Data Protection in Relational Databases

A Guide for Database Administrators and Developers

# Introduction

Data protection is a crucial aspect of database management, as it ensures the confidentiality, integrity, and availability of the data stored in the database. Data protection also helps to comply with the legal and ethical requirements of data privacy and security. In this document, we will discuss some of the best practices for data protection in relational databases, focusing on the topics of access control, encryption, and audit logging.

# Access Control

Access control is the process of granting or denying permissions to users or roles to access, modify, or perform operations on the data in the database. Access control helps to prevent unauthorized or malicious access to the data, as well as to enforce the principle of least privilege, which states that users or roles should only have the minimum level of access required to perform their tasks. Access control can be implemented at different levels of the database, such as the schema, table, column, row, or cell level. The view and the stored procedures also need to have controlled access.

Some of the best practices for access control in relational databases are:

* Use a strong authentication mechanism to verify the identity of the users or roles accessing the database. For example, use passwords, tokens, certificates, or biometrics to authenticate users or roles.
* Use a role-based access control (RBAC) model to assign permissions to users or roles based on their roles or responsibilities. For example, create roles such as admin, manager, analyst, or developer, and grant them different levels of access to the data in the database.
* Use a fine-grained access control (FGAC) model to restrict access to specific data elements based on the context or conditions of the access request. For example, use views, triggers, functions, or procedures to filter, mask, or redact the data based on the user, role, time, location, or purpose of the access request.
* Use a data classification scheme to categorize the data in the database based on their sensitivity or importance. For example, use labels such as public, internal, confidential, or secret to classify the data, and apply different access control policies based on the data classification.
* Use a data retention policy to specify how long the data should be stored in the database, and how it should be disposed of when it is no longer needed. For example, use policies such as deletion, anonymization, or encryption to remove or protect the data after a certain period of time. Personal Identifiable Information (PII) must be deleted in 7 days when a user requests opt-out from our service.
* Use a data backup and recovery plan to ensure that the data can be restored in case of a data loss or corruption event. For example, use backup tools, schedules, and locations to create and store copies of the data, and use recovery tools, procedures, and tests to restore the data in case of a disaster. We take snapshots of the database every 15 minutes, and incremental backups of the database are taken daily. A full backup of the database is performed once a month.

# Encryption

Encryption is the process of transforming the data in the database into an unreadable or unintelligible form, using a secret key or algorithm. Encryption helps to protect the data from unauthorized or malicious access, modification, or disclosure, as well as to comply with the legal and ethical requirements of data privacy and security. Encryption can be applied at different levels of the database, such as the data-at-rest, data-in-transit, or data-in-use level.

Some of the best practices for encryption in relational databases are:

* Use a strong encryption algorithm and key to encrypt the data in the database. For example, use algorithms such as AES256, RSA, or ECC, and keys of sufficient length and randomness to encrypt the data. PII must be encrypted.
* Use a secure key management system to store, distribute, rotate, and revoke the encryption keys.
* Use a transparent data encryption (TDE) feature to encrypt the data at rest in the database. For example, use TDE to encrypt the entire database, the database files, or the database columns, without requiring any changes to the application code or queries.
* Use a secure transport layer security (TLS) protocol to encrypt the data in transit between the database and the application. For example, use TLS to encrypt the communication channel, the data packets, or the data fields, using certificates and ciphers to establish and maintain the encryption.
* Use a homomorphic encryption (HE) technique to encrypt the data in use in the database. For example, use HE to encrypt the data before performing any computations or operations on it, and decrypt the results after the computations or operations are done.

# Audit Logging

Audit logging is the process of recording the activities or events that occur in the database, such as the authentication, authorization, access, modification, or deletion of the data, or the execution of the queries or commands. Audit logging helps to monitor and audit the data usage and behavior in the database, as well as to detect and respond to any anomalies or incidents. Audit logging can be configured at different levels of the database, such as the database, the schema, the table, the column, or the row level. The log must be kept for 24 months and make sure the logs are stored in an immutable storage to prevent alteration.

Some of the best practices for audit logging in relational databases are:

* Use a comprehensive and consistent audit policy to define what, when, where, how, and by whom the data activities or events should be logged. For example, use a policy to specify the types, levels, formats, and destinations of the audit logs, and the IP address, users, roles, or actions that should be logged.
* Use a reliable and secure audit meachanism to generate, store, and transmit the audit logs. For example, use a mechanism that ensures the accuracy, completeness, integrity, and availability of the audit logs, and protects them from unauthorized or malicious access, modification, or deletion.
* Use an efficient audit system to manage and analyze the audit logs. For example, use a system that can handle the volume, velocity, and variety of the audit logs, and provide the functionality, performance, and usability of the audit analysis.
* Use a proactive and reactive audit strategy to review and act on the audit logs. For example, use a strategy that involves regular and automated audit reviews, alerts, and reports, as well as timely and appropriate audit responses, investigations, and resolutions.