CREATE ROLE name; explained

The `CREATE ROLE` statement is used in various database management systems (DBMS) to create a new role, which is essentially a named collection of privileges that can be assigned to users or other roles. Below is a detailed explanation of the syntax, usage, and variations of the `CREATE ROLE` statement across different systems.

General Syntax

The general syntax for creating a role is as follows:

```
CREATE ROLE [IF NOT EXISTS] role_name [WITH ADMIN {CURRENT_USER | CURRENT_ROLE | user | r
```

Key Components

- **role_name**: This is the name of the role you want to create. It must be unique within the database system and typically has a maximum length (e.g., 128 characters in MariaDB and MySQL)[1] [2].
- **IF NOT EXISTS**: This optional clause prevents errors if the role already exists. Instead of an error, a warning is issued[2].
- **WITH ADMIN**: This clause specifies which user or role has administrative privileges over the newly created role, allowing them to grant this role to others. If omitted, the current user is granted admin rights by default[1].

Behavior and Privileges

- To execute the `CREATE ROLE` command, the user must have specific privileges, such as `CREATE USER` or `CREATE ROLE`[2] [3].
- When a role is created, it initially has no privileges and is locked (i.e., it cannot be used until granted to users)[2].
- The command either succeeds for all specified roles or fails entirely if any error occurs, rolling back any changes made during that command execution[2].

Variations Across Different DBMS

MySQL/MariaDB

- In MySQL and MariaDB, roles can be created without any associated privileges initially. The syntax allows for multiple roles to be created at once. For example:
  ```
  CREATE ROLE 'admin', 'developer';
  ```
- If you attempt to create a role that already exists without using `IF NOT EXISTS`, an error will occur[2].

PostgreSQL

- In PostgreSQL, roles can have attributes such as `LOGIN`, `SUPERUSER`, and other permissions defined at creation. For example:
  ```
  CREATE ROLE bob WITH LOGIN;
  ```
- This command creates a role that can log in to the database[3] [4].

Oracle

- In Oracle databases, the `CREATE ROLE` statement allows for the specification of various attributes and requires adherence to naming conventions. For instance:
  ```
  CREATE ROLE my_role;
  ```
- Role names must follow specific naming rules and cannot begin with certain prefixes in container databases (CDB)[5].

SQL Server

- In SQL Server, roles are created at the database level and can be configured with permissions afterward using `GRANT`, `DENY`, and `REVOKE` commands[6].

Examples

Here are some practical examples of using the `CREATE ROLE` command:

1. **Creating a Simple Role**:
   ```
   CREATE ROLE developer;
   ```
2. **Creating a Role with Admin Privileges**:
   ```
   CREATE ROLE analyst WITH ADMIN current_user;
   ```
3. **Using IF NOT EXISTS**:
   ```
   CREATE ROLE IF NOT EXISTS guest;
   ```

4. **Creating Multiple Roles**:

```
CREATE ROLE 'admin', 'editor', 'viewer';
```

In summary, the `CREATE ROLE` statement is essential for managing user privileges in database systems, allowing for organized access control and security management. Each DBMS may have specific nuances in implementation, but the core functionality remains consistent across platforms.

⁂

1. https://mariadb.com/kb/en/create-role/
2. https://dev.mysql.com/doc/refman/8.4/en/create-role.html
3. https://neon.tech/postgresql/postgresql-administration/postgresql-roles
4. https://www.postgresql.org/docs/current/sql-createrole.html
5. https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/CREATE-ROLE.html
6. https://learn.microsoft.com/en-us/sql/t-sql/statements/create-role-transact-sql?view=sql-server-ver16