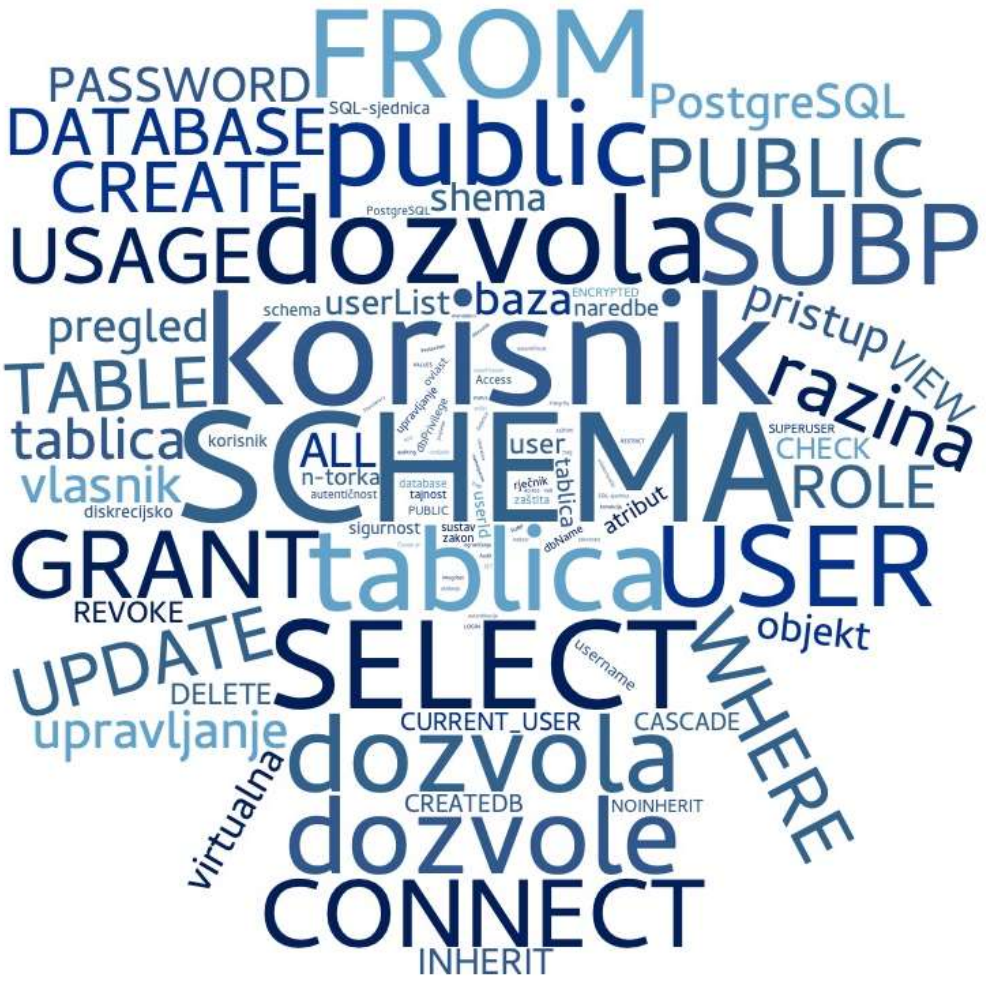
Baze podataka

Predavanja

17. Sigurnost baze podataka

Svibanj, 2021.



Integritet i sigurnost baze podataka

- Pojmovi integritet i sigurnost baze podataka se često spominju zajedno, međutim radi se o dva različita aspekta zaštite podataka
  - Integritet baze podataka (*database integrity*) - operacije nad podacima koje korisnici obavljaju **su ispravne** (tj. uvijek rezultiraju konzistentnim stanjem baze podataka)
    - "podaci se štite od ovlaštenih korisnika"
  - Sigurnost baze podataka (*database security*) - korisnici koji obavljaju operacije nad podacima **su ovlašteni** za obavljanje tih operacija
    - "podaci se štite od neovlaštenih korisnika"

Među ovim pojmovima postoje i sličnosti. U oba slučaja:

- moraju biti definirana **pravila** koja korisnici ne smiju narušiti
- pravila se pohranjuju u rječnik podataka
- SUBP nadgleda rad korisnika - osigurava poštivanje pravila

Oblici narušavanja sigurnosti i moguće posljedice

- Oblici narušavanja sigurnosti baze podataka su:
  - neovlašteno čitanje podataka
  - neovlaštena izmjena podataka
  - neovlašteno uništavanje podataka
- Moguće posljedice su:
  - krađa ili prijevara
  - gubitak tajnosti
    - odnosi se na podatke kritične za funkcioniranje organizacije
    - npr. krađa recepture - rezultira gubitkom konkurentnosti na tržištu
  - gubitak privatnosti
    - odnosi se na osobne podatke
    - npr. krađa podataka o zdravstvenom stanju osobe - rezultira sudskim procesom protiv vlasnika baze podataka
  - gubitak raspoloživosti
    - npr. uništenjem dijela podataka

Protumjere

- sigurnost baze podataka se osigurava zaštitom na nekoliko razina
  - **zaštita na razini SUBP**
    - spriječiti pristup bazama podataka ili onim dijelovima baza podataka za koje korisnici nisu ovlašteni
  - **zaštita na razini operacijskog sustava**
    - spriječiti pristup radnoj memoriji računala ili datotekama u kojima SUBP pohranjuje podatke
  - **zaštita na razini računalne mreže**
    - spriječiti presretanje poruka (*sniffing*) na internetu i intranetu
  - **fizička zaštita**
    - fizički zaštititi lokaciju računalnog sustava
  - **zaštita na razini korisnika**
    - spriječiti da ovlašteni korisnici nepažnjom ili namjerno (npr. u zamjenu za mito ili druge usluge) omoguće pristup podacima neovlaštenim osobama

Aspekti zaštite podataka

- **zakonski, socijalni i etički aspekt**
  - ima li vlasnik baze podataka zakonsko pravo na prikupljanje i korištenje podataka
  - npr. smije li zdravstvena ustanova koja, u skladu sa zakonom prikuplja podatke o pacijentima, te iste podatke koristiti pri donošenju odluke hoće li svog bivšeg pacijenta zaposliti
- **strategijski aspekt**
  - tko definira pravila pristupa - tko određuje kakve ovlasti ima pojedini korisnik baze podataka, ...
- **operativni aspekt**
  - kako osigurati poštivanje pravila - kojim mehanizmima se osigurava poštivanje definiranih pravila, na koji način su lozinke zaštićene, koliko često se mijenjaju, ...

Ustav RH - Članak 37.

Svakom se jamči sigurnost i tajnost osobnih podataka. Bez privole ispitanika, osobni se podaci mogu prikupljati, obrađivati i koristiti samo uz uvjete određene zakonom.

Zakonom se uređuje zaštita podataka te nadzor nad djelovanjem informatičkih sustava u Republici.

Zabranjena je uporaba osobnih podataka suprotna utvrđenoj svrsi njihovoga prikupljanja.

- Zakon o zaštiti osobnih podataka
- GDPR - General Data Protection Regulation
  Opća uredba o zaštiti osobnih podataka koja se primjenjuje od 25. svibnja 2018. godine.

Korisnici SUBP i ovjera autentičnosti

- administrator sustava (operacijskog sustava ili SUBP) omogućuje korisniku pristup sustavu (operacijskom sustavu ili SUBP) definiranjem jedinstvenog identifikatora korisnika (*user name*, *user ID*, *login ID*) i pripadne lozinke (*password*) koja je poznata samo dotičnom korisniku i sustavu
- korisnik koji pristupa sustavu (operacijskom sustavu ili SUBP) poznavanjem lozinke ovjerava svoju autentičnost (*authentication*)
- za ovjeru autentičnosti korisnika SUBP može koristiti

  - vlastite mehanizme

    ili
  - vanjske mehanizme (npr. operacijski sustav)

Autorizacija i modeli upravljanja pristupom

- Autorizacija je postupak kojim se određenom korisniku dodjeljuje dozvola za obavljanje određenih vrsta operacija (čitanje, izmjena, brisanje, ...) nad određenim objektima baze podataka (tablica, pogled, atribut, ...)
  - podaci o dodijeljenim dozvolama pohranjuju se u rječnik podataka
- Prije obavljanja svake operacije, SUBP provjerava ima li korisnik dozvolu za obavljanje operacije nad objektom
  - upravljanje pristupom (*access control*)
- Današnji SUBP podržavaju dva različita modela upravljanja pristupom podacima
  - **Mandatno upravljanje pristupom** (MAC-*Mandatory Access Control*)
  - **Diskrecijsko upravljanje pristupom** (DAC-*Discretionary Access Control*)

Diskrecijsko upravljanje pristupom

- većina današnjih SUBP podržava diskrecijsko upravljanje pristupom
  - podržano je SQL standardom
- određenom korisniku se eksplicitno dodjeljuje dozvola za obavljanje određene operacije nad određenim objektom
  - dozvole su opisane trojkama <korisnik, objekt, vrsta operacije>

    <horvat, ispit, čitanje>

    <horvat, ispit, izmjena>

    <horvat, predmet, čitanje>

    <novak, predmet, čitanje>
  - kada korisnik novak pokuša obaviti operaciju čitanja objekta (tablice) predmet, SUBP provjerava postoji li dozvola u obliku trojke <novak, predmet, čitanje>
- u preostalom dijelu predavanja razmatrat će se diskrecijsko upravljanje pristupom

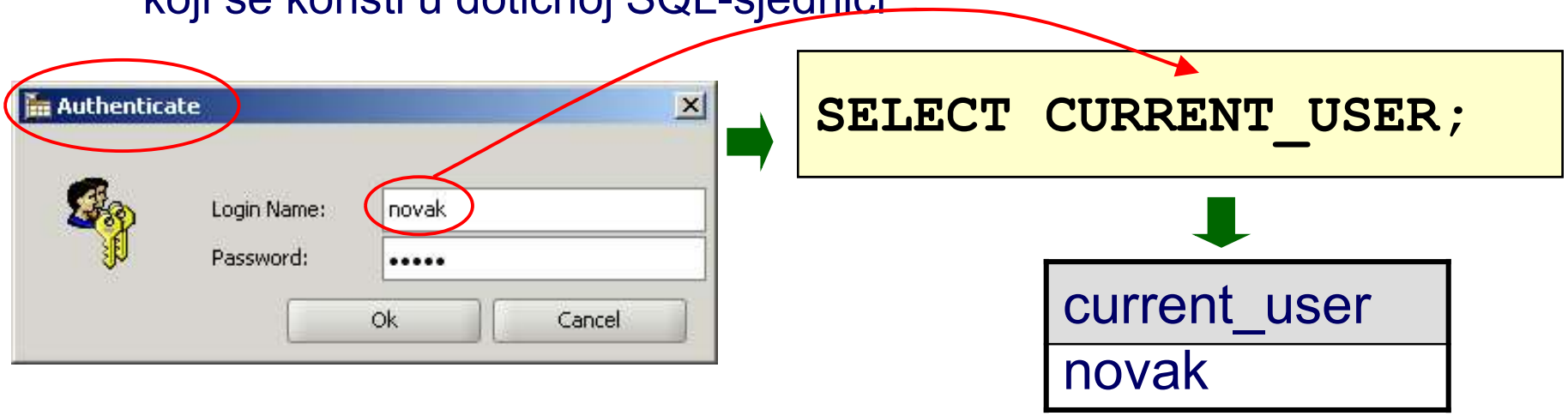
Mandatno upravljanje pristupom

- manji broj SUBP podržava mandatno upravljanje pristupom
  - koristi se relativno rijetko u odnosu na diskrecijsko upravljanje pristupom
- mandatno upravljanje pristupom je primjenjivo u sustavima u kojima se dozvole dodjeljuju na temelju pozicije korisnika u hijerarhiji neke organizacije (vojska, državna uprava, ...)

- svaki **objekt** dobiva oznaku razine tajnosti (*classification level*), npr. povjerljivo, tajno, vrlo tajno, ...
- svakom **korisniku** dodjeljuje se oznaka razine ovlasti (*clearance level*)
  - korisnici mogu obavljati operacije nad onim objektima za koje imaju odgovarajuću razinu ovlasti

Korisnici u SQL-u

- autentificirani korisnik
  - pri uspostavljanju SQL-sjednice korisnik se prijavljuje svojim identifikatorom korisnika, te lozinkom ovjerava svoju autentičnost
  - funkcija CURRENT_USER vraća vrijednost identifikatora korisnika koji se koristi u dotičnoj SQL-sjednici



- bilo koji korisnik (PUBLIC)
  - dodjelom dozvole "korisniku" PUBLIC, dozvolu za obavljanje operacije dobivaju svi sadašnji i budući korisnici

Korisnici u PostgreSQL-u

```
CREATE USER name [ [ WITH ] option [ ... ] ]
where option can be:
      SUPERUSER | NOSUPERUSER
    | CREATEDB | NOCREATEDB
    | CREATEUSER | NOCREATEUSER
    | [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
    | INHERIT | NOINHERIT
    |...
```

- Korisnici se i u PostgreSQL sustavu definiraju na razini SUBP-a, a ne na razini pojedinačne baze podataka
- Za SUPERUSER-a ne postoje ograničenja:

```
CREATE USER the_boss WITH SUPERUSER
                          PASSWORD 'superSecret';
```

- *„Superuser status is dangerous and should be used only when really needed.”*
- CREATEDB - korisnik dobiva ovlast kreiranja baze podataka na PostgreSQL SUBP. NOCREATEDB - preddefinirano ponašanje.

Korisnici u PostgreSQL-u

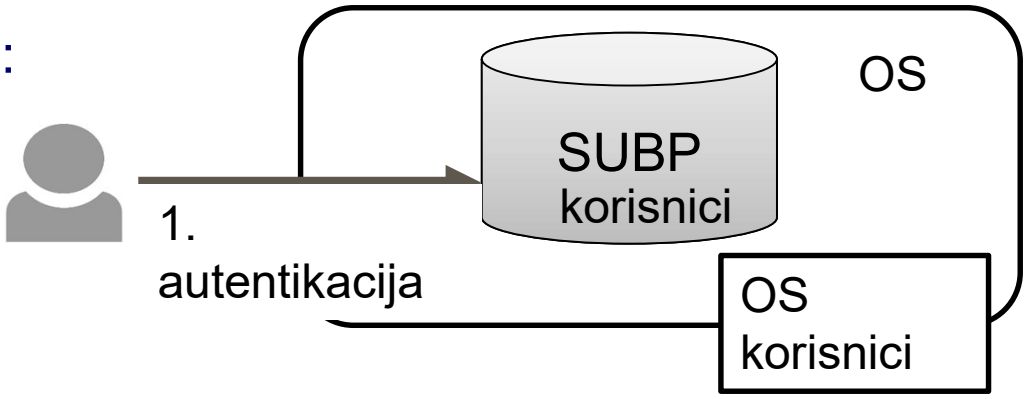
```
...
     | CREATEUSER | NOCREATEUSER
...
```

- CREATUSER - korisnik dobiva ovlast kreiranja drugih korisnika na PostgreSQL SUBP. NOCREATEUSER - preddefinirano ponašanje.
- **INHERIT | NOINHERIT –** bit će objašnjeno kasnije

```
CREATE USER bpadmin WITH CREATEDB CREATEUSER
                         PASSWORD 'bpadminPwd';
```

Metode autentikacije
(PostgreSQL podržava čak 9):

- OS (trust auth)
- Vlastita (password auth)
- …



Objekti i vlasnici objekata u SQL-u

- **Objekti**
  - tablica (relacija, *table*)
  - atribut (stupac tablice, *column*)
  - virtualna tablica (pogled, *view*)
  - sheme (schema) (**to nisu relacijske sheme R{A, B, …}**)
  - baza podataka
- **Vlasnik objekta (*object owner*)**
  - vlasnik objekta je korisnik koji je kreirao objekt, npr:
    - vlasnik baze podataka je korisnik koji je kreirao bazu podataka
    - vlasnik tablice je korisnik koji je kreirao tablicu
  - vlasnik objekta implicitno dobiva dozvole za obavljanje **svih** vrsta operacija nad objektom, uključujući dozvole za:
    - dodjeljivanje svih vrsta dozvola nad tim objektom drugim korisnicima
    - uništavanje objekta

SUBP, baza podataka i sheme (eng. schemas)

- SUBP (*server instance*, *PostgreSQL cluster*) općenito sadrži više (N) baza podataka
  - Korisnici se definiraju na razini cijelog SUBP-a
  - To ne znači da imaju pravo pristupa svim bazama podataka
  - Ne mogu postojati dva korisnika „ivan”
  - Korisnik se pri spajanju na SUBP zapravo spaja na odabranu bazu podataka (npr. *studadmin*)
- PostgreSQL:
  - Baza podataka sadrži jednu ili više **shema**
  - Sheme sadrže tablice, virtualne tablice (, funkcije, …)
  - Različite sheme mogu sadržavati istoimene tablice
  - Sheme analogne s:
    - Mapama u datotečnom sustavu (s tim da se ne mogu gnijezditi)
    - Imenskim područjima (*namespaces*) u programskim jezicima

SUBP/BP/Shema

Svaka nova baza podataka sadrži shemu **`public`** .

Zašto sheme?

- Omogućiti višekorisnički pristup bazi podataka, pri čemu želimo razdijeliti korisnike, odnosno pristup objektima baze podatka (tablice, funkcije, …)
- Organizirati tablice u logičke grupe, kako bi s njima lakše upravljali (npr. javno, interno, admin, …)
- Uspostaviti sustav dozvola (objašnjeno kasnije)

Sheme - SQL

- Stvaranje:

```
CREATE SCHEMA student;

CREATE TABLE student.postavke (
          username TEXT primary key,
          cm_skin  TEXT not null
);
```

- Pristup:

```
-- schema.table
-- database.schema.table
SELECT * FROM student.postavke
```

- Brisanje:

```
DROP SCHEMA student;
-- cannot drop schema student because
-- other objects depend on it
DROP SCHEMA student CASCADE;
-- obrisani i sadržani objekti!!
```

Shema *public* je opcionalna, može se obrisati.

Određivanje sheme (SSP - *schema search path*)

- Ako se u SQL naredbi ne upotrijebi puno ime tablice, SUBP traži tablicu koristeći SSP (SSP je lista shema):
  - Koristi se **prva** pronađena tablica
  - Ako se ne pronađe, javlja se greška (iako tablica tog imena možda postoji u nekoj drugoj shemi, koja nije sadržana u korištenom SSP!)
- Prva shema u SSP se zove **trenutna shema**
- Ako pri kreiranju novih objekata (tablica, …) ne navedemo ime sheme, objekt će se kreirati u trenutnoj shemi.
- Funkcija ***current_schema()*** vraća ime trenutne sheme
- Trenutni SSP se može dobiti naredbom:

```
SHOW search_path;
```

Rezultat je:

search_path
"$user", public

Što znači:

- *"$user"* je prva shema u kojoj će PostgreSQL tražiti objekte, ako postoji. Shema ima isto ime kao trenutni korisnik
- Ako *"$user"* ne postoji, trenutna shema postaje *public*

Određivanje sheme - Primjer

Vlasnik baze podataka obavio je sljedeće naredbe:

```
CREATE USER tibor WITH PASSWORD 'tiborPwd';
CREATE SCHEMA student;
CREATE SCHEMA nastavnik;
```

Nakon uspostavljanja korisničke sjednice s bazom podataka, *tibor* obavlja sljedeću naredbu:

```
CREATE TABLE ocjena
(sifOcjena  INTEGER PRIMARY KEY,
 opisOcjena VARCHAR(2)NOT NULL)
```

U kojoj shemi će biti kreirana tablica ocjena?

U shemi *public* jer shema *tibor* ne postoji. Gornja naredba je ekvivalentna naredbi:

```
CREATE TABLE public.ocjena
(sifOcjena  INTEGER PRIMARY KEY,
 opisOcjena VARCHAR(2)NOT NULL)
```

Vrste dozvola

- Dozvole na razini baze podataka (***dbPrivilege***)
  - Korisnik, da bi pristupio bazi podataka, mora imati dozvolu pristupa bazi podataka
- Dozvole na razini sheme (***schemaPrivilege***)
  - Unutar baze podataka korisnik dobiva dozvole za pojedine sheme
- Dozvole za objekte unutar sheme (***tablePrivilege***)
  - Unutar sheme korisnik dobiva dozvole na pojedine tablice ili virtualne tablice

Vrste dozvola u SQL-u na razini baze podataka

- Različiti SUBP imaju različita rješenja za dodjeljivanje dozvola na razini baze podataka.
- PostgreSQL:
  - **CONNECT**
    - Dozvoljava spajanje (uspostavljanje SQL-sjednice) na bazu podataka
    - Korisnik koji se spojio na bazu podataka može obavljati operacije nad objektima za koje je dobio dozvolu od vlasnika objekta ili je njihov vlasnik
    - Preddefinirano ponašanje je da korisnik PUBLIC ima CONNECT dozvolu na bazu podataka u PostgreSQL SUBP – to znači da bilo koji korisnik koji se prijavio na sustav može pristupiti bilo kojoj bazi podataka unutar sustava
  - **CREATE**
    - Dozvoljava stvaranje novih shema u bazi podataka

Vrste dozvola u SQL-u na razini sheme

- PostgreSQL:
  - **USAGE**
    - Nužan preduvjet za pristupanje objektima sadržanima u shemi. Ne podrazumijeva nikakve daljnje dozvole za konkretne objekte u shemi.
  - **CREATE**
    - Dozvoljava stvaranje novih objekata (tablice, funkcije, …) u shemi.
  - Preddefinirano ponašanje:
    - Korisnik nema dozvolu pristupa nijednom objektu sheme kojoj nije vlasnik.
    - Za pristup mu vlasnik sheme treba dodijeliti dozvolu USAGE
    - Za kreiranje objekata u shemi, dodatno mora dobiti CREATE
    - PUBLIC ima CREATE i USAGE dozvole za shemu *public*

Vrste dozvola u SQL-u na razini [virtualne] tablice

- **SELECT [(*columnList*)]**
  - čitanje n-torki (ili vrijednosti navedenih atributa) [virtualne] tablice
- **UPDATE [(*columnList*)]**
  - Izmjena n-torki (ili vrijednosti navedenih atributa) [virtualne] tablice
- **INSERT [(*columnList*)]**
  - unos n-torki (ili vrijednosti navedenih atributa) [virtualne] tablice
- **DELETE**
  - brisanje n-torki [virtualne] tablice
- **ALL PRIVILEGES**
  - sve do sada navedene vrste operacija nad [virtualnom] tablicom

- itd. gore je naveden samo dio dozvola

SQL naredbe za dodjeljivanje i ukidanje dozvola

- GRANT *dbPrivilege* ON DATABASE name TO { PUBLIC | *userList* }
- REVOKE *dbPrivilege* ON DATABASE name FROM { PUBLIC | *userList* }

- GRANT *schemaPrivilege* ON SCHEMA name TO { PUBLIC | *userList* }
- REVOKE *schemaPrivilege* ON SCHEMA name FROM { PUBLIC | *userList* }

- GRANT *tablePrivilegeList* ON { *tableName* | *viewName* }
  TO { PUBLIC | *userList* }
  [WITH GRANT OPTION]
- REVOKE *tablePrivilegeList* ON { *tableName* | *viewName* }
  FROM { PUBLIC | *userList* }
  [CASCADE | RESTRICT]

PostgreSQL - preddefinirane dozvole korisnika PUBLIC

- Ključna riječ PUBLIC (**<> shema *public*!**)
  - Označava sve korisnike, čak i one koji će tek nastati
- PgSQL - preddefinirane dozvole korisnika PUBLIC:
  - Dozvola uspostavljanja konekcije sa svim bazama na PgSQL SUBP

```
GRANT CONNECT ON DATABASE * TO PUBLIC;
```

  - Dozvole USAGE i CREATE za sve sheme public u svim bazama na PgSQL SUBP

```
GRANT ALL (USAGE, CREATE) ON SCHEMA public TO
PUBLIC;
```

  - Primijetite da PUBLIC nema nikakvu dozvolu na razini tablica u shemi **`public`**

Primjer

Mnogi koriste ovakav sustav (u produkciji), za nešto strože inicijalne postavke sigurnosti:

```
--ACCESS DB
REVOKE CONNECT ON DATABASE dbName FROM PUBLIC;
GRANT  CONNECT ON DATABASE dbName TO user;

--ACCESS SCHEMA
REVOKE ALL     ON SCHEMA public FROM PUBLIC;
GRANT  USAGE   ON SCHEMA public  TO user;

--ACCESS TABLES (pretpostavka je da postoje dolje navedene uloge)
GRANT SELECT                ON ALL TABLES IN SCHEMA public TO read_only;
GRANT SELECT, INSERT,
      UPDATE, DELETE        ON ALL TABLES IN SCHEMA public TO read_write;
GRANT ALL                   ON ALL TABLES IN SCHEMA public TO admin;
```

Primjer 1 (PostgreSQL):

student	matBr	ime	prez	pbr	adresa

ispit	matBr	nazPred	datIsp	ocj

Korisnik bpadmin treba
- kreirati bazu podataka studBaza.
- korisniku PUBLIC ukinuti dozvolu spajanja na studBaza
- korisniku PUBLIC ukinuti sve dozvole za shemu *public* u studBaza
- kreirati tablice student i ispit
- kreirati korisnike *horvat*, *novak* i *kolar* i omogućiti im spajanje na studBaza i korištenje *public* sheme u studBaza

- korisnik *horvat* treba dobiti dozvole:
  - pregled svih podataka u tablicama student i ispit
  - unos, izmjena, brisanje svih podataka u tablici ispit
- korisnik *novak* treba dobiti dozvole:
  - pregled svih podataka u tablici student
  - izmjena poštanskog broja i adrese u tablici student
- korisnik *kolar* treba dobiti dozvolu:
  - pregled svih podataka u tablici student, osim adrese

Primjer 1 (nastavak, PostgreSQL):

postgres ← **naredbu obavlja korisnik *postgres (SUPERUSER)***

```
CREATE USER bpadmin WITH CREATEDB CREATEROLE
                         PASSWORD 'bpadminPwd';
```

➡ korisnik bpadmin dobiva dozvolu kreiranja baza podataka i korisnika.

bpadmin ← **naredbe obavlja korisnik *bpadmin***

```
CREATE DATABASE studbaza;

REVOKE CONNECT ON DATABASE studBaza
   FROM PUBLIC;
```

➡ korisnik bpadmin je vlasnik baze podataka studBaza. Može ukinuti preddefiniranu dozvolu CONNECT korisniku PUBLIC.

postgres

```
REVOKE ALL ON SCHEMA public FROM PUBLIC;
```

➡ vlasnik sheme *public* u svakoj bazi podataka je korisnik *postgres* (specifičnost PgSQL). Korisnik *bpadmin* nema ovlasti za ovu naredbu.

bpadmin

```
CREATE TABLE student (...);
CREATE TABLE ispit (...);

CREATE USER horvat;
CREATE USER kolar;
CREATE USER novak;
```

➡ kreiranje novih objekata u bazi. tablice će biti kreirane u shemi public.

➡ kreiranje korisnika s mogućnošću uspostavljanja SQL-sjednice na razini SUBP

Primjer 1 (nastavak, PostgreSQL):

bpadmin

```
GRANT CONNECT ON DATABASE studbaza TO horvat;
GRANT CONNECT ON DATABASE studbaza TO novak;
GRANT CONNECT ON DATABASE studbaza TO kolar;

GRANT USAGE ON SCHEMA public TO horvat;
GRANT USAGE ON SCHEMA public TO novak;
GRANT USAGE ON SCHEMA public TO kolar;

GRANT SELECT ON student TO horvat;


GRANT SELECT, INSERT
    , UPDATE, DELETE ON ispit
  TO horvat;


GRANT SELECT ON student TO novak;


GRANT UPDATE(pbr, adresa)
  ON student TO novak;

GRANT SELECT(matBr, ime
           , prez, pbr)
  ON student TO kolar;
```

- dozvole spajanja na studBaza. Treba jer je ukinuta CONNECT dozvola za PUBLIC.
- dozvola korištenja sheme *public*. Treba jer je ukinut USAGE i CREATE za PUBLIC.
- dozvola korisniku *horvat* za pregled podataka u tablici student
- dozvole korisniku horvat za pregled, unos, izmjenu i brisanje podataka u tablici ispit
- dozvola korisniku *novak* za pregled podataka u tablici student
- dozvola korisniku *novak* za izmjenu vrijednosti atributa u tablici student
- dozvola korisniku *kolar* za pregled svih podataka u tablici student, osim adrese

Primjer 2 (PostgreSQL):

bpadmin

```
CREATE DATABASE studBaza;
CREATE SCHEMA student;

CREATE TABLE student.postavke (
  username text primary key, ...);
CREATE TABLE postavkePub(
 username text primary key, ...);
```

korisnik bpadmin kreira bazu podataka studBaza, te dvije tablice, jednu u shemi student, drugu u PUBLIC shemi

Sjetimo se: PostgreSQL (*default*) daje CONNECT dozvolu korisniku PUBLIC!

tibor

```
CREATE TABLE postavkeTib (...)
INSERT INTO postavkeTib VALUES (..);
```

Može, jer :

- ima CONNECT (bez CONNECT ne bi mogao uspostaviti SQL-sjednicu),
- ima USAGE i CREATE za shemu public u kojoj se stvara *postavketib*
- je vlasnik *postavketib* pa može obaviti INSERT

tibor

```
SELECT * FROM postavkePub;
INSERT INTO postavkePub VALUES(...);
SELECT * FROM student.postavke;
INSERT INTO student.postavke ...;
CREATE TABLE student.T2(...);
CREATE SCHEMA moja;
```

NE može, jer:

- USAGE na shemu *public* ne uključuje dozvole za operacije nad tablicama
- Nije mu dana dozvola za student.postavke
- Nema dozvole (USAGE) za shemu student
- Nema dozvole za stvaranje sheme

Primjer 2 (nastavak):

tibor

```
DROP TABLE postavkePub;
```

ne može jer nije vlasnik objekta (niti je SUPERUSER)

kolar

```
SELECT * FROM postavkePub;
```

NE može, jer nema dozvole za operacije nad postavkePub

tibor

```
GRANT CONNECT ON DATABASE
 studBaza TO kolar;
```

ne može jer nije SUPERUSER

tibor

```
GRANT SELECT
  ON postavkeTib TO kolar;
```

Može, jer je **vlasnik** tablice *postavkeTib*

Primjer 2 (nastavak):

postgres

```
GRANT CREATE ON DATABASE
 studBaza TO tibor;
```

➡ Može, jer je SUPERUSER

tibor

```
CREATE SCHEMA tibor;
```

➡ Može, jer sad ima dozvolu

tibor

```
CREATE TABLE tibor.postavke(...);
GRANT SELECT ON tibor.postavke TO kolar;
```

➡ Može, jer je vlasnik sheme

kolar

```
SELECT * FROM tibor.postavke;
```

➡ Ne može, jer nema dozvolu na shemu (ima samo na tablicu)

tibor

```
GRANT USAGE ON SCHEMA tibor TO kolar;
```

➡ Može, jer je vlasnik sheme

kolar

```
SELECT * FROM tibor.postavke;
```

➡ Može

Dodjeljivanje prenosivih dozvola

- Ako se korisniku dozvola dodijeli uz navođenje opcije WITH GRANT OPTION, korisnik će moći dodjeljivati tu istu dozvolu ostalim korisnicima (unatoč tome što nije vlasnik objekta)

Primjer:

korisnik1

```
CREATE TABLE ispit (...);
GRANT SELECT ON ispit TO korisnik2 WITH GRANT OPTION;
GRANT SELECT ON ispit TO korisnik3 WITH GRANT OPTION;
```

korisnik2

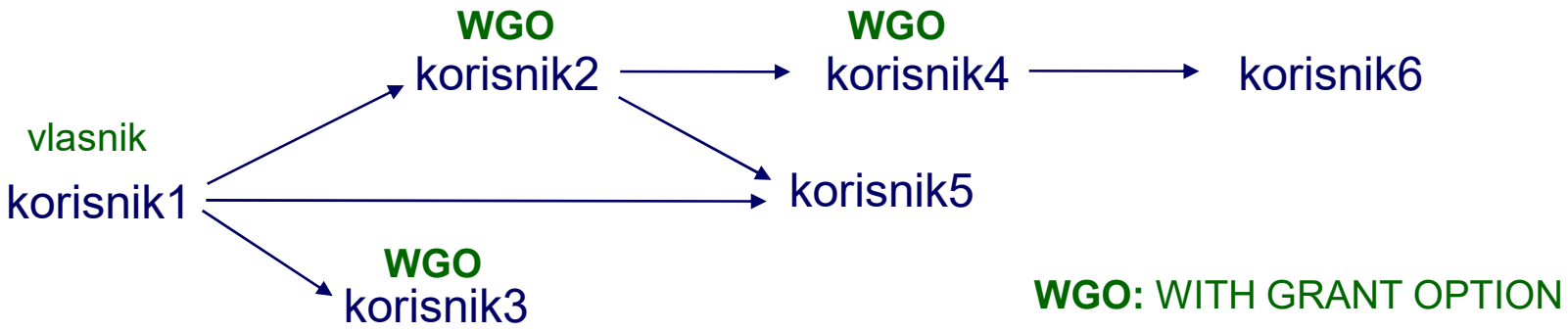
```
GRANT SELECT ON ispit TO korisnik4 WITH GRANT OPTION;
GRANT SELECT ON ispit TO korisnik5;
```

korisnik4

```
GRANT SELECT ON ispit TO korisnik6;
```

korisnik1

```
GRANT SELECT ON ispit TO korisnik5;
```



Ukidanje dozvola

- korisnik koji je dozvolu dodijelio, tu istu dozvolu može ukinuti naredbom REVOKE

Primjer:
- vlasnik baze podataka studBaza je korisnik bpadmin
- vlasnik tablice mjesto je korisnik horvat

horvat

```
GRANT SELECT, UPDATE ON mjesto TO novak WITH GRANT OPTION;
```

novak

```
GRANT SELECT, UPDATE ON mjesto TO kolar;
```

- npr. naredbu:

```
REVOKE UPDATE ON mjesto FROM kolar;
```

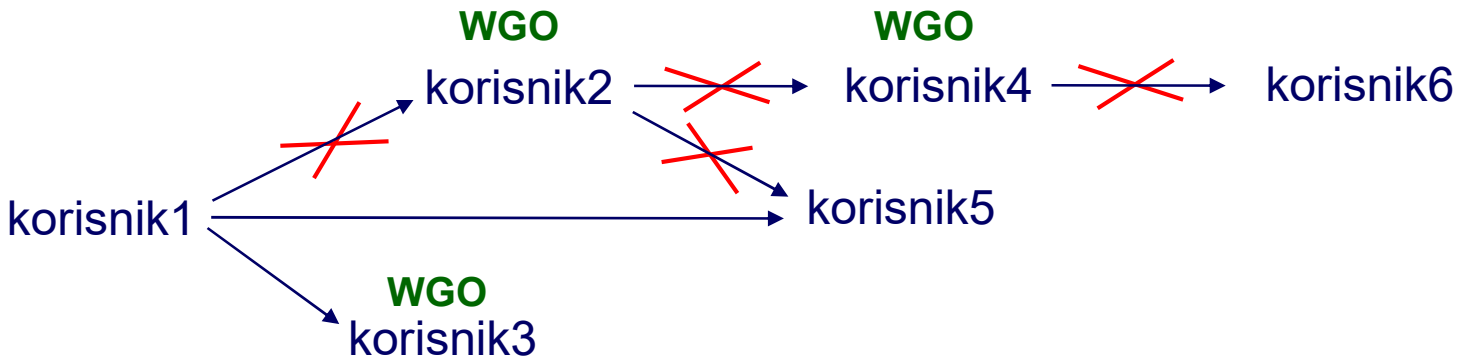
- može obaviti korisnik novak jer je novak korisnik koji je dozvolu dodijelio

Ukidanje dozvola dodijeljenih temeljem WITH GRANT OPTION

- ukidanjem dozvole korisniku x (koji je dozvole dalje dodjeljivao temeljem ovlasti stečene pomoću WITH GRANT OPTION) uz primjenu opcije CASCADE, dozvola se ukida i svim ostalim korisnicima koji su dotičnu dozvolu stekli od korisnika x (neposredno ili posredno)

Primjer: `korisnik1`

```
REVOKE SELECT ON ispit FROM korisnik2 CASCADE;
```



- obavljanjem naredbe dozvolu gube korisnik2, korisnik4 i korisnik6
- korisnik5 će izgubiti dozvolu koju je dobio od korisnika2, ali će zadržati dozvolu koju je dobio od korisnika1
- ukoliko se opcija CASCADE ne navede, naredba REVOKE neće uspjeti ako postoje dodatne neposredne dozvole

Primjena virtualnih tablica u kontekstu dozvola

ispit

mbrSt	nazPred	datIsp	ocj
100	Fizika	1.5.2010	3
102	Matematika	7.9.2009	1
102	Matematika	9.2.2010	5
107	Fizika	5.4.2012	4

- vlasnik tablice ispit je korisnik horvat
- korisniku novak treba omogućiti pregled samo prosječnih ocjena po predmetima
- korisniku kolar treba omogućiti pregled, unos, izmjenu i brisanje samo za ispite iz predmeta Fizika

horvat

```
CREATE VIEW prosjek (nazPred, prosOcj) AS
  SELECT nazPred, AVG(ocj)
    FROM ispit
    GROUP BY nazPred;
GRANT SELECT ON prosjek TO novak;

CREATE VIEW ispitFizika AS
  SELECT * FROM ispit
    WHERE nazPred = 'Fizika'
  WITH CHECK OPTION;
GRANT SELECT, INSERT, UPDATE, DELETE
  ON ispitFizika TO kolar;
```

zašto je nužno virtualnu tablicu ispitFizika kreirati uz opciju WITH CHECK OPTION?!

Dodjeljivanje kontekstno ovisnih dozvola

ispit

mbrSt	sifPred	datIsp	ocj
100	100	1.5.2010	3
102	200	7.9.2009	1
102	200	9.2.2010	5
107	300	5.4.2012	4

nast

sifNast	imeN	prezN	userId
1001	Slavko	Kolar	kolar
1002	Ivo	Ban	ban
1003	Ana	Novak	novak

predaje

sifNast	sifPred
1001	100
1001	200
1002	200
1003	200
1003	300

- vlasnik tablica je korisnik horvat
- svakom nastavniku (korisnicima kolar, ban, novak) omogućiti pregled i izmjenu ispita samo iz predmeta koje predaju

LOŠE RJEŠENJE!

horvat

```
CREATE VIEW kolarIspiti AS
  SELECT * FROM ispit
    WHERE sifPred IN (
      SELECT sifPred FROM predaje
        WHERE sifNast = 1001) WITH CHECK OPTION;
GRANT SELECT, UPDATE ON kolarIspiti TO kolar;
```

- ponoviti za svakog nastavnika: banIspiti, novakIspiti, ...
- nova virtualna tablica za svakog novog nastavnika (≈150 na FER-u)
- svaki nastavnik upit nad tablicom ispit mora pisati na drugačiji način

Dodjeljivanje kontekstno ovisnih dozvola

ispit

mbrSt	sifPred	datIsp	ocj
100	100	1.5.2010	3
102	200	7.9.2009	1
102	200	9.2.2010	5
107	300	5.4.2012	4

nast

sifNast	imeN	prezN	userId
1001	Slavko	Kolar	kolar
1002	Ivo	Ban	ban
1003	Ana	Novak	novak

predaje

sifNast	sifPred
1001	100
1001	200
1002	200
1003	200
1003	300

ISPRAVNO RJEŠENJE!

horvat

```
CREATE VIEW ispitiZaNastavnike AS
  SELECT * FROM ispit
    WHERE sifPred IN (
      SELECT sifPred FROM predaje, nast
       WHERE predaje.sifNast = nast.sifNast
         AND userId = CURRENT_USER) WITH CHECK OPTION;
GRANT SELECT, UPDATE ON ispitiZaNastavnike TO kolar;
GRANT SELECT, UPDATE ON ispitiZaNastavnike TO ban;
GRANT SELECT, UPDATE ON ispitiZaNastavnike TO novak;
```

- "sadržaj" virtualne tablice ovisit će o identifikatoru nastavnika koji je ostvario SQL-sjednicu
- smije li se nastavnicima dozvoliti izmjena vrijednosti atributa userId u tablici nast ili sadržaj tablice predaje?!

Dodjeljivanje istih dozvola velikom broju korisnika

PROBLEM:

- svakom nastavniku treba dodijeliti dozvole za
  - pregled, unos i izmjenu podataka o ispitima za predmete koje predaje, pregled podataka iz tablice *nast*, iz tablice *predaje*, itd.
  - 150 nastavnika⇒150 puta treba obaviti niz naredbi za dodjelu dozvola:

```
GRANT SELECT, INSERT, UPDATE ON ispitiZaNastavnike TO kolar;
GRANT SELECT ON predmet TO kolar;
GRANT SELECT ON nast TO kolar;
...
-- ponoviti za svakog od 150 nastavnika
```

- za svakog novog zaposlenog nastavnika ponoviti postupak
- kada nastavnik ode u mirovinu, mora se obaviti niz REVOKE naredbi
- ako se promijene pravila pristupa (npr. odluči se da nastavnici mogu brisati "svoje" ispite), promjena se mora provesti za svakog nastavnika posebno:

```
GRANT DELETE ON ispitiZaNastavnike TO kolar;
-- ponoviti za svakog od 150 nastavnika
```

PostgreSQL uloge

RJEŠENJE:

- definira se uloga (*role*), npr. *nastavnik*
- dozvole se, umjesto direktno korisnicima, dodjeljuju novoj ulozi
- uloga može predstavljati jednog ili više korisnika
- uloge se, kao i korisnici, definiraju na razini cijelog SUBP-a

```
CREATE ROLE name [ [ WITH ] option [ ... ] ]
where option can be:
      | CREATEDB | NOCREATEDB
      | CREATEROLE | NOCREATEROLE
      | [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
      | INHERIT | NOINHERIT
      | LOGIN | NOLOGIN
...
```

Nalik opcijama CREATE USER naredbe

INHERIT znači da uloga (automatski) nasljeđuje dozvole eventualnih dodatnih uloga koje su joj dodijeljene. INHERIT je preddefinirano ponašanje.

Dodjeljivanje istih dozvola velikom broju korisnika

```
CREATE ROLE nastavnik;
GRANT SELECT, INSERT, UPDATE ON ispitiZaNastavnike TO nastavnik;
GRANT SELECT ON nast TO nastavnik;
GRANT SELECT ON predaje TO nastavnik;
...
```

- svakom nastavniku, umjesto cijelog niza dozvola, dovoljno je dodijeliti dozvolu za korištenje uloge nastavnik

```
GRANT nastavnik TO kolar;
GRANT nastavnik TO ban;
...
```

- uloga/korisnik aktivira drugu ulogu uz pomoć naredbe SET ROLE
- Ako je korisnik kreiran s preddefiniranom INHERIT opcijom (to je slučaj u našem primjeru) nije potrebno aktivirati ulogu jer ionako automatski ima sve njene dozvole.

ban:
```
SET ROLE nastavnik;
```

- ako nastavnik s identifikatorom korisnika ban ode u mirovinu

```
REVOKE nastavnik FROM ban;
```

- ako nastavnici trebaju dobiti dozvolu za brisanje "svojih" ispita

```
GRANT DELETE ON ispitiZaNastavnike TO nastavnik;
```

Problem

- nastavnici (odnosno aplikacije koje nastavnici koriste) moraju u upitima o ispitima koristiti virtualnu tablicu ispitiZaNastavnike

```
SELECT * FROM ispitiZaNastavnike WHERE ocj = 1;
```

- dekan (npr. korisnik s identifikatorom novosel), za razliku od nastavnika, dobiva sve dozvole nad tablicom ispit. U upitima o ispitima mora koristiti tablicu ispit

```
SELECT * FROM ispit WHERE ocj = 1;
```

- kada korisnik novosel prestane biti dekan, ukinut će mu se dozvola nad tablicom ispit, a dodijeliti dozvola nad virtualnom tablicom ispitiZaNastavnike. U svojim upitima morat će koristiti virtualnu tablicu ispitiZaNastavnike

```
SELECT * FROM ispitiZaNastavnike WHERE ocj = 1;
```

RJEŠENJE: Upotreba Schema Search Patha

SUPERUSER

```
REVOKE ALL  ON SCHEMA public FROM PUBLIC;
CREATE ROLE nastavnik;
CREATE ROLE dekan;

CREATE SCHEMA nastavnik;
CREATE SCHEMA dekan;

GRANT USAGE ON SCHEMA nastavnik TO nastavnik;
GRANT USAGE ON SCHEMA dekan TO dekan;

CREATE VIEW dekan.ispitizasve     AS SELECT * FROM ispit ...;
CREATE VIEW nastavnik.ispitizasve AS SELECT * FROM ispitiZaNastavnike;

GRANT SELECT, … ON nastavnik.ispitizasve TO nastavnik;
GRANT SELECT, … ON dekan.ispitizasve     TO dekan;

CREATE USER horvat  WITH PASSWORD 'horvatPwd';  --WITH INHERIT default
CREATE USER novosel WITH PASSWORD 'novoselPwd'; --WITH INHERIT default

GRANT CONNECT ON DATABASE studadmin TO horvat;
GRANT CONNECT ON DATABASE studadmin TO novosel;

GRANT nastavnik TO horvat;
GRANT dekan     TO novosel;
```

RJEŠENJE: Upotreba Schema Search Patha

- sada i dekan i nastavnici mogu koristiti isto ime objekta kada postavljaju upite o ispitima:

horvat

```
SET ROLE nastavnik;
SELECT * FROM ispitizasve WHERE ocj = 1;
```

novosel

```
SET ROLE dekan;
SELECT * FROM ispitizasve WHERE ocj = 1;
```

- Zbog korištenja preddefinirane SSP, PostgreSQL prvo traži tablice u shemi "$user" te svaki korisnik treba aktivirati odgovarajuću ulogu.
  - Ako bi korisnik želio **samo** dobiti odgovarajuće dozvole, ne bi morao aktivirati ulogu, jer je sve dozvole dobio temeljem preddefiniranog svojstva u naredbi CREATE USER/ CREATE ROLEINHERIT u PostgreSQL .
  - Korisnik mora aktivirati ulogu kako bi mu se pridijelio odgovarajući SSP
- Kada korisnik novosel prestane biti dekan:

SUPERUSER

```
REVOKE dekan      FROM novosel;
GRANT nastavnik   TO novosel;
```

novosel

```
SET ROLE nastavnik;
SELECT * FROM ispitizasve WHERE ocj = 1;
```

Pazi - postoji problem u rješenju

- korisnik **horvat** uspostavlja sjednicu

```
SELECT SESSION_USER, CURRENT_USER
```

session_user	current_user
horvat	horvat

- **horvat** aktivira ulogu **nastavnik**

```
SET ROLE nastavnik;
SELECT SESSION_USER, CURRENT_USER
```

session_user	current_user
horvat	nastavnik

Kad korisnik ima aktiviranu neku ulogu, tada funkcija CURRENT_USER vraća **ime dotične uloge**!

Zbog toga, pri definiciji kontekstno ovisnih virtualnih tablica koje se odnose na trenutnog korisnika, umjesto funkcije CURRENT_USER treba koristiti **SESSION_USER**

```
CREATE VIEW ispitiZaNastavnike AS
  SELECT * FROM ispit
    WHERE sifPred IN (
      SELECT sifPred FROM predaje, nast
       WHERE predaje.sifNast = nast.sifNast
         AND userId = CURRENT_USER) WITH CHECK OPTION;
```

Treba koristiti **SESSION_USER**!

Praćenje rada korisnika (*auditing*)

- evidentirati svaki pristup osjetljivim podacima u posebnoj datoteci za praćenje rada korisnika (*Audit Trail*)
- tipičan zapis datoteke sadrži sljedeće informacije:
  - SQL naredba koja se izvršava (*statement source*)
  - mjesto s kojeg je upućen zahtjev (terminal, IP adresa računala)
  - identifikator korisnika koji je pokrenuo operaciju
  - datum i vrijeme operacije
  - n-torke, atributi na koje se zahtjev odnosi
  - stara vrijednost n-torke
  - nova vrijednost n-torke
- sama činjenica da se prati "trag" obavljenih operacija nad podacima, često je dovoljna za sprečavanje zloporabe