# Challenges With Managing Permissions Using Database Roles

Databases enable RBAC through the database roles which are used for administering user access to schemas, tables and views. However, as organizations become more data driven and often have hundreds of users and applications accessing data stored in databases, data warehouses, and data lakes, governing the security of sensitive datasets becomes increasingly unwieldy in practice.

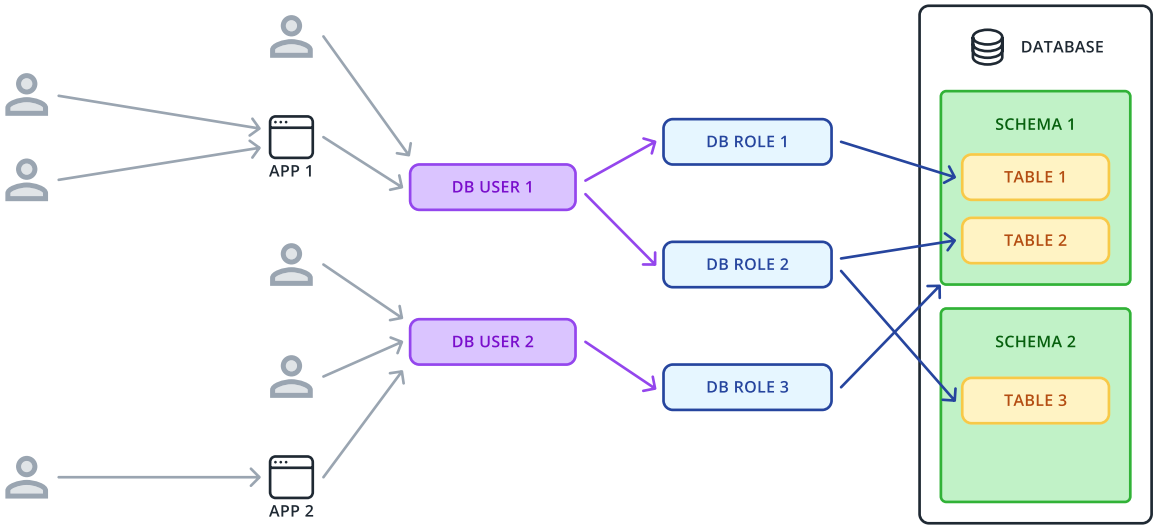
This white paper below describes the challenges of administering data security governance policies using database roles, and an alternative approach that addresses those challenges.

## The Database RBAC Model

The database RBAC model comprises of the following:

- **A securable object:** This generally includes databases, schemas, tables and views.
- **A role authorized to access the securable object:** These are defined within the database. New roles can be created by administrators at any time.
- **Privileges assigned to the role:** These privileges enumerate the actions that the role is authorized for. Examples would include the ability to create new roles, delete tables, create new schemas, read from specific tables, etc.
- **Database users:** These are accounts that are created within the database. Individual users may be using distinct user accounts or shared accounts (these latter are generally referred to as service accounts).

- **Roles a database user is authorized to assume:** Each user must assume a role to perform any action within the database. These mappings of users to roles are maintained within the database.




In this model, the database role is the key artifact that is used to manage permissions for which user is allowed to do what within a database. All privileges are associated with a role and users must assume the appropriate role to perform the task they need.

## Limitations of Using Database Roles for Data Security Governance

Using database roles to manage permission proves to be very cumbersome in practice, both when trying to tame an existing database that has been in use for some time, as well as when starting from scratch with a new workload and database. This is generally because of the following reasons:

- **Evolving needs of the data team:** Managing permissions for roles mandates mapping groups of users with the same data access needs to the same set of roles. These mappings get out of date quickly as people join and leave, and new datasets and tasks show up. Historically, DBAs are skilled at managing data and systems, but are not custodians of access. Additionally, these database roles don't inherit from typical IAM entitlements which makes keeping permissions in sync with users' job functions infeasible.

- **Coarse granularity of control:** Enforcing policies using roles is hard because they are often too coarse-grained — roles are often granted at the schema, table or view level, and are cumbersome to maintain at column levels. Security teams, on the other hand, often care about column level and row level access to data, which creates a gap between their needs and the capabilities of database RBAC.
- **Not suitable as a security policy framework:** In RBAC, each role grants certain privileges to the user and the overall set of privileges for a user is the union of privileges from their respective roles. On the other hand, a security policy framework needs to be based on a "least privilege" model where each policy can specify the set of users who can access the data and with what constraints. Access must be allowed only if none of the policies governing access to the data are violated. For example, if a PCI policy and a CCPA policy specify different sets of users authorized to access certain credit card numbers, only users authorized by both policies should be allowed to access this data.
- **Lack of semantic association with sensitive datasets:** A common challenge that teams face is that looking at GRANT and REVOKE statements doesn't inform security teams whether or not access is being given to any sensitive data. Generally, teams invest in data discovery and classification tools to tag sensitive datasets, but there is no association between those tags and database roles.
- **Lack of transparency:** When it comes to security and governance policies, security teams strive to prioritize transparency and collaboration to build a culture of trust. Database roles unfortunately are not helpful in this regard, because of limited tooling to report on (often complex) inheritance hierarchies and database-specific logic of how privileges are inherited.
- **Shadow access:** Many users access databases using an application (BI tool, notebook, etc) which accesses data using a service account / single role, generally with a large number of privileges. This makes it impossible to manage those users differently from each other. Addressing this requires repetitive permissions management across multiple tools and applications, which is challenging.

- **Lack of separation of duties:** Most security teams lack the know-how, tools and bandwidth to administer database roles across a large team of users. As such, they have to resort to asking the data teams to essentially manage roles and privileges on their own, resulting in a lack of separation of duties, a cornerstone of most security and governance programs.

| CHALLENGE | CONSEQUENCE |
|---|---|
| Users accessing using tools and apps | Shadow access which violates policies |
| Fluid user-to-role assignment | Over-permissions privileges and dormant roles |
| Roles and privileges managed by data teams | Lack of separation of duties and assurance |
| Lack of clear reporting of roles and privileges | Increased risk and complex remediation |
| Permissions granted on tables and views | Difficult to provide row and column level policies |
| Permissions not based on a least privilege model | Not possible to create multiple independent policies governing access to the same data |
| Lack of visibility into where sensitive data resides | Difficult to timely update grants based on creation or deletion of sensitive data |

**What are companies doing today?**

The conventional approach to managing permissions within databases comes down to the following:

- **Inventory creation:** Build an inventory of all users, roles within the database, all privileges associated with each role, and all schemas, tables or views with sensitive data in them.
- **Analyze roles:** Understand for each roles what privileges have not been exercised in the last 90 or 180 days.
- **Analyze users:** Build a complete map of which user can access what sensitive datasets, using a combination of the roles that they are assigned.

Armed with this analysis, teams then routinely prune excessive privileges, put to rest dormant roles, and refresh what users are mapped to which roles. These tasks are typically performed every 3-6 months to take care of drifts in privileges and configurations as the needs of the data team evolve, new data models are acquired, and users join and leave the teams.

This process is often very tedious and breaks down at scale, resulting in risks and inefficiencies. There are several tools, such as DSPM, which attempt to smoothen these workflows, but ultimately result in only minor improvements on status quo (for more details, **read our blog**).

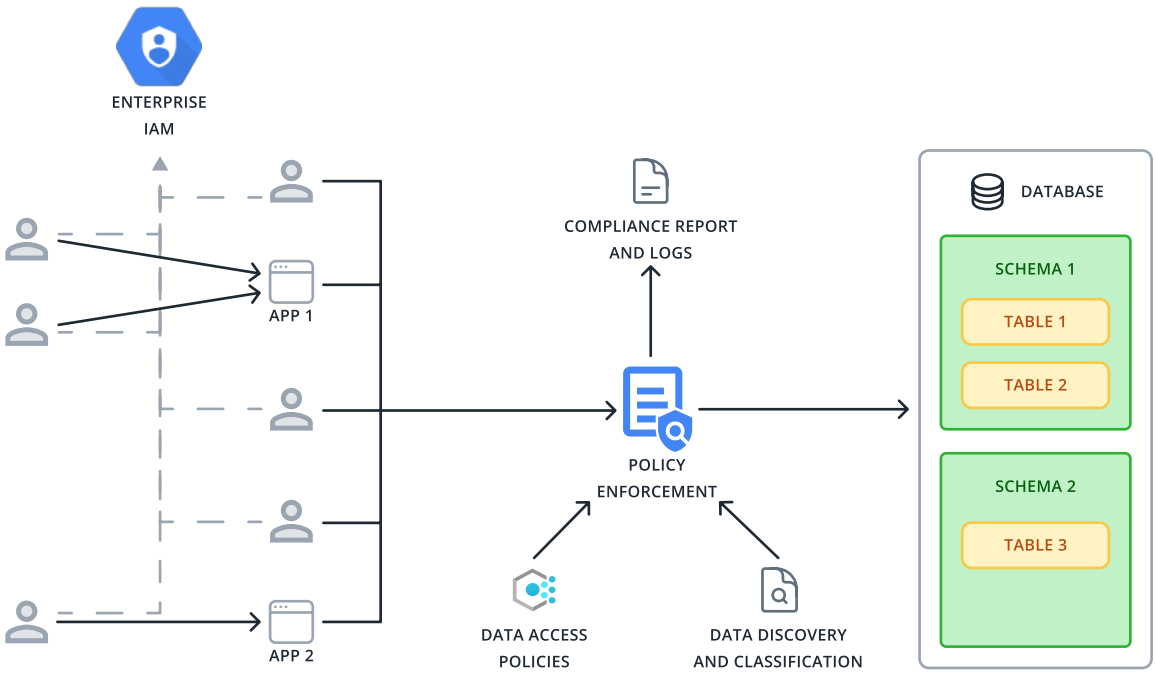
## The Solution: Data Security Contracts

A data security contract is a shared understanding between data teams and security teams that documents who is authorized to access which data and with what constraints. In the context of data engineering, the term data may refer to any set of databases, schemas, tables, views, columns or rows. This contract must be enforceable, typically through a set of policies which can be easily reviewed by both data and security teams.  Additionally, the contract is associated with an ever evolving index of tags which results from a continuous data discovery and classification process.

## Characteristics of a Data Security Contract

- **Decoupled from database roles to streamline management:** All policies are specified on user identities (and associated entitlements and SSO groups) instead of database roles to centralize management and enforce separation of duties.
- **Integrated with IAM to align with corporate standards:** Permissions in a contract are integrated with entitlements defined in IAM systems which ensures that security policies align with broader organizational standards, enhancing security and governance.
- **Policy as Code to promote transparency and automation:** By expressing policies as code and leveraging existing IAM systems, it's possible to automate and standardize entitlements, improving efficiency and reducing the chance of human error.
- **Enforced on direct and application users:** Policies must be enforced uniformly on both direct and application users - so if someone accesses a dataset through a console or a BI tool, the policy is applied consistently. This eliminates the challenges and risks associated with shadow access.
- **Coupled with a continuous data discovery & classification process:** For policies to be effective, they must operate on all sensitive data in scope. As such they must be specified on data tags that get updated continuously as the schema and datasets evolve, generated by an ongoing data discovery and classification process.
- **Based on the principles of least privilege:** Typically, multiple teams will be involved in specifying data security and governance policies, eventually resulting in overlap and conflicts. Therefore, the most restrictive constraints specified for accessing a certain data item among all policies must be enforced.

A data access model based on a data security contract is illustrated below.

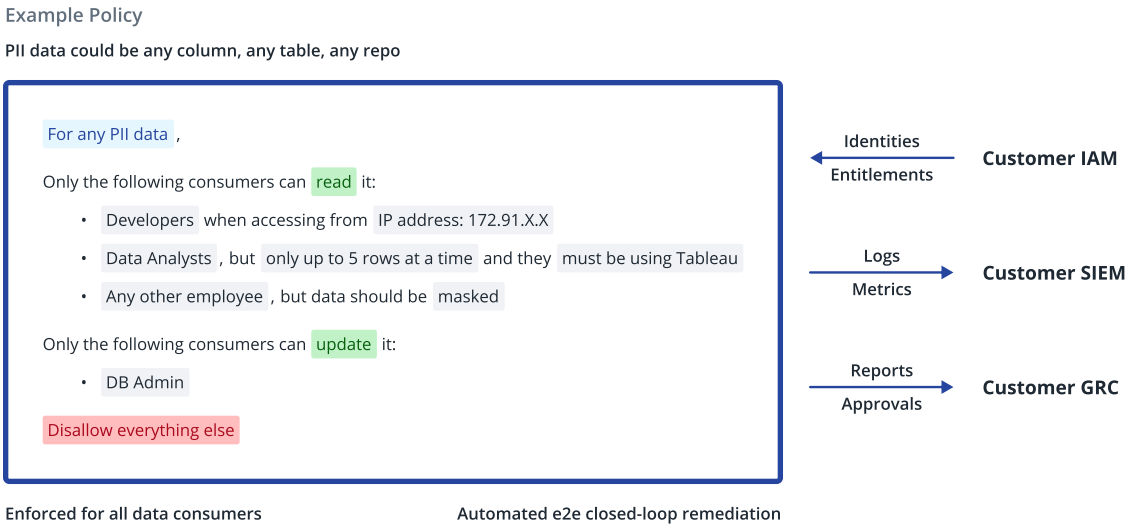



## Benefits of a Data Security Contract

- Users access data using their corporate identities and credentials. This means that it is possible to centrally grant and revoke privileges (for example, when users join, leave, or change roles in the organization).
- New locations of sensitive data are continuously discovered and protected as applications evolve.
- Uniform policies based on user entitlements are applied to the sensitive data regardless of where it is stored.
- Security teams can create and update data access controls and policies without needing to know database specific commands and concepts. The data teams can focus on their core mandate instead of being forced to implement and update security policies on behalf of the security team.

## How Cyral Provides An Enforceable, Easy to Manage Data Security Contract

Cyral helps companies address this challenge by relying on user identities and entitlements as defined in the organization's IAM services for defining policies. The below example illustrates how Cyral can help define highly granular permissions on datasets within a single or multiple databases.




- The policy is defined on a dataset PII - what specific tables, columns, views, etc it refers to is completely abstracted away from the user. Cyral performs continuous discovery & classification to keep this tag up-to-date.
- Cyral can be leveraged as a gateway for all privileged operations - including reads, updates, deletes, etc.
- Policies specify what operation is permitted by which user, and their information and entitlements are pulled from customers' IAM services - this eliminates the need to manually update policies as the entitlements evolve.
- The operations themselves - masking, row limits, network constraints, etc - are made consistent across various databases, even when they are not supported within the database themselves.
- All orchestration can be easily linked to customers' existing tools for logging, ITSM, etc.

## About

Cyral provides controls for privacy, compliance, governance and protection thereby reducing risk, complexity, and cost for managing structured data. The Cyral platform discovers data, unifies access controls for users and applications, enables fine-grained authorization policies and provides complete monitoring and reporting. This comprehensive coverage enables risk-based governance, limits the blast radius of data-related incidents and reduces overhead and costs. Cyral's technology allows customers to implement data security controls using their existing, centralized entitlements, thereby simplifying administration and automating remediation. Customers use Cyral to accomplish least privilege, data minimization, spillage prevention and Zero Trust.

For more information, visit **www.cyral.com** or follow **@CyralInc** on X.