
Enhancing Relational Database Security by Metadata Segregation

MYSQL Technical Paper Implementation



SUJITH MEDISETTY
U00881475



Introduction / Abstract








This presentation introduces a database system aimed at safeguarding sensitive information. Through the division of data into smaller tables based on its sensitivity (segregation), security is ensured while upholding data integrity. The utilization of encryption for sensitive data adds an additional layer of protection. The system's workflow dynamically establishes logical relationships between tables, thereby ensuring data integrity. In essence, this approach provides a solution for protecting sensitive data within databases.

Segregating Information based on its sensitivity level






To mitigate the risk unauthorized access, sensitive data is partitioned into distinct tables. The entity relationship diagram reflects this segregation, emphasizing the separation of highly sensitive attributes such as **Patient_Health_Insu_No** and **Patient_Social_Insu_No**. By relocating these attributes to separate tables—**Patient_Health_INS_Table** and **Patient_Social_INS_Table**—the system reinforces security measures for heightened protection.

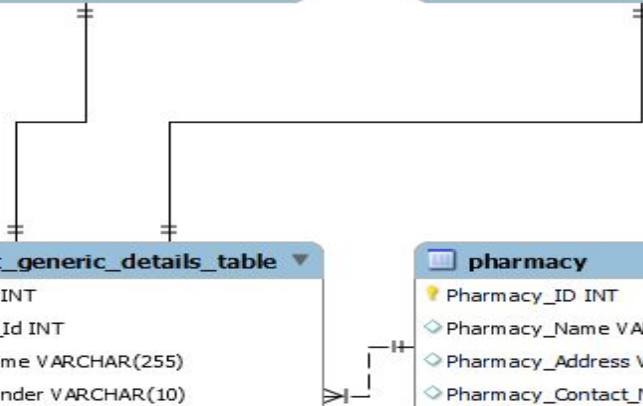
patient_social_ins_table	
	Patient_Id INT
	Patient_Social_Insu_No VARBINARY(255)
Indexes	
Triggers	

patient_health_ins_table	
	Patient_Id INT
	Patient_Health_Insu_No VARBINARY(255)
Indexes	
Triggers	

patient_generic_details_table	
	Patient_Id INT
	Pharmacy_Id INT
	Patient_Name VARCHAR(255)
	Patient_Gender VARCHAR(10)
	Patient_Birthdate DATE
	Patient_Address VARCHAR(255)
	Patient_Contact_No VARCHAR(15)
Indexes	

pharmacy	
	Pharmacy_ID INT
	Pharmacy_Name VARCHAR(255)
	Pharmacy_Address VARCHAR(255)
	Pharmacy_Contact_No VARCHAR(15)
Indexes	

operation_log	
	Log_ID INT
	Timestamp TIMESTAMP
	Operation VARCHAR(50)
	Table_Name VARCHAR(255)
	Row_ID INT
Indexes	



Dynamic Referential Integrity

Instead of directly establishing referential integrity within tables, the system employs predefined PL/SQL procedures to create relationships at runtime. These procedures, residing in the database engine, dynamically establish logical connections between tables based on parameters such as table names and column names. This dynamic approach enhances security by preventing reverse engineering attacks and ensures data integrity in real-time.

Please refer the [code here](#)

Reducing Unauthorized Access via Encryption

By combining data segregation with encryption techniques, the risk of unauthorized access is significantly mitigated. Triggers are implemented to encrypt sensitive information, such as Patient_Social_Insu_No and Patient_Health_Insu_No, during insertion into their respective tables. This encryption ensures that even if unauthorized parties gain partial access, the obtained information remains minimal.

Please refer the [code here](#)

Audit Logging

To ensure traceability, triggers are implemented to log insertions, deletions, and updates on sensitive tables like Patient_Social_INS_Table and Patient_Health_INS_Table. The Operation_Log table captures the operation type, affected table, and corresponding row ID for comprehensive auditing. Provides a means to trace logs in case of unauthorized access or data manipulation attempts.

Please refer the [code here](#)

Performance Evaluation

The performance evaluation demonstrates varying execution times for inserting and retrieving records under different configurations. Inserting 10,000 records ranges from 32 to 37 seconds, with the highest time observed when encryption and log triggers are both active. Retrieving 10,000 records without decryption shows minimal time impact, while decryption introduces a slight delay, with the highest recorded time being 15 seconds. These findings highlight the trade-offs between security measures and performance in the database system.

Please see the [execution times here](#)

For validation screenshots, Paper Link, ER diagram and more please refer the [readme](#) file or [click here](#)

THANK YOU