SQL Change Guard

Compliance and Data Protection

Audit, Evidence and Data Protection

KVKK | ISO 27001 | COBIT | ITIL

1. Approach

SQL Change Guard turns database changes and production data access into an auditable process. Every action becomes audit evidence, which makes meeting regulatory requirements easier.

2. KVKK and Personal Data Protection

- Sensitive data detection: patterns such as national ID, card number, IBAN and phone captured via regex and validation algorithms (checksum, Luhn, mod-97)
- Automatic masking: sensitive columns masked with full mask or partial mask
- Extra approval for unmasked data: an unmasked request requires separate security approval
- Encrypted delivery: query results shared as an encrypted ZIP
- Traceability: a record of who took which data and for what reason is kept (data processing evidence)
- On-premise: personal data never leaves the corporate network

3. Audit Trail

- DML audit trail: date, user and environment record of INSERT/UPDATE/DELETE
- DDL audit trail: CREATE/ALTER/DROP tracked with object history
- DCL/privilege changes: complete record of GRANT, REVOKE, CREATE/DROP USER
- Approval evidence: timestamped record of the approver for each action
- Immutable logs: hash-chained audit that cannot be deleted
- Evidence integrity verification: audit records can be verified, with one click, as unchanged since creation by comparison against a separate immutable copy

4. Standard Mapping

Standard / Requirement	SQL Change Guard Contribution
KVKK / GDPR	**Sensitive data masking, access approval, processing record, encrypted delivery**
ISO 27001	**Access control, change management, audit logging, separation of duties**

COBIT	**Change governance, approval workflows, provable controls**
ITIL	**Change management process, release management, record keeping**
Banking / PCI DSS	**Evidence of who accessed what, who approved, who executed**

5. Audit Evidence Package

The full lifecycle of a change or data access can be gathered into a single evidence package: request, parse results, risk score, approvals, execution log, rollback and audit records. 30 built-in reports can be provided to auditors in Excel and PDF.

In addition, the full lifecycle of any single request can be downloaded as one sealed evidence file (PDF and JSON) with a single click. The file carries a package seal (SHA-256) and can be verified through the system as unchanged since it was generated. Permission to download this file is governed by a separate right, and the download itself is audited.

Segregation of duties is reported in four states: satisfied, not enforced, violated, not applicable. A request that produced no approval step is never reported as satisfied. The evidence dossier also shows the person who triggered the execution separately from the worker service, and lists the rule set the request went through in a control environment section. This way the document tells the truth even in a loosely configured organisation.

Paths that could bypass the approval flow are closed: release packages accept approved requests only and revalidate the approval status and segregation of duties for every request at execution time, marking a request as manually applied is subject to the requester restriction, and raising an emergency request requires a minimum role. These three paths are the bypass scenarios auditors question most often.

Risk band and applied policy: The evidence dossier contains the risk band of the request, the rule that determined the band, whether a non-skippable standard was triggered, and the name of the approval policy applied to the request. When no policy matched, that fact is recorded as well. This makes it possible to prove afterwards why a request required no approval.

6. Compliance Reports (Examples)

- Critical object access and sensitive column touches
- Manual data modifications and detection of operations without a WHERE clause
- DB permission movements, privileged account operations, user permission matrix
- QR masking log and query result activity

7. Disclaimer

SQL Change Guard provides a control and evidence infrastructure that supports compliance processes. Ultimate compliance responsibility should be assessed together with the organization's own policies and practices.

Encryption and Key Management

SQL Change Guard protects sensitive data in three distinct categories, each with a cryptographic method suited to its purpose. Secrets that never need to be recovered are hashed one-way; secrets

that the application must recover in order to operate are stored with authenticated encryption. This ensures every piece of data is protected with the most appropriate, industry-standard technique.

1. User passwords- BCrypt (one-way hash)

User passwords are stored hashed with BCrypt (work factor 12, with a random per-record salt). This is a one-way operation: the plaintext password is never stored anywhere and cannot be recovered. Authentication is performed by re-hashing the entered password and comparing it to the stored hash. Even if the database is compromised, passwords cannot be reversed.

2. Server database passwords- AES-256-GCM (encryption)

Passwords used to connect to target databases (Servers.SqlPassword) are stored at rest with AES-256-GCM authenticated encryption. GCM mode provides both confidentiality and integrity: each record includes a random nonce and an authentication tag, so tampered data cannot be decrypted. The application decrypts the password only briefly in memory when connecting to the target database.

3. QR package zip passwords- AES-256-GCM (encryption)

The passwords generated for the password-protected zip files of QueryResult (QR) packages are stored using exactly the same principle as server passwords, encrypted with AES-256-GCM. This means all reversible secrets use a single, authenticated cryptographic method, improving both security and clarity of the system.

Audit trail integrity- HMAC-SHA256

Audit records are protected with a hash chain, and the current version uses HMAC-SHA256. The key is not stored in the database and comes only from the application configuration, so that even someone with database access alone cannot recompute the audit chain and silently alter history.

Key management

All encryption keys are 256-bit (32 bytes, base64) and are consolidated under the Security section of the application configuration: Security:ServerPasswordKey (server passwords), Security:QrZipPasswordKey (QR zip passwords), and Audit:HmacKey (audit trail). In production, keys are supplied via user-secrets or environment variables; they are never embedded in source code or the database.

Summary: user passwords use a one-way hash (BCrypt); server and QR zip passwords use two-way encryption (AES-256-GCM); the audit trail uses HMAC-SHA256.

8. Contact

- Web: www.sqlchangeguard.com
- Email: onur.egilmez@sqlchangeguard.com | info@sqlchangeguard.com
- Phone: +90 505 621 29 02
- LinkedIn: linkedin.com/company/sqlchangeguard

Sensitive Data Masking: Validated Detection and Server Level Regime

Masking works through two independent mechanisms that cover each other's blind spots.

The first is the classification catalog. It looks at the column name and never at the data. For example, every column whose name contains TCKN is masked. This mechanism also covers numeric columns, which is the only protection in systems that keep identity numbers as numbers. Every rule carries a label shown to the auditor, such as identity, contact, health, education or payment card. A ready made starter set covering finance, retail, healthcare and education is installed with the product.

The second is data patterns. These ignore the column name and inspect cell contents. The important part is that a pattern can be bound to a validation algorithm: a checksum for the national identity number, Luhn for card numbers, mod-97 for IBANs. As a result an eleven digit order number is not masked for nothing, while a real identity number is. This removes unnecessary unmasked data requests.

Column preview. While a request is being saved, the system connects to the target server and asks which columns the query returns. The query is not executed and no row is read. This lets the requester and the approver see which columns will be masked before delivery. Even when the query uses an alias, the source column is identified and masking cannot be bypassed.

Server level regime. Because masking is a property of the data, the setting lives on the server definition. There are two regimes: full masking and content only. The content only regime is intended for environments that hold synthetic data. It does not mask columns by name, but if real data is loaded onto that server the validated patterns take effect and masking returns automatically. This regime cannot be selected without a written reason, and a change is written to the audit trail as a separate event. When a request targets several servers the decisions are made per server: the result from a production server can be masked while the result from a server holding synthetic data stays open.

Evidence. For every delivery the system records, column by column, why a column was masked: classification catalog, data pattern, approved unmasked request or detection time budget exceeded. The record is per server, and the regime and exemption reason are frozen at delivery time, so the evidence stays intact even if the server definition is changed later. Masking can be switched off with an organisation wide parameter, but even then every delivery is recorded and marked with a red verdict in the evidence dossier.

Learning loop. Columns caught by a pattern but missing from the catalog collect as candidates; a database administrator turns them into permanent rules or ignores them. The candidate list holds no data values, only column, table and pattern names. This way the classification sharpens over time without anyone looking at production data.

Evidence: Which Rules Applied on the Day

Rules change over time. A setting may be strict today and have been relaxed six months ago. The question asked during an audit is this: which rules applied to this request on the day it was processed? The system answers that question from a record frozen on that day, not from the current settings.

Rule set snapshot. Whenever a request is saved, the rule set in force at that moment is written to a single audit record: the identity and control flags of the target servers (rollback requirement, automatic execution, masking regime), the approval and segregation of duties rules, the sensitive data masking settings, the result file access rules, the list of SQL standards that were blocking on that day, and the licensed modules that were open. The record does not change when a rule changes later.

Settings at delivery time. A request may be saved on one date and executed on another. Because masking happens at execution time, the threshold, sample size, sampling mode and time budget in force at that moment are frozen into the delivery record as well. This makes it possible to answer with certainty why no column was masked in a given delivery.

Control changes are recorded as separate events. Changing a setting and relaxing a control are not the same thing. Relaxing a masking regime, removing a rollback requirement, enabling automatic execution, disabling a detection pattern, deleting a classification rule, adding an address to the recipient pool, stopping a maintenance task and changing licensed modules are each written under their own event name. Relaxing and tightening receive different names. The audit screen can search by event name, so which control was relaxed, when and by whom is visible in a single list.

Two independent record layers. The application audit trail holds events that carry business meaning and is visible on screen. A second layer at the database level works independently of the application, so a change made directly to the database with an administration tool is also captured. The two layers do not replace each other, one completes the other.

Visible in the evidence dossier. The request level evidence dossier lists the rule set in force on the day the request was processed in its own readable section. An auditor sees both what happened and under which rules it happened in a single document.

Release package scope. Release packages carry change requests only. Query result requests cannot be part of a package; a result file is only delivered through its own flow, which applies sensitive data masking and writes the audit record. This rule is enforced on the server side as well, independently of the user interface.

Four Eyes Principle for Settings Changes

A settings change that weakens a control should not be made by a single person. The system can put settings screen changes behind approval: the change is not applied immediately, it waits for approval, and the previous value stays in effect until it is approved.

How it works. A pending change is held in a separate record; the real settings table is not touched until approval is given. As a result the rest of the system keeps reading the previous value and a half applied setting never occurs. On approval the change is applied through the very same path as if

the user had saved it, so the licence limit, validation and concurrency checks all run again at that moment.

Approval authority. A settings change is approved by another user who holds the permission of that screen. No separate approver role is defined; the permissions already exist in the role definition. Nobody can approve or reject a change they requested themselves, and this rule is not optional. The approval list is filtered by permission: a person only sees approvals for the screens they are authorised for.

Scope. The feature is governed by one organisation wide parameter with three options: disabled, control settings only, all settings. The default is disabled. Under control settings only, server definitions, user records, the role permission matrix, approval policies, SQL standards, the critical object list, the sensitive column catalog, sensitive data patterns, approved recipient addresses, maintenance jobs and control parameters such as masking, approval and segregation of duties wait for approval, while operational settings do not slow the flow down. The feature itself is a control: turning it on is applied immediately, turning it off requires approval. Password reset and user unlock are deliberately out of scope: they are help desk actions and are written to the audit record.

On screen. A record with a pending change is marked on its settings screen and cannot be edited again; a notice at the top of the screen shows how many changes are waiting. After saving, the user is told clearly that the change was not applied but sent for approval. A separate approval screen shows the previous and the requested value side by side and highlights the fields that differ.

Records. Requesting, approving, rejecting and failing to apply an approved change are written as separate audit events. At approval the change also produces its own audit record, so who approved it and what the value became can be traced separately. Rejection requires a reason.

Deployment note. Because nobody can approve their own request, at least two users must hold each screen permission in scope. In practice this comes down to a single requirement: at least two system manager users must be defined.

The Auditor Role

What audit teams usually need is one thing: to see. Who did what, which approval it passed, which data went to whom. None of that requires administrator rights. SQL Change Guard answers this need with a dedicated role.

What the auditor can do. See every request, review audit records, use reports and the dashboard, produce and download the evidence dossier for a request, verify the integrity of the audit trail and view notification logs.

What the auditor cannot do. Raise a request, approve, execute, use the sandbox, manage any settings screen, or download a query result file.

The last item is deliberate. Query result data never enters the evidence package, so the auditor reaches the evidence without reaching production data. Auditing does not require seeing personal data, so this boundary does not weaken the audit; it narrows the organization data surface.

There is no adding or deleting evidence. The evidence dossier is produced from the request itself and no editable evidence store exists. If an auditor could add or delete evidence, the evidence would stop being evidence.

Segregation of duties. The auditor cannot be selected as an approver in a policy step, and the auditor role cannot be granted approve or execute permissions. Someone who audits must not become someone who approves, so this rule is not optional.

Deployment note. The product ships with a single administrator account and no ready made auditor account. Each organization defines its own auditor, so no installation carries an audit account with a known password.

Who Did What in Settings Approvals

While the four eyes rule is on, one person requests a settings change and another person holding the same permission approves it. Both names must be visible on the record; otherwise an audit gets a single answer to "who changed this setting" and that answer is incomplete.

The last change column. Every settings screen inside the four eyes scope now carries a new column. The top line shows who changed the record and when. The green badge below shows the approver. When the badge is present the top line carries a Changed by label; without it a single name is shown. Hovering the cell shows the creator, the person who changed the record and the approver on three lines.

Whose name goes on the record. When approval is given the change is applied and the name written on the record is the person who REQUESTED it. The approver is recorded on the settings approval entry and in the audit trail. A single row therefore shows both the requester and the approver.

Two separate audit entries are written. One is the change itself and carries the requester name. The other is the approval entry and carries the approver name, time and IP address. The IP address at approval time belongs to the approver, so it is not written next to the requester; one row carrying wrong information is enough to weaken an audit trail.

The refresh button. Because another user gives the approval, the list on screen can become stale at any moment. The refresh button at the top right of the settings page refreshes the data of the open tab. When a decision is made on the approvals screen, the related screens refresh by themselves.

The settings approvals tab. This tab is visible only to users holding a permission of a screen inside the four eyes scope. Users with view only permissions do not see the tab, because the filtered list would be empty for them anyway.

How Times Are Shown

Every time value in the system is stored in universal time (UTC). Local time is not a property of the data but a presentation choice. Two users in different cities therefore look at the same event and see the same instant; only the text on screen follows their own clock.

On screen. Dates and times are shown in the time zone of your browser. No separate setting is needed.

In documents. PDF, Excel and e-mail are produced on the server, where there is no browser, so the display time zone is read from the installation setting and written onto the document. A reader never has to guess what the time refers to. This choice is deliberate: if the time zone depended on

the browser of the person requesting the document, the evidence dossier for the same request would produce two different documents for two auditors.

In date filters. When you pick a date range, the day is your local day. The start day is included from 00.00 and the end day is included until the end of that day.

In the audit trail. Audit records are shown with seconds, so the order of two events inside the same minute can be read.