SQL Change Guard

Product Overview

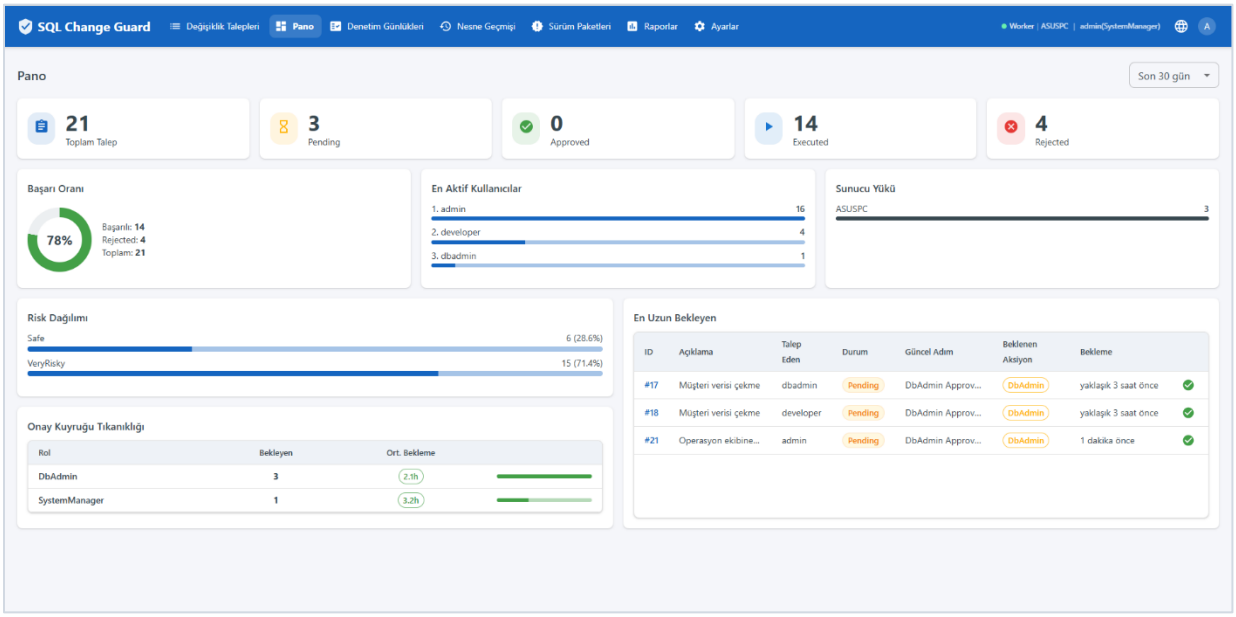
Database Operations Governance Platform

SQL Server | PostgreSQL | Oracle

1. Executive Summary

SQL Change Guard is a Database Operations Governance solution that manages database changes, production data access and compliance in a single platform. It is not merely a schema deployment tool; it makes the full lifecycle, from change request to deployment and from data request to delivery, policy-based and auditable.

It offers the same governance model for SQL Server, PostgreSQL and Oracle. It runs fully on-premise and requires no internet connection.



Dashboard

2. The Problem

In most organizations, database operations run in three separate and manual places: change deployment, production data requests and audit. This fragmentation creates risk:

- Unapproved or incorrect changes reaching production
- Critical mistakes such as UPDATE/DELETE without a WHERE clause slipping through
- No visibility into whether data leaving production contains personal information or was masked
- Inability to prove who approved, who executed and who received the data during an audit

3. Three Governance Layers

3.1 Database Change Governance

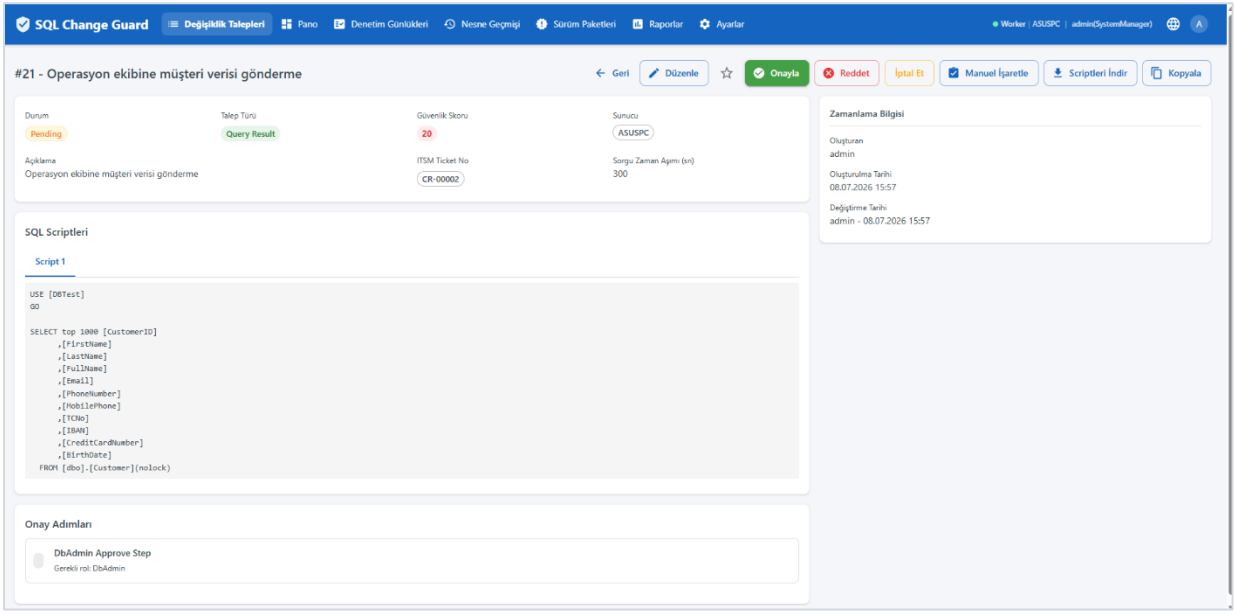
- Change Request: end to end lifecycle of DDL/DML/DCL changes

- Role-based, multi-step, policy-driven approval workflow
- Pre-deployment validation with 84+ SQL standard rules
- Automatic 0-100 risk score per script (Critical / High / Medium / Low)
- Sandbox validation: run and roll back on a real environment before going live
- Release packages: ordered, dependency-aware deployment of multiple changes
- Automated rollback generation (with Oracle implicit-commit atomicity guarantee)
- Version history and object history (who, when, which request, which script)

3.2 Production Query Governance

This is the layer that sets SQL Change Guard apart from migration-based tools. Requesting data from production becomes central and auditable.

- Query (SELECT) request and policy-based approval workflow
- Critical table and column detection
- Automatic masking for sensitive data (validated regex detection of national ID, card, IBAN, phone; full mask or partial mask)
- Separate security approval for unmasked data requests
- Verification of first-time email addresses
- Encrypted ZIP delivery of results, download reason and download tracking
- Multi-server queries and Excel export

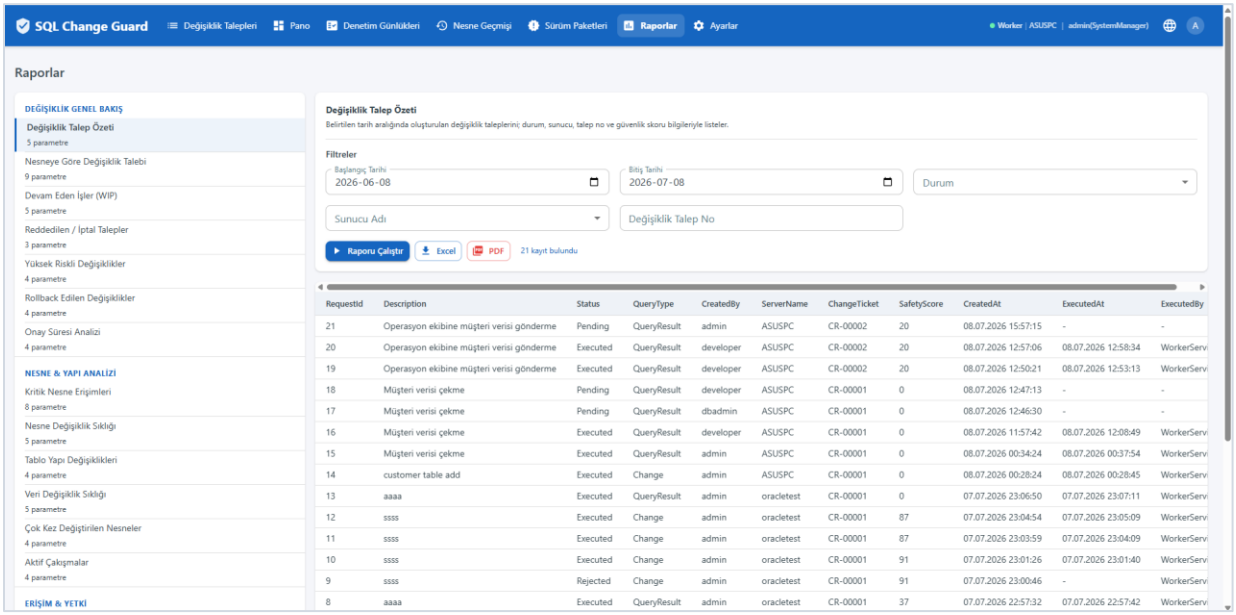


Query Result with automatic masking

3.3 Compliance and Audit

- Immutable audit records protected by a hash chain

- Approval, download and execution history
- Evidence packages for audits
- Per-request evidence file: all evidence of a single request (identity, segregation of duties, approvals, script fingerprint, risk score, execution and audit trail) is downloaded as one sealed file (PDF and JSON) with a single click
- Evidence integrity verification: audit records are verified as unchanged since they were written, by comparison against a separate immutable copy
- 30 built-in reports: DML/DDL/DCL audit, critical object access, sensitive column touches, privilege movements, privileged account operations, QR masking log and more
- Alignment with KVKK, ISO 27001, COBIT and ITIL



Reports - 30 built-in reports, Excel/PDF export

4. Policy Engine

No manual decisions; everything is policy driven. An example production query flow:

- A production query request arrives
- If it touches a critical table, DB Admin approval is required
- If it contains sensitive columns, data is masked automatically
- If unmasked data is requested, additional security approval applies
- The result is delivered encrypted and every step is audited

5. Modules

Module	Function
Change Management	**End to end lifecycle of DDL/DML/DCL requests**
Automatic Validation	**Pre-deployment checks with 84+ SQL standard rules**
Risk Scoring	**0-100 risk score and classification per script**
Sandbox Test	**Safe test on a real environment with run and roll back**
Rollback	**Automated rollback, Oracle implicit-commit awareness**
Release Package	**Dependency management and ordered deployment**
Production Query Governance	**Production SELECT requests, policy and masking**
Sensitive Data Masking	**Regex-based detection and column-level masking**
Policy Engine	**Multi-level approval by critical table and sensitive column**
Reporting	**30 built-in reports, Excel/PDF export**

Object History	**Full change history of any object**
Server Management	**Manage multiple SS/PG/Oracle servers in one place**

6. Roles and Authorization

All actions are authorized on the backend; UI control alone is not sufficient. Roles: Viewer, Requester, Request Manager, DB Admin, Change Manager, System Manager. Separation of duties is supported.

7. Why SQL Change Guard?

- Change, production data access and compliance in one platform
- The same governance model for SQL Server, PostgreSQL and Oracle
- Set apart from migration tools by its production data access and masking layer
- Fully on-premise; no data leaves the corporate network
- Audit infrastructure ready for banking and enterprise compliance

One Platform. Database Change Governance + Production Query Governance + Compliance and Audit = Secure Database Operations.

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.