# deploy.SetRolePermissions – Technical Documentation

Version: 2.0  
Date: 2025-08-01  
Page: 1 of 2

## Summary

Grants role-based permissions across schemas, tables, and functions. Supports idempotent creation, dynamic role GRANTs, and default privilege setup.

## Internal Logic

1. Loop over each role in `p\_role\_names`:  
 - CREATE ROLE if it doesn't exist  
 - Log action in `info.account\_log\_history`  
  
2. If `p\_users\_to\_assign` is set:  
 - GRANT each role to each user  
  
3. For each schema in `p\_apply\_schema\_privs\_to\_roles`:  
 - If `p\_revoke\_all\_first`, REVOKE ALL PRIVILEGES on objects  
 - GRANT listed `p\_table\_privileges` on TABLES  
 - GRANT `p\_function\_privileges` on FUNCTIONS  
 - If default flags are true, alter default privileges using ALTER DEFAULT PRIVILEGES

## Dry-Run vs Execute Mode

- All SQL statements are logged either way  
- `p\_execute\_flag = FALSE` logs intent only  
- `TRUE` runs actual CREATE, GRANT, ALTER, etc.

## Security Considerations

- Assumes caller has CREATE ROLE and schema/table/function GRANT access  
- Consider wrapping in `SECURITY DEFINER` for automation  
- Logs to `info.account\_log\_history` for traceability

## Limitations

- Does not verify user/role existence before GRANT  
- Schema must exist in `information\_schema.schemata`  
- Only supports TABLE and FUNCTION object types

## References

- https://www.postgresql.org/docs/current/sql-createrole.html  
- https://www.postgresql.org/docs/current/sql-grant.html  
- https://www.postgresql.org/docs/current/sql-alterdefaultprivileges.html

## TL;DR

Manages role creation and privilege grants with dry-run support. Used for structured RBAC setup.