ALGEBRA

Pristup podacima iz programskog koda

Vježbe 02 – Postgres funkcionalnosti i postavljanje instance (Docker)

Prije nego što počnemo!

Extracurricular classes

ALGEBRA

EEG projekt!

Sign up with this link

ALGEBRA

Sadržaj

- Podatkovni tipovi
- SQL upiti – podsjetnik
  - DDL, DML, agregatne funkcije, podupiti
- Sequence
- Indeksi
- Transakcije
- Docker setup

SQL

- DDL (Data Definition Language)
  - CREATE, DROP, ALTER
- DML (Data Modifying Language)
  - CRUD (INSERT, SELECT, UPDATE, DELETE)
  - filtracija (WHERE) i sortiranje (ORDER BY)
  - agregatne funkcije (AVG, COUNT, SUM, MIN, MAX) s grupiranjem (GROUP BY)

SQL - funkcije

```
CREATE OR REPLACE FUNCTION species_category(sp_name TEXT)
RETURNS TEXT AS $$
BEGIN
    IF sp_name ILIKE 'a%' THEN
        RETURN 'Group A';
    ELSIF sp_name ILIKE 'b%' THEN
        RETURN 'Group B';
    ELSE
        RETURN 'Other';
    END IF;
END;
$$ LANGUAGE plpgsql IMMUTABLE;
```

- Funkcije mogu biti **IMMUTABLE**, **STABLE** i **VOLATILE** (ovisno o kontekstu korištenja)

Podatkovni tipovi - 1

bigint	**int8**	signed eight-byte integer
bigserial	**serial8**	autoincrementing eight-byte integer
bit [(n)]		fixed-length bit string
bit varying [(n)]	varbit [(n)]	variable-length bit string
boolean	**bool**	logical Boolean (true/false)
box		rectangular box on a plane
bytea		binary data (“byte array”)
character [(n)]	**char** [(n)]	fixed-length character string
character varying [(n)]	**varchar** [(n)]	variable-length character string
cidr		IPv4 or IPv6 network address
circle		circle on a plane
date		calendar date (year, month, day)
double precision	**float**, **float8**	double precision floating-point number (8 bytes)
inet		IPv4 or IPv6 host address
integer	**int**, **int4**	signed four-byte integer
interval [fields] [(p)]		time span
json		textual JSON data
jsonb		binary JSON data, decomposed

ALGEBRA

Podatkovni tipovi - 2

line		infinite line on a plane
lseg		line segment on a plane
macaddr		MAC (Media Access Control) address
macaddr8		MAC (Media Access Control) address (EUI-64 format)
money		currency amount
numeric [(p, s)]	decimal [(p, s)]	exact numeric of selectable precision
path		geometric path on a plane
pg_lsn		PostgreSQL Log Sequence Number
pg_snapshot		user-level transaction ID snapshot
point		geometric point on a plane
polygon		closed geometric path on a plane
real	**float4**	single precision floating-point number (4 bytes)
smallint	**int2**	signed two-byte integer
smallserial	**serial2**	autoincrementing two-byte integer
serial	**serial4**	autoincrementing four-byte integer
text		variable-length character string

Podatkovni tipovi - 3

time [(p)] [without time zone]		time of day (no time zone)
time [(p)] with time zone	**timetz**	time of day, including time zone
timestamp [(p)] [without time zone]		date and time (no time zone)
timestamp [(p)] with time zone	**timestamptz**	date and time, including time zone
tsquery*		text search query
tsvector*		text search document
txid_snapshot		user-level transaction ID snapshot (deprecated; see pg_snapshot)
uuid		universally unique identifier
xml		XML data

*ovi tipovi će biti spomenuti u predavanju koje će se baviti stringovima i njihovim podudaranjem

Preuzeto iz:
https://www.postgresql.org/docs/current/datatype.html

Postgres Sequence

- Funkcije:
  - `nextval('seq')`
    - Inkrementira i vraća sljedeću vrijednost
  - `currval('seq')`
    - Posljednja vrijednost korištena u ovoj sesiji
  - `setval('seq', N)`
    - Postavlja vrijednost sekvence na N
- Od PG 10+, identity stupci (`GENERATED ALWAYS AS IDENTITY`) se preferiraju na `SERIAL` tipovima.

Postgres Sequence

- Sekvence su specijalne tablice koje sadrže jedan red čija je svrha generiranje jedinstvenih vrijednosti
- Kreirana implicitno kada se koristi `SERIAL/BIGSERIAL` podatkovni tip ili eksplicitno putem `CREATE SEQUENCE`
  - u pozadini je sadržan brojač u kataloškoj relaciji koji je pohranjen u `pg_sequence` sistemskom katalogu
  - Jednom kad je inkrementirana, vrijednost je izgubljena čak i ako je transkacija resetirana (*rollback*) – osigurava jedinstvenost

Index

`CREATE INDEX name ON table [USING <algorithm>] (column, <…>)`

- B-Tree (default, operatori: `<, <=, =, >=, >`)
- GIN – full-text search / JSONB /arrays
- GiST – geometric, full-text
- BRIN – veliki sekvencijski podaci (time-series)
- Hash – usporedbe čiste jednakosti

- Single index
- Partial index
  - Samo redovi koji zadovoljavaju dani uvjet će biti indeksirani
- Composite index

ALGEBRA

Index

- Koji su problemi s indeksiranjem?
  - Umetanje i brisanje !!
- `animal` tablica sadrži podatke o taksonomski određenjima različitih životinjskih vrsta (`kingdom`, `phlyum`, `class`, `order`, `familiy`, `genus`, `species`)

```
create index animal_canonical_name_idx on animal(canonical_name)
```

- Koji upiti će koristit novostvoreni indeks?
  - hint: koristite `EXPLAIN`

Transakcije

- Transkacije je logička jedinica rada koja izvrašava jedan ili više SQL upita kao jedinstvenu atomičku operaciju.
- Ili će se sve stavke izvršiti (**commit**) ili nijedna (**rollback**)

```
BEGIN;
-- operacije…
COMMIT;
```

- Opcionalni *reset* ukoliko stvari pođu po zlu:

```
ROLLBACK;
```

- Možemo stvoriti *savepointe* i resetirati stanje do specifičnog *savepointa* koristeći njegovo ime:
  - U tijelu transkacije: `SAVEPOINT point1;`
  - Potom reset: `ROLLBACK point1;`

Problemi prilikom izvođenja transkacija

- Zaboravite li izvršiti COMMIT, transakcija zadržava lockove i blokira druge transakcije
- Transkacije koje se izvode duže vremena spriječavaju VACUUM proces koji oslobođa zauzeti prostor mrtvih n-torki

```
SELECT pid, usename, state, query,
xact_start
FROM pg_stat_activity
WHERE state = 'active';
```

- Dvije transakcije koje čekaju oslobođenje locka ove druge - **deadlocks**, PostgreSQL detektira takvo stanje i odbacuje jednu transakciju

```
SELECT * FROM pg_locks;
```

Razine izolacije

Razina	Opis	Riješava problem
READ UNCOMMITTED	Ponaša se kao READ COMMITTED (Postgres ne dozvoljava *dirty reads*)	—
READ COMMITTED (default)	Vidi samo one podatke koji su *committed* nakon svakog iskaza	Sprječava prljava čitanja (eng. *dirty reads*)
REPEATABLE READ	Vidi snapshot od početka izvođenja transkacije	Sprječava neponavljauća čitanja (eng. *nonrepeatable reads*)
SERIALIZABLE	Potpuno izvršavanje transakcija u seriji: najsigurnije i najsporije	Sprječava sve anomalije

Usklađenost s ACID-om

Svojstvo	Značenje	PostgreSQL implementacija
Atomicity	Sve naredbe u transkaciji se izvrše ili se nijedna ne izvrši	putem COMMIT / ROLLBACK
Consistency	Baza podataka radi tranziciju iz jednog stanja u drugo	putem ograničenja, okidača i transkacijskog integriteta
Isolation	Paralelne transakcije ne utječu jedna na drugu	kroz MVCC (Multi-Version Concurrency Control)
Durability	Jednom kad su podaci zapisani, možemo računati da su spremljeni na disku bez obzira na nepredviđene događaje	kroz WAL (Write-Ahead Logging)

Postavka u oblaku

- S prošlih vježbi trebali biste imati Supabase besplatnu instancu Postgresa u oblaku
  - Ukoliko nemate, napravite ju
  - Spojite se na bazu podataka koristeći DBeaver, DataGrip (osobna preporuka) ili neku od VS Code ekstenzija
    - Možete upite izvršavati i kroz Supabase web sučelje
- Istražite osnovne koncepte kroz podatke u sistemskim tablicama (spomenute na ovim i prijašnjim vježbama)
- Uspostavite konekciju i potom izvršavajte naredbe koristeći programski kod (preporuka C#/Npgsql)!

Docker Setup

- Postavljanje putem Dockera zahtjeva instaliran Docker ☺
  - Na Windowsu – WSL i sve ostalo kako je opisano vodičem
  - Detalji o Postgres *imageu* i svemu ostalom: https://hub.docker.com/_/postgres

```
docker run --name example-01 -e
POSTGRES_PASSWORD=mysecretpassword -d postgres
```


ALGEBRA

Sirova konekcija

- Koristit ćete je i u prvom projektu
- Bilo koja biblioteka koja **nije ORM implementacija**
- U primjerima na satu koristit će se Npgsql

Zadatak #1

- Istražite barem ove navedene sistemske tablice:
  - `pg_database`
  - `pg_stat_database`
  - `pg_stats`
  - `pg_stat_user_tables`
  - `pg_stat_activity`
  - `information_schema.tables`

Zadatak #2

- Izlistajte sve baze podataka i njihvov sadržaj koristeći
  - `pg_database`
    - `oid, datname`
  - `pg_stat_database`
    - `xact_commit`
    - `xact_rollback`

Zadatak #3

- Using PPPK-Vjezbe-01-Seed.sql datoteku koja se nalazi na IE kako biste postavili podatke u bazi
- Analizirajte definiciju tablica i sadržaj
- Saznajte sljedeće podatke:
  - Top 5 studenata po prosječnim brojem bodova
  - Najpopularniji ispiti (po broju prijava)
  - Postotak prolaznosti

Zadatak #4

- Umetnite podatke o dva studenta unutar transakcije
- Napravite pogrešku prilikom drugo umetanja (npr. dvostruki primarni ključ) i pokažite da se transakcija resetirala (eng. *rollback*)
- Potom izvršite istu transakciju kroz programski kod!

Task #5

- Napravite tablicu imena `sensor_reading` koja sadrži tri brojčane vrijednosti: `humidity`, `temperature`, `AQI` kao I primarni ključ
- Programatički stvorite CSV datoteku s 100000 nasumičnih očitanja senzora
- Umetnite podatke iz CSV datoteke u bazu koristeći:
  - `INSERT`
  - `COPY`

Što smo naučili?

- Postgres arhitekturu
- Ponovili i proširili znanje o SQL/PostgreSQL-u
- Kako se spojiti na instancu Postgresa
  - Potrebno je pronaći odgovarajuću biblioteku
  - Appropriate database connection library for langauge of your choice (e.g. C# - **Ngpsql**)
- Ne zaboravite proučiti projektni zadatak dostupan na IE!

Izazov Alpha

- Napravite klijentsku aplikaciju koja se spaja i koristi podatke (i metapodatke) iz PostgreSQL-a
  - po REPL (*read-evaluate-print loop*) principu
- Dodatno, implementirajte `\t <table_name>` I `\c <dat_name>` po uzoru na postojeće rješenje `psql` (REPL komandno-linijska aplikacija)
  - Probajte rekreirati što više tih specifičnih funkcionalnosti spomenutih na vježbama
- Obavezno formatirajte rezultat `SELECT` naredbi!!!
- Rok: **16. listopada** (sljedeći četvrtak)

ALGEBRA