Biblioteka za generiranje polustrukturiranih tekstnih datoteka

Tehnička dokumentacija

Verzija 1.0

Studentski tim: Ivan Derdić

Nastavnik: izv. prof. dr. sc. Alan Jović

Fakultet elektrotehnike i računarstva, Zagreb
18.01.2024.

Sadržaj

1 Opis proizvoda

Ovaj proizvod služi generiranju polustrukturiranih tekstnih datoteka. To su tekstne datoteke koje sadrže strukturirane podatke, ali ne u obliku koji je pogodan za strojno čitanje. Strukturirani podaci su tablice različitih shema. Ovaj proizvod služi za generiranje takvih datoteka s generiranim podacima. Podaci se generiraju na temelju funkcija definiranih od strane korisnika. Korisnik definira funkcije koje generiraju podatke za svaku kolonu tablice. Korisnik također definira i shemu tablice. Na temelju toga proizvod generira podatke za svaku kolonu tablice i sprema ih u datoteku/datoteke.

Ovaj proizvod ima implementirane sljedeću funkcionalnosti:

- generiranje velike količine podataka,
- mogućnost korisnika da definira funkcije za generiranje podataka,
- strukture potrebne za kreiranje različitih tablica,
- generiranje konačne sheme podataka u JSON formatu i
- dinamično generiranje količine podataka i datoteka.

Ovaj projekt je napisan u programskom jeziku Rust. Rust je programski jezik koji je nastao 2010. godine. Rust je nastao kao projekt u Mozilla Corporationu. Rust je programski jezik koji je dizajniran za sigurnost, performanse i paralelizam. Rust je statički tipiziran programski jezik. Rust je programski jezik koji je dizajniran za razvoj sustava. Rust je odabran zato što ima dobro dizajniran oblikovni obrazac iteratora koji je pogodan za generiranje podataka. Također postoji biblioteka Rayon koja omogućuje paralelizaciju iteratora. Korištenjem iteratora i biblioteke Rayon moguće je generirati velike količine podataka u kratkom vremenu.

2 Strukture biblioteke

Biblioteka sadrži tri strukture podataka koje su prikazane na slici 1. Struktura `Column` predstavlja jedna stupac tablice. Struktura `Table` predstavlja jednu tablicu. Struktura `ExportFile` predstavlja jednu datoteku koja sadrži podatke. Podaci se generiraju na razini retka tablice.

Struktura `Column` se sastoji od polja:

`name` koje predstavlja naziv stupca tablice,

`size` koje predstavlja veličinu vrijednosti stupca u bajtima,

`sql_type` koje predstavlja SQL tip podataka stupca i

`generator` koje predstavlja funkciju koja generira podatke za stupac.

Struktura `Table` se sastoji od polja:

`id_value` koje se koristi za identifikaciju tablice u generiranoj datoteci, također se moža razmatrati kao ime tablice,

`columns` koje predstavlja stupce tablice,

`delimiter` koje predstavlja znak koji se koristi za razdvajanje vrijednosti stupaca u generiranoj datoteci,

`percent_size` koje predstavlja postotak količiune podataka koji tablica zauzima u generiranoj datoteci i

`row_size_bytes` koje predstavlja veličinu jednog retka tablice u bajtima.

Struktura `Table` sadrži funkcije:

`generate_table_row` koja generira podatke za jedan redak tablice i

`generate_table` koja generira podatke za cijelu tablicu.

Funkcija *generate_table* koristi funkciju *generate_table_row* za generiranje podataka za svaki redak tablice. Funkcija *generate_table_row* koristi funkcije koje su definirane od strane korisnika za generiranje podataka za svaku kolonu tablice.

Struktura `ExportFile` se sastoji od polja:

`tables` koje predstavlja tablice koje se nalaze u datoteci,

`number_of_files` koje predstavlja broj datoteka koje se generiraju i

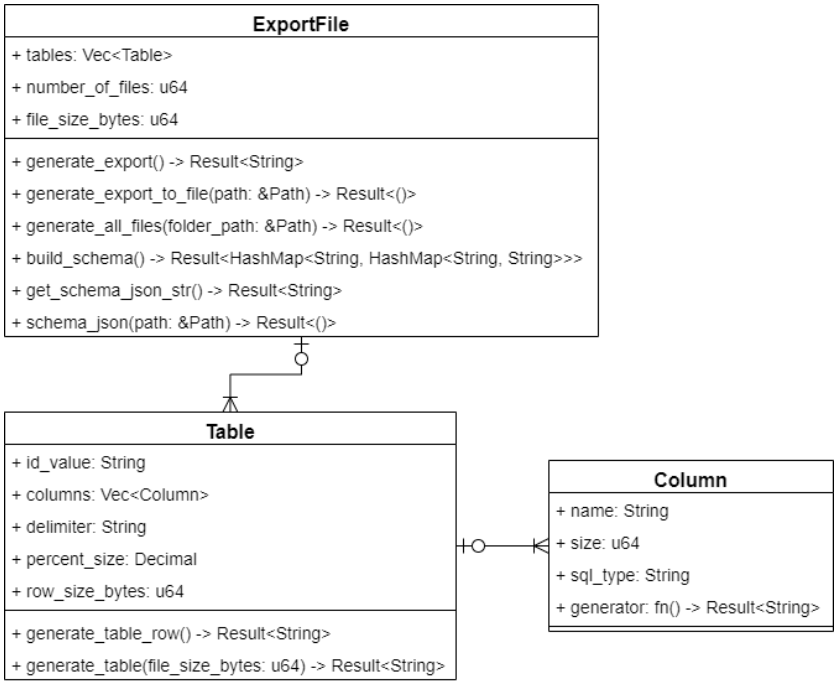
`file_size_bytes` koje predstavlja veličinu jedne datoteke u bajtima.

Prilikom kreiranja strukture `ExportFile` korisnik definira veličinu podataka koje želi generirati i broj datoteka u koje će se podaci spremiti. Temeljem toga se računa veličina jedne datoteke.

Struktura `ExportFile` sadrži funkcije:

`generate_export` koja generira podatke za jednu datoteku,

`generate_export_to_file` koja koristeći funkciju *generate_export* generira podatke za jednu datoteke te ih sprema u datoteku,



Slika 1: UML dijagram struktura biblioteke.

`generate_all_files` koja koristeći funkciju *generate_export_to_file* generira podatke za sve datoteke i sprema ih u datoteke,

`build_schema` koja generira shemu podataka kao `HashMap` strukturu podataka,

`get_schema_json_str` koja generira shemu podataka u JSON formatu i

`schema_to_json_string` koja koristeći funkciju `get_schema_json_str` generira shemu podataka u JSON formatu i sprema je u dateteku.

2.1 Ograničenja podataka u strukturama

Generatorske funkcije za strukturu `Column` moraju imati neovisna unutarnja stanja. To znači da se ne bi smijele koristiti globalne varijable ili varijable koje su definirane izvan funkcije. Razlog tome je što se generatorske funkcije pozivaju paralelno. Moguće je koristiti glocalne varijable ili varijable koje su definirane izvan funkcije ako se koristi mehanizam za sinkronizaciju pristupa tim varijablama. Osvisno o implementaciji tog mehanizma moguće je da će se performanse smanjiti. U najgorem slučaju cijeli program će se izvrašavati slijedno.

Za strukturu `ExportFile` vrijedi da je zbroj postotnih veličina tablica jedak jedan. Kako bi bilo moguće generirati shemu podataka unutar jedne tablice ne smije postojati više stupaca istog naziva. Također ne smije postojati više tablica s istim nazivom. Ukoliko nije potrebno generiranje sheme podataka moguće je imati više stupaca s istim nazivom i više tablica s istim nazivom. Tada će se u generiranoj datoteci pojaviti više stupaca s istim nazivom i više tablica s istim nazivom.

3 Korištenje biblioteke

Prvi korak u korištenju biblioteke je dodavanje biblioteke u datoteku `Cargo.toml`. Poterbno je dodati i biblioteku `rust_decimal`. Također preporučeno je dodati i biblioteku `Anyhow`. Biblioteka `Anyhow` služi za lakše rukovanje greškama. Datoteka `Cargo.toml` je datoteka u kojoj se nalaze informacije o projektu i ovisnostima projekta. Biblioteke se dodaju u odjeljak `dependencies`. Primjer dodavanja biblioteke u datoteku `Cargo.toml` prikazan je na primjeru programskog koda 1.

Programski kod 1: Odjeljak `dependencies` u `Cargo.toml` datoteci s potrebnim bibliotekama.

```
[dependencies]
diplomski_projekt = { git = "https://github.com/IvanDerdicDer/diplomski_projekt" }
anyhow = "1.0.75"
rust_decimal = "1.32"
```

Nakon dodavanja biblioteke u datoteku `Cargo.toml` potrebno je uvesti biblioteku u programski kod. Primjer uvoza biblioteke u programski kod prikazan je na primjeru programskog koda 2.

Programski kod 2: Uvoz biblioteke u programski kod.

```
use diplomski_projekt::*;
use anyhow::Result;
use rust_decimal::Decimal;
```

Nakon uvoza biblioteke u programski kod potrebno je definirati funkcije koje generiraju podatke za svaku kolonu tablice. Primjer definiranja funkcije za generiranje podataka za stupac tablice prikazan je na primjeru programskog koda 3.

Programski kod 3: Jednostavna funkcija za generiranje podataka stupca tablice.

```
fn column_generator() -> Result<String> {
    Ok("1".to_string())
}
```

Sljedeći korak je kreirati strukture `Table` sa stupcima tablice i strukturu `ExportFile` sa tablicama. Primjer kreiranja strukture `Table` i strukture `ExportFile` prikazan je na primjeru programskog koda 4.

Programski kod 4: Kreiranje strukture `Table` i strukture `ExportFile`.

```
let table = Table::new(
    "TEST_TABLE".into(),
    vec![Column::new(
        "test_column".into(),
        1,
        "INT".into(),
        column_generator,
    )],
    "|".into(),
    Decimal::from(1),
);
```