Guide to Creating Temporary and Restricted Access for Developers in PostgreSQL

- راهنمای ایجاد دسترسی موقت و محدود برای توسعه‌دهندگان در پستگرس‌کیوال

راهنمای ایجاد دسترسی موقت و محدود برای توسعه‌دهندگان در پستگرس‌کیوال

هدف این مستند، ایجاد دسترسی کنترل‌شده، موقت و قابل مدیریت برای توسعه‌دهندگان روی پایگاه داده است؛ به‌طوری‌که هر توسعه‌دهنده فقط برای مدت مشخص و با سطح دسترسی مورد‌نیاز بتواند به پایگاه داده متصل شود.
در این ساختار بهتر است مجوزها مستقیماً به هر کاربر داده نشوند. ابتدا یک نقش برای سطح دسترسی ایجاد می‌شود و سپس کاربران موقت عضو آن نقش می‌شوند.

اتصال به پستگرس‌کیوال

ابتدا وارد پاد مربوط به پایگاه داده شوید:

```
psql -U postgres
```

ابتدا نقش مشترک را یک بار بساز:

```
CREATE ROLE developers_rw_all NOLOGIN;
```

به فضای پاد برمی‌گردیم:

```
\q
```

ایجاد دسترسی

```
for db in $(psql -U postgres -Atc "
SELECT datname
FROM pg_database
WHERE datallowconn = true
  AND datistemplate = false;
```

```
"); do

    echo "Configuring database: $db"

    psql -U postgres -d "$db" <<'SQL'

SELECT format(
    'GRANT USAGE ON SCHEMA %I TO developers_rw_all;',
    nspname
)
FROM pg_namespace
WHERE nspname <> 'information_schema'
  AND nspname NOT LIKE 'pg_%'
\gexec

SELECT format(
    'GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA %I TO developers_rw_all;',
    nspname
)
FROM pg_namespace
WHERE nspname <> 'information_schema'
  AND nspname NOT LIKE 'pg_%'
\gexec

SELECT format(
    'GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA %I TO developers_rw_all;',
    nspname
)
FROM pg_namespace
WHERE nspname <> 'information_schema'
  AND nspname NOT LIKE 'pg_%'
\gexec

SQL

    psql -U postgres -d postgres -c \
    "GRANT CONNECT ON DATABASE \"$db\" TO developers_rw_all;"

done
```

ایجاد کاربر موقت برای توسعه‌دهندگان

برای هر توسعه‌دهنده یک کاربر جداگانه ایجاد می‌شود.

```
CREATE ROLE dev_alice
LOGIN
PASSWORD 'StrongTemporaryPassword123!'
VALID UNTIL '2026-09-29 15:00:00+03:30';
```

تمدید زمان دسترسی

```
ALTER ROLE dev_alice
VALID UNTIL '2026-09-29 18:00:00+03:30';
```

ایجاد کاربر عادی برای توسعه‌دهندگان

برای هر توسعه‌دهنده یک کاربر جداگانه ایجاد می‌شود.

```
CREATE ROLE dev_alice
LOGIN
PASSWORD 'StrongTemporaryPassword123!';
```

اضافه کردن کاربر به نقش دسترسی

پس از ایجاد کاربر، آن را عضو نقش اصلی می‌کنیم:

```
GRANT developers_rw_all TO dev_alice;
```

بررسی عضویت کاربر در نقش‌ها

برای مشاهده اطلاعات کلی کاربر:

```
\du dev_alice
```

برای مشاهده اینکه کاربر عضو چه نقش‌هایی است:

```
SELECT
    member_role.rolname AS user_name,
    group_role.rolname AS member_of
FROM pg_auth_members am
```

```
JOIN pg_roles member_role
    ON member_role.oid = am.member
JOIN pg_roles group_role
    ON group_role.oid = am.roleid
WHERE member_role.rolname = 'dev_alice';
```

بررسی زمان انقضای کاربر

برای مشاهده زمان پایان اعتبار گذرواژه:

```
SELECT
    rolname,
    rolcanlogin,
    rolvaliduntil
FROM pg_roles
WHERE rolname = 'dev_alice';
```

مشاهده تمام کاربران دارای قابلیت ورود

```
SELECT
    rolname,
    rolvaliduntil
FROM pg_roles
WHERE rolcanlogin = true
ORDER BY rolname;
```

مشاهده مجوزهای نقش

```
SELECT
    grantee,
    table_schema,
    table_name,
    privilege_type
FROM information_schema.role_table_grants
WHERE grantee = 'developers_rw_all'
ORDER BY table_schema, table_name, privilege_type;
```

بررسی دسترسی واقعی یک کاربر

برای بررسی دسترسی واقعی کاربر روی پایگاه داده، شما و یک جدول مشخص:

```
SELECT
    has_database_privilege(
        'dev_alice',
        'devops_test',
        'CONNECT'
    ) AS can_connect,

    has_schema_privilege(
        'dev_alice',
        'public',
        'USAGE'
    ) AS can_use_schema,

    has_table_privilege(
        'dev_alice',
        'public.employees',
        'SELECT'
    ) AS can_select,

    has_table_privilege(
        'dev_alice',
        'public.employees',
        'INSERT'
    ) AS can_insert,

    has_table_privilege(
        'dev_alice',
        'public.employees',
        'UPDATE'
    ) AS can_update,

    has_table_privilege(
        'dev_alice',
        'public.employees',
        'DELETE'
    ) AS can_delete;
```

قطع فوری دسترسی

اگر لازم باشد دسترسی کاربر قبل از زمان تعیین‌شده قطع شود:

```
ALTER ROLE dev_alice NOLOGIN;
```

اگر کاربر از قبل نشست فعال داشته باشد، برای قطع نشست‌های فعال:

```
SELECT pg_terminate_backend(pid)
FROM pg_stat_activity
WHERE usename = 'dev_alice';
```

در صورت نیاز به فعال کردن دوباره ورود:

```
ALTER ROLE dev_alice LOGIN;
```

حذف کاربر پس از پایان دسترسی

اگر پس از پایان کار دیگر نیازی به نگهداری کاربر نباشد، ابتدا عضویت او از نقش اصلی حذف می‌شود:

```
REVOKE developers_rw_all FROM dev_alice;
```

سپس کاربر حذف می‌شود:

```
DROP ROLE dev_alice;
```