Engineer

RBAC

права, надані ролі, а не людині

Що це таке

RBAC визначає, що комусь дозволено, за ролями, які ця людина має, а не за тим, хто вона. Абревіатура розшифровується як role-based access control, українською — рольове керування доступом або керування доступом на основі ролей, а NIST, де модель формалізували, називає його також «role based security». Роль — це іменований набір дозволів, наприклад «бухгалтер» чи «читання в продакшні». Користувачам призначають ролі, і кожен дозвіл, який користувач застосовує, надходить через одну з них.

Загальну модель запропонували Девід Феррайоло та Рік Кун із NIST у 1992 році як альтернативу тодішньому мандатному й дискреційному керуванню доступом. У 1996 році Сандху, Койн, Файнстайн і Юмен розклали її на рівні: RBAC0 з користувачами, ролями й дозволами, RBAC1 з ієрархіями ролей, RBAC2 з обмеженнями та RBAC3 з тим і іншим. Обидві лінії об'єднали, і вони стали стандартом ANSI/INCITS 359-2004, переглянутим у 2012 році. Його еталонна модель має чотири компоненти. Базовий RBAC (Core RBAC), єдиний обов'язковий для будь-якої системи, містить призначення користувачів ролям і дозволів ролям, а також сесію, у якій користувач активує частину призначених йому ролей. Ієрархічний RBAC дає змогу старшій ролі успадковувати молодшу. Статичне розділення обов'язків обмежує, які ролі можна призначити одному користувачеві разом, а динамічне — які ролі можуть бути активні одночасно в одній сесії: так людина може мати і роль, що створює платіж, і роль, що його затверджує, але не користуватися обома за один раз.

Відмінність від звичайної групи менша, ніж здається. FAQ NIST пояснює це так: група зазвичай є набором користувачів, і більшість групових механізмів дозволяє прикріпити дозвіл просто до користувача, що FAQ називає «лазівкою» (loophole), через яку зв'язок між користувачами й дозволами важко контролювати. RBAC закриває її, прив'язуючи дозволи лише до ролей.

Обґрунтування — в іншому рядку того самого FAQ: ролі в організації відносно стабільні, тоді як користувачів і дозволів багато і вони швидко змінюються, тож коли весь доступ іде через ролі, ним простіше керувати й простіше його переглядати.

Бази даних часто реалізують цю модель напряму. У PostgreSQL з версії 8.1 для цього є лише один вид сутності — роль, яка може бути користувачем, групою або і тим, і тим. Роль, задумана як група, зазвичай не має LOGIN, а членство в ній надають так само, як будь-який інший привілей. Два параметри членства визначають, як член групи користується її привілеями: INHERIT автоматично дає членові привілеї групи, а SET дозволяє перемкнутися на цю роль командою SET ROLE, що близько до ідеї RBAC активувати роль лише в тій сесії, якій вона потрібна.

```
-- роль несе дозволи; під нею ніхто не входить
CREATE ROLE reporting NOLOGIN;
GRANT USAGE ON SCHEMA sales TO reporting;
GRANT SELECT ON ALL TABLES IN SCHEMA sales TO reporting;

-- людина отримує роль, а не таблицю
CREATE ROLE alex LOGIN;
GRANT reporting TO alex;

-- потужна роль, яку вмикають свідомо
CREATE ROLE schema_admin NOLOGIN;
GRANT schema_admin TO alex WITH INHERIT FALSE;
-- згодом, у тій сесії, якій вона потрібна: SET ROLE schema_admin;
```

Застосуйте поняття до рішення

Застосуйте «Що таке RBAC?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Облікові записи, паролі й права налаштовані так, щоб доступ відповідав роботі, а звільнені втрачали його того ж дня.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Написала 1 000 правил RBAC для Linux-серверів — 1,000 правил доступу написано, щоб усунути вразливість

Написала 1 000 правил RBAC для Linux-серверів Зв'язатися з нами →

Чому це важливо

Керування доступом має відповісти на два запитання, які тягнуть у різні боки: чи можна виконати цей запит зараз і хто що може в усій системі. Більшість моделей добре відповідає на перше. RBAC створено для другого, бо

відповідь — це короткий перелік ролей і людей у кожній, який аудитор чи керівник справді може прочитати. Альтернатива — керування доступом на основі атрибутів (ABAC), де кожне рішення перевіряє правила щодо властивостей користувача, ресурсу й контексту, і NIST оцінює її коротко: таку систему може бути легко налаштувати, але складно супроводжувати.

Практичне рішення — де ролей перестає вистачати. Модель 1992 року надає доступ «лише якщо» вимоги ролі виконано, і це формулювання обрано навмисно, щоб лишити місце для додаткових перевірок у момент доступу. Роль лікаря дозволяє читати медичні картки, але не картку кожного пацієнта: зв'язок між цим лікарем і цим пацієнтом — факт про дані, а не назва посади. Саме там потрібен другий механізм, наприклад безпека на рівні рядків у базі даних чи правило на атрибутах у застосунку, накладений поверх ролей, а не замість них.

Наша рекомендація — спершу моделювати доступ як ролі, названі за роботою, яку люди справді виконують, із дозволами, прив'язаними лише до ролей і ніколи напряму до облікового запису для входу. Перелік ролей має бути таким коротким, щоб одна людина могла прочитати його повністю. Додавайте правила на рівні рядків чи атрибутів лише там, де роль явно не може провести межу, і нехай база даних забезпечує все, що може, бо перевірка в базі діє для кожного клієнта, зокрема для шляху в коді, про який хтось забув.

Як це виглядає на практиці

У 2022–2024 роках, у межах роботи з інженерії баз даних, доступ до кожного середовища розробників, інстансу застосунку, бази даних PostgreSQL і MS SQL та Linux-сервера регулювали 1 000 правил RBAC. Сама модель була простою; складність полягала в обсязі. Дозволи розподілили між повторно використовуваними категоріями, як-от «доступ лише для читання до виробничих баз даних», і зіставили з ролями розробників, інстансів застосунків та адміністраторів баз даних. Розробники мали доступ лише до призначених їм серверів, а дозволи інстансів застосунків обмежили, щоб запобігти латеральному переміщенню. Ansible послідовно застосовував правила в усіх середовищах, а самі правила задокументували, щоб команди могли їх перевіряти чи змінювати. Результатом роботи став набір правил на основі ролей і повторюваний спосіб їх застосування та перевірки.

Ролі рідко працюють самі. Коли картографічний застосунок перейшов з MSSQL на PostgreSQL із PostGIS, рольове керування доступом запровадили разом із безпекою на рівні рядків (row-level security), щоб забезпечити політики захисту даних для кількох груп користувачів. Таке поєднання — звичний розподіл праці: ролі визначають, яких таблиць група може торкатися, а політики рядків — які рядки в них.

Найсуворіший варіант узагалі прибирає таблиці з досяжності. У рівні даних PostgreSQL, побудованому на функціях 2025 року, кожна операція застосунку — це одна з 1 275 функцій у 34 схемах. Роль, під якою входить API, mariner, має EXECUTE на функціях застосунку та USAGE на схемах і більше нічого, а це приблизно 4 000 явних грантів замість одного загального. Функції виконуються як SECURITY DEFINER, належать окремій ролі без входу із зафіксованим search_path, а обліковий запис суперкористувача лишається для міграцій і cron. Скомпрометований API все одно обмежувався б тим, що дозволяють функції. Ціну теж названо прямо: нове поле означає міграцію та зміну функції.

Де виникають проблеми

- **Ролі множаться, доки їх неможливо охопити поглядом.** Кожен виняток стає новою роллю, як-от «розробник, крім білінгу» чи «розробник плюс один звіт», і невдовзі ролей більше, ніж людей. Вибух ролей зводить нанівець саму причину RBAC — перелік, достатньо короткий для перегляду. Краще дати людині дві-три ролі, ніж створювати окрему на кожен виняток, і об’єднувати ролі, чиї набори дозволів стали однаковими.
- **Дозвіл потрапляє просто на обліковий запис.** Одна термінова заявка, один GRANT прямо користувачеві — і в нього з’являється доступ, невидимий для перегляду ролей. Це та сама лазівка, яку NIST описує в групових системах, відтворена вручну. Надавайте дозволи лише ролям, а запит для перегляду нехай шукає привілеї в ролей, які можуть входити в систему.
- **Типові налаштування роздають доступ раніше за будь-яку роль.** У PostgreSQL ім’я PUBLIC означає всі ролі, і за замовчуванням воно отримує EXECUTE на нові функції, а також CONNECT і TEMPORARY на бази даних. У базах, оновлених із PostgreSQL 14 чи старішої версії, воно ще й може створювати об’єкти в схемі public. Відкликайте ці типові права свідомо, найкраще в тій самій транзакції, що створює об’єкт.

- **Ролі тільки накопичуються.** Люди змінюють роботу й зберігають стару роль поруч із новою, а тимчасовий доступ стає постійним, бо прибрати його — нічиє завдання. Кожне призначення потребує причини й дати наступного перегляду, а зі зміною роботи мають зникати ролі, що до неї належали.
- **Деякі облікові записи обходять правила.** Суперкористувачі та ролі з атрибутом BYPASSRLS ігнорують безпеку на рівні рядків, і власник таблиці зазвичай теж, якщо для таблиці не ввімкнено FORCE ROW LEVEL SECURITY. Застосунок, що підключається як власник чи суперкористувач, перетворює кожну політику рядків на прикрасу, тож підключайте його через роль, яка нічим не володіє.

Що почитати далі

- NIST Role Based Access Control project — архівна домашня сторінка проєкту: історія моделі, стандарт 2004 року та його перегляд 2012 року.
- NIST RBAC FAQ — ролі й групи, від RBAC0 до RBAC3, розділення обов’язків і чотири компоненти стандарту.
- PostgreSQL: Role Membership — групові ролі, надання й відкликання членства, параметри INHERIT і SET.
- PostgreSQL: Row Security Policies — заборона за замовчуванням, які ролі обходять політики рядків, і FORCE ROW LEVEL SECURITY.
- PostgreSQL: Privileges — що PUBLIC отримує за замовчуванням і як це відкликати.
- SQL Server: Database-level roles — фіксовані й користувацькі ролі всередині бази даних SQL Server.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке RBAC?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/rbac/