteriorate). On the other hand, the FUJITSU Enterprise Postgres audit log allows a flexible and efficient way to accurately monitor logs. And being an independent output destination, existing monitoring of messages in the server log is not affected.

PCI DSS compliance

PCI DSS is a set of security standards designed to ensure that all companies that accept, process, store or transmit credit card information maintain a secure environment.

FUJITSU Enterprise Postgres is compliant with all PCI DSS requirements. The table below lists the PCI DSS requirements that relate specifically to audit logs (audit trails).

PCI DSS requirement	Database security measure
Requirement 10	Track and monitor all access to network resources and cardholder data
10.1	Implement audit trails to link all access to system components to each individual user.
10.2.	Implement automated audit trails for all system components to reconstruct the following events
10.2.1.	All individual user accesses to cardholder data
10.2.2.	All actions taken by any individual with root or administrative privileges
10.2.3	Access to all audit trails
10.2.4	Invalid logical access attempts
10.2.5	Use of and changes to identification and authentication mechanisms – including but not limited to creation of new accounts and elevation of privileges-and all changes, additions, or deletions to accounts with root or administrative privileges
10.2.6	Initialisation, stopping, or pausing of the audit logs
10.2.7	Creation and deletion of system-level objects
10.3	Record at least the following audit trail entries for all system components for each event: • User identification (PCI DSS requirement 10.3.1) • Type of event (PCI DSS requirement 10.3.2) • Date and time (PCI DSS requirement 10.3.3) • Success or failure indication (PCI DSS requirement 10.3.4) • Origination of event (PCI DSS requirement 10.3.5) • Identity or name of affected data, system component, or resource (PCI DSS requirement 10.3.6)
10.4	Using time-synchronisation technology, synchronise all critical system clocks and times and ensure that the following is implemented for acquiring, distributing, and storing time • Critical systems have the correct and consistent time
10.5	Secure audit trails so they cannot be altered
10.5.1	Limit viewing of audit trails to those with a job-related need
10.5.2	Protect audit trail files from unauthorised modifications
10.5.3	Promptly back up audit trail files to a centralised log server or media that is difficult to alter
10.5.5.	Use file-integrity monitoring or change-detection software on logs to ensure that existing log data cannot be changed without generating alerts
10.6	Review logs and security events for all system components to identify anomalies or suspicious activity. Note: Log harvesting, parsing, and alerting tools may be used to meet this requirement
10.7	Retain audit trail history for at least one year, with a minimum of three months immediately available for analysis (for example, online, archived, or restorable from backup)

Why use pgaudit?

PostgreSQL provides basic statement logging via the standard logging facility with "log_statement = all". This is acceptable for monitoring and other usages but does not provide the level of detail generally required for an audit. It is not enough to simply list what the user requested. It must also be possible to establish what happened while the database was satisfying each request, and that is the information that pgaudit provides.

Audit log output modes

In pgaudit, two types of audit log can be output.

Session audit logging

Session audit logging provides a detailed log of all SQL executed by a user in the backend, information relating to starting and connecting databases, and information related to errors.

In session audit logging, by specifying the log output conditions and filtering the log records to be output, performance degradation due to outputting large volumes of log records can be prevented.

Object audit logging

When SELECT, INSERT, UPDATE, or DELETE are executed for specific objects (tables and columns), object audit logging outputs these as a log. Also logged are object operations for which privileges have been assigned to specified roles.

Object audit logging can control log output at an even finer level of granularity than session audit logging.

Pgaudit setup

1. Create the pgaudit configuration file

The pgaudit configuration file describes the information required for pgaudit actions.

FUJITSU Enterprise Postgres can output audit records to a dedicated audit log or to the server log.

■ To output to a dedicated audit log, omit the logger parameter or set it to 'auditlog'

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

Also specify the log directory and name, permissions, and rotation settings.

Audit records will be output as audit log records, therefore the status information and message severity level according to the log_line_prefix parameter in postgresql.conf will not be output.

■ To output to the server log, set the logger parameter to 'serverlog'

```
[output]
logger='serverlog'
```

Audit records will be output as log messages, therefore the status information and message severity level according to the log_line_prefix parameter in postgresql.conf will be output to the beginning of each audit record.

2. Configure postgresql.conf

The pgaudit extension must be loaded in shared_preload_libraries.

Example of postgresql.conf
In the example below, only the parameters that need to be configured when using the dedicated audit log feature are described

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

3. Start the instance

4. Load the pgaudit extension

Use CREATE EXTENSION to load the pgaudit extension.

```
CREATE EXTENSION pgaudit;
```

5. Configure the parameters in the pgaudit configuration file

Specify the information required for pgaudit actions. The pgaudit configuration file comprises three sections:

- output
- option
- rule

The output section and the option section can only be defined once, whereas multiple rule sections can be defined.

5. Restart the instance

This will apply the changes to the pgaudit configuration file.

Session audit logging

Configuration

The session audit logging parameters are configured in the rule section of the pgaudit configuration file.

The following values specify which classes of statements will be logged by session audit logging:

- **READ:** SELECT, COPY FROM
- **WRITE:** INSERT, UPDATE, DELETE, TRUNCATE, COPY TO
- **FUNCTION:** Function call, DO
- **ROLE:** GRANT, REVOKE, CREATE ROLE, ALTER ROLE, DROP ROLE
- **DDL:** All DDLs (such as CREATE and ALTER) other than the DDLs of the ROLE class
- **CONNECT:** Events relating to connecting (request, authenticate, and disconnect)
- **SYSTEM:** Instance start, promotion to primary server
- **BACKUP:** pg_basebackup
- **ERROR:** Event completed by an error (PostgreSQL error codes other than 00). This class can be used if ERROR or lower level is specified for the log_min_messages parameter in postgresql.conf.
- **MISC:** Other commands (such as DISCARD, FETCH, CHECKPOINT, and VACUUM)

Multiple classes can be specified using a comma-separated list, for example, class = 'READ, WRITE'.

Output format

The following items are logged:

- Log header: "AUDIT: SESSION"
- Class
- SQL start time
- Host name
- Backend process ID
- Application name
- User name
- Database name
- Virtual transaction ID
- Statement ID
- Substatement ID
- Command tag
- SQLSTATE
- Object type
- Object name
- Error message
- SQL
- SQL parameters

Sample log entries

```
AUDIT: SESSION,WRITE,2017-03-12 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', 'some description');",<not logged>
AUDIT: SESSION,READ,2017-03-12 19:00:58 PDT,[local],
 19944,psql,appuser,postgres, 2/8,2,1,SELECT,,TABLE,
 myschema.account,,SELECT * FROM myschema.account;,
 <not logged> myschema.account;,<not logged>

 19944,psql,appuser,postgres,2/7,1,1,INSERT,,TABLE,
 myschema.account,,"INSERT INTO myschema.account
 (id, name, password, description) VALUES (1, 'user1',
 'HASH1', 'some description');",<not logged>
AUDIT: SESSION,READ,2017-03-12 19:00:58 PDT,[local],
 19944,psql,appuser,postgres, 2/8,2,1,SELECT,,TABLE,
 myschema.account,,SELECT * FROM myschema.account;,
 <not logged> myschema.account;,<not logged>
```

Object audit logging

Configuration

Object-level audit logging is implemented using roles. Define the role that will be used for audit logging in the role parameter of the option section. If there are privileges for commands executed by a role, or if privileges have been inherited from another role, those command operations are output as audit records.

Output format

The following items are logged:

- Log header: "AUDIT: OBJECT"
- Statement ID
- Substatement ID
- Class name
- Command tag
- Object type
- Object name
- SQL
- SQL parameters

Sample log entries

```
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>
```

Viewing the audit logs

By using the contrib module file_fdw, the audit logs can be accessed using SQL. As an example, the procedure below explains how to view the audit log of session audit logging, which has been output to a dedicated log file.

1. Load file_fdw

   Use CREATE EXTENSION to load the file_fdw extension.

```
CREATE EXTENSION file_fdw;
```

2. Create an external server

   Use CREATE SERVER to create an external server managed by file_fdw

```
CREATE SERVER auditlog FOREIGN DATA WRAPPER file_fdw;
```

3. Create an audit log table

   Use CREATE FOREIGN TABLE to define the table columns of the audit log, the CSV file name, and format.

```
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-2017-03-12.log', format 'csv' );
```

4. View the audit log

   Use SELECT to view the audit log

```
SELECT * FROM auditlog;
     header          |  class    |    sql_start_time        | ...
---------------------+-----------+--------------------------+----
AUDIT: SESSION       | WRITE     | 2017-03-12 19:00:49+09   | ...
AUDIT: SESSION       | READ      | 2017-03-12 19:00:58+09   | ...
```

Published by
Fujitsu Australia Software Technology Pty Ltd
14 Rodborough Road, Frenchs Forest NSW 2086, Australia

Document number 14975a. 2018-10-02 WW EN

Conclusion

The dedicated audit log feature of FUJITSU Enterprise Postgres further enhances data accountability, traceability, auditability and compliance. These are critical areas that can make or break your business. Not only do your customers expect their personal information to be safe, but various authorities demand it.

Contact us

If you have any questions about the audit logging features of FUJITSU Enterprise Postgres or how to implement a secure and accountable database system, please contact our support team.

About Fujitsu

Fujitsu is the 5th largest IT service provider in the world, offering a full range of technology products, solutions and services. Around 160,000 Fujitsu employees support customers in over 100 countries.

Published by
Fujitsu Australia Software Technology Pty Ltd
14 Rodborough Road, Frenchs Forest NSW 2086, Australia

Document number 14975b. 2018-10-02 WW EN