SQL Change Guard

Technical Specification

Architecture, Security, Integration and Requirements

SQL Server | PostgreSQL | Oracle

1. Architecture Overview

SQL Change Guard has a layered architecture and runs fully on-premise.

- Presentation layer: React-based web interface
- Application layer: REST API (.NET 10)
- Background service: Windows Service for scheduled jobs (Quartz scheduler)
- SQL analysis layer: a dedicated parser per database (ANTLR4-based PL/SQL parser for Oracle)
- Policy Engine and Audit Engine
- Application database: SQL Server (management data); a separate Immutable database for tamper-evident audit
- Target databases: SQL Server, PostgreSQL, Oracle

2. Database Independence

Vendor dependencies are isolated. Each target database has its own parser, splitter, executor and rollback generator; the same governance model runs across all three.

- SQL Server: T-SQL parser (ScriptDom-based), GO batch separator
- PostgreSQL: dollar-quote aware ; separator
- Oracle: PL/SQL block protection, standalone / and ; separators; DDL/DCL/TRUNCATE implicit-commit atomicity guarantee

3. Security

- Authentication: JWT-based; LDAP / Active Directory integration
- Multi-factor authentication (MFA) support
- Role-based, backend-enforced authorization (endpoint, service, screen, action, worker)
- Server passwords stored at rest encrypted with AES-GCM
- Query results delivered as encrypted ZIP
- By default the zip password is not mailed; it is shown in the portal after a written reason and every view is audited
- Immutable audit: records protected by a hash chain
- Separation of duties

4. Integration

- LDAP / Active Directory (users and identity)
- SMTP (notification and approval emails)

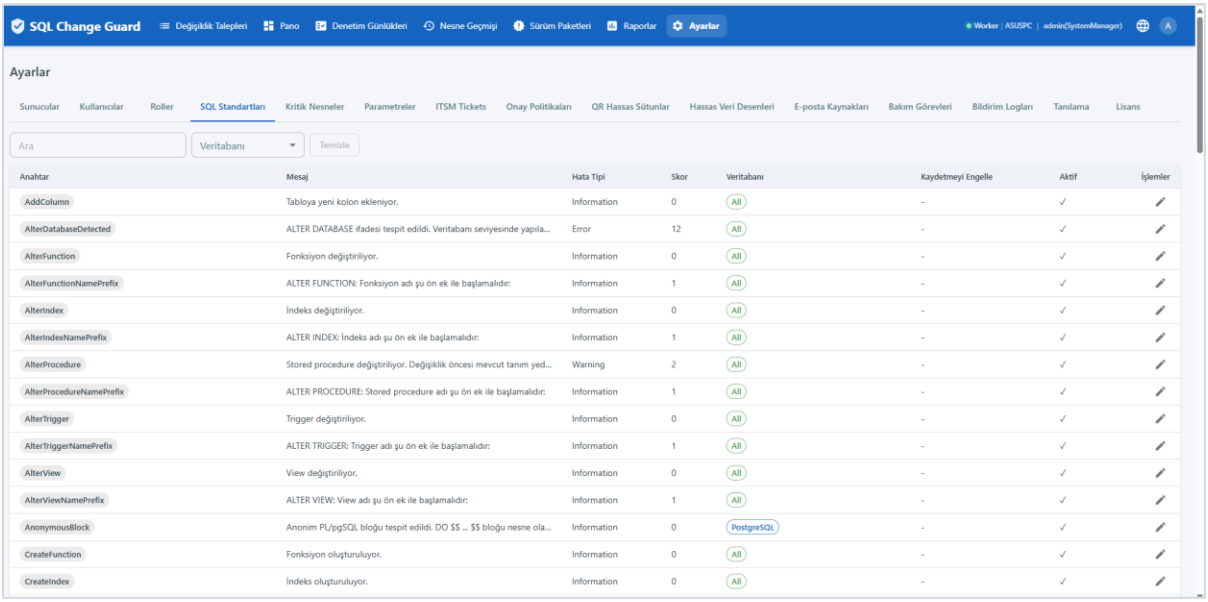
- REST API (consumption from external systems)
- ITSM ticket field (link a ticket to a change request; format validated by regex)

5. Configuration and Rules

- Configuration is managed via appsettings and the Parameters table; no hardcoded values
- 84+ SQL standard rules can be enabled/disabled from the UI and filtered by database type
- Approval policies defined by server, environment and query type
- Sensitive data patterns (regex) and critical object list are manageable

- In critical object definitions the database and schema fields can be left empty: an empty database means all databases and an empty schema means all schemas. The object type is always required and must match exactly. The same rule applies to all three database types.

- All dates stored in UTC; local time shown to the user

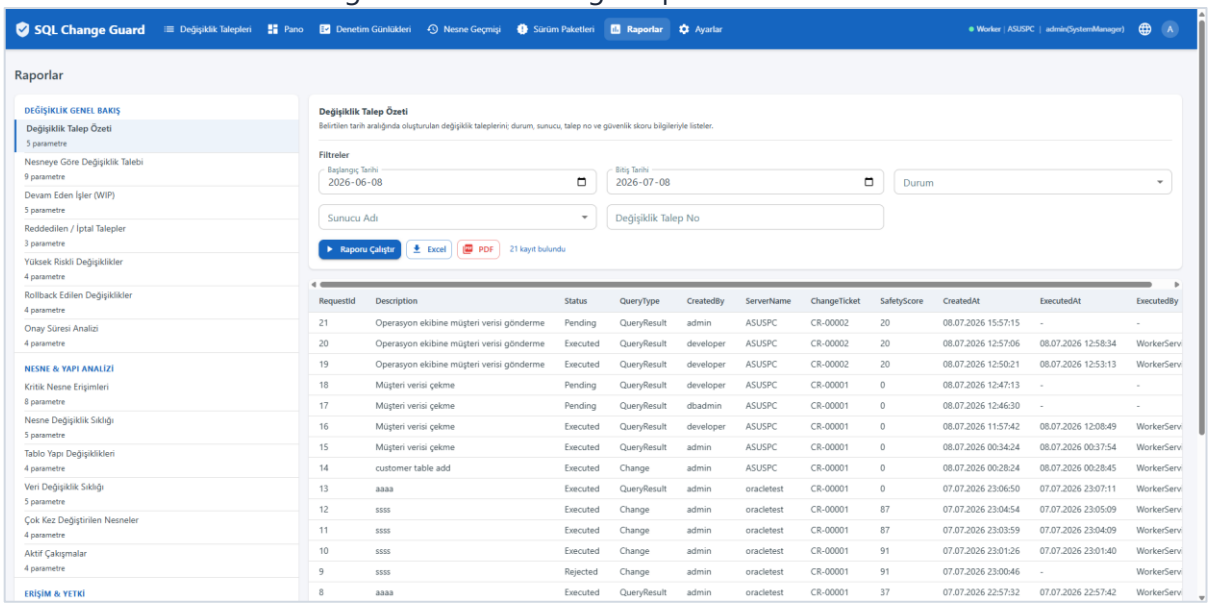


SQL Standard Rules - enable/disable and filter by database type

6. Audit and Reporting

30 built-in reports export to Excel and PDF; dynamic report definitions are supported. Report access has two layers: the screen permission and a per report role restriction. The restriction is enforced on

the server both when listing and when running a report.



Opening a role restricted report is written to the audit trail as well: who opened which report, with which filters, and how many rows were returned. No row data is stored in the record. Unrestricted reports are not logged, so the audit trail stays readable.

Adding, updating and deleting an ITSM ticket is written to the audit trail too. Ticket data is not a control setting, so it does not go through dual control approval; the audit record is the only evidence.

In the evidence dossier the target is written as server name, database engine and database name together. The database name comes from the default database of the server definition; when the script contains a USE statement it overrides that at run time, so it is shown on its own line as well.

A query result request can target more than one server; server names are separated by semicolons. Server resolution honours that separator: database type and name are resolved per server, automatic execution is only enabled when every target server allows it, and when one of the servers no longer exists the machine does not execute.

Reports - 30 built-in reports in five groups

- Change Governance: request summary, request by object, work in progress, rejected and cancelled, high risk changes, rollback changes, approval and execution duration, execution activity, approval step detail, sandbox test results, control exceptions
- Object & Structure: critical object access, object change frequency, table structure changes, data modification frequency, multi release objects, active conflicts
- Access & Authorization: privileged account operations, user DDL activity, user permission matrix, user access changes
- Data & Privacy: query result activity, sensitive column touches, masking activity, sensitive column catalog, masking exempt servers, manual data modifications, data access and export log
- Configuration & Compliance: configuration changes, settings approval trail

7. System Requirements

Component	Requirement
Application database	**SQL Server 2016 or later**
Server environment	**Windows Server (IIS/Kestrel + Windows Service)**
Runtime	**.NET 10**
Target databases	**SQL Server, PostgreSQL, Oracle**
Client	**Modern web browser (Chrome, Edge, Firefox)**
Network	**Internal; no internet connection required**

8. Installation

The system is designed to be stood up in an enterprise environment within 30 minutes. The schema is applied automatically via numbered, checksum-protected migration scripts. Environment dependencies are minimal.

9. Contact

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

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.