Praktikum iz objektno orijentisanog programiranja (13S112POOP)
Projektni zadatak – C++

Napisati skup klasa sa odgovarajućim metodama, konstruktorima, operatorima i destruktorima za realizaciju softverskog sistema za simulaciju rada sa bazama podataka.

Korisnik (naručilac) softvera, želi da softver pruži sledeće funkcionalnosti:

- Interakcija sa korisnikom putem tekstualnog menija (konzola) ili grafičkog korisničkog interfejsa
- Kreiranje baze podataka
- Izvršavanje upita
    - Kreiranje/brisanje tabela – DDL (CREATE TABLE, DROP TABLE)
    - Izmene podataka - DML (SELECT, INSERT, UPDATE, DELETE)
- Eksportovanje baze podataka
- Kraj rada

Za uspešno rešenje zadatka potrebno je izvršiti analizu zahteva. Kao rezultat analize, potrebno je dopuniti i precizirati funkcionalnu specifikaciju softverskog alata. Na osnovu specifikacije, potrebno je napisati sistem klasa u jeziku C++ koje realizuju traženi softver. U nastavku su navedeni neki elementi specifikacije. Od studenata se očekuje da dopune one stavke koje nisu dovoljno precizno formulisane, odnosno dodaju nove stavke (tamo gde to ima smisla) ukoliko uoče prostor za unapređenje. Izmene i dopune specifikacije mogu da donekle odudaraju od zahteva naručioca softvera u onoj meri u kojoj to neće narušiti traženu funkcionalnost. Takođe, priloženi UML dijagram koji opisuje zahtevani softver se ne mora obavezno poštovati, već samo predstavlja skicu potencijalnog rešenja. Prilikom izrade specifikacije voditi računa o potencijalnom unapređenju softvera na osnovu naknadnih zahteva.

Prilikom izrade rešenja, od studenata se očekuje intenzivno korišćenje svih onih mogućnosti koje pružaju specifikacija jezika C++ i biblioteka STL, kao što su šablonske funkcije, kolekcije, algoritmi, regularni izrazi, iteratori, lambda izrazi i sl. **Rešenja koja ne vode računa o ovom aspektu neće moći da dobiju maksimalan broj poena.** Takođe, voditi računa o **objektno orijentisanom dizajnu rešenja**, čistoći, čitkosti i komentarisanju programskog koda.

Funkcionalna specifikacija

U nastavku je zadat deo korisničkih zahteva koje treba razraditi i, po potrebi, dopuniti tako da se dobije funkcionalna aplikacija.

Interakcija sa korisnikom

Korisnik može da interaguje sa programom bilo izborom odgovarajućih opcija iz tekstualnog menija putem tastature ili izborom u datom trenutku dostupnih opcija putem grafičkog korisničkog interfejsa. Nije potrebno realizovati oba načina. Interakcija u slučaju grafičkog interfejsa može da se vrši putem tastature ili miša. U zavisnosti od izabrane opcije i njenih parametara, program izvršava zadatu opciju ili ispisuje poruku greške. Poruka greške treba da bude što je moguće detaljnija da bi korisniku pomogla da grešku otkloni. Sve eventualne parametre koji su potrebni prilikom rada aplikacije je potrebno zatražiti od korisnika. Ukoliko korisnik ne zada ništa, koristiti vrednosti fiksirane u programu.

Pokretanje programa

Prilikom pokretanja programa korisniku se nudi da kreira novu bazu podataka ili da učita postojeću. Potrebno je definisati format koji bi podržao čuvanje svih podataka iz baze podataka. Za parsiranje i generisanje fajlova u ovom formatu **nije** dozvoljeno korišćenje gotovih biblioteka.

Osnovna manipulacija

Korisniku putem menija ili grafičkog korisničkog interfejsa treba dozvoliti sledeće opcije po učitavanju baze podataka:

- Izvršavanje upita
- Eksportovanje baze podataka
- Napuštanje programa

Izvršavanje upita

Korisnik upite zadaje u tekstualnom formatu. Upit je potrebno parsirati i izvršiti nad trenutnom bazom podataka. U slučaju sintaksne greške prilikom parsiranja upita potrebno je ispisati odgovarajuću poruku koja obavezno mora sadržati informaciju o tome gde je do greške došlo. Do greške može doći i prilikom izvršavanja ispravno napisanog upita i u tom slučaju je korisniku potrebno dati detaljno objašnjenje u kom trenutku i zašto je došlo do greške. Primer: selektuje se kolona koja ne postoji u zadatoj tabeli – korisniku treba ispisati koja kolona i u kojoj tabeli ne postoji. Prilikom parsiranja upita potrebno je ignorisati sve blanko znakove, a upit je opciono moguće završiti znakom **;**.

Potrebno je podržati sledeće naredbe:

- **Naredba za kreiranje nove tabele**
  Naredba je oblika: CREATE TABLE *naziv_tabele* (*lista_kolona*)

  lista_kolona = *lista_kolona* **,** *naziv_kolone*
  lista_kolona = *naziv_kolone*

  Pretpostaviti da su sve kolone tekstualnog tipa i da se nazivi kolona prilikom navođenja uvek odvajaju znakom zapeta. Greška je ukoliko tabela sa zadatim nazivom već postoji u tekućoj bazi podataka. Naziv tabele se sme sastojati samo od malih i velikih slova alfabeta.

- **Naredba za brisanje tabele**
  Naredba je oblika: DROP TABLE *naziv_tabele*
  Greška je ukoliko tabela sa zadatim nazivom ne postoji u tekućoj bazi podataka.

- **Naredba za čitanje**
  Naredba je oblika: SELECT *selektor* FROM *naziv_tabele* [WHERE *lista_uslova*]

  selektor = ***** | *lista_kolona*
  lista_uslova = *lista_uslova* AND *uslov*
  lista_uslova = *uslov*
  uslov = *naziv_kolone* **=** "*vrednost*"
  uslov = *naziv_kolone* **<>** "*vrednost*"

  Izvršavanjem ove naredbe dobijaju se podaci iz zahtevanih kolona odabrane tabele i to za one zapise koji ispunjavaju zadate uslove. Moguće je navesti spisak kolona koje je potrebno prikazati ili iskoristiti znak ***** kojim se selektuju sve kolone tabele. Zapise je moguće filtrirati korišćenjem opcione *WHERE* klauzule. U tom slučaju zadaje se lista uslova odvojena tekstom *AND*. Potrebno je podržati proveru na jednakost i nejednakost.

  BONUS 10 POENA:
  Implementirati spajanje tabela pomoću INNER JOIN.

  Primer:
  SELECT p.name, c.name
  FROM Person p
  INNER JOIN Country c ON p.countryID = c.ID
  WHERE c.continent = 'Europe';

- **Naredba za upis**
  Naredba je oblika: INSERT INTO *naziv_tabele* (*lista_kolona*) VALUES *lista_zapisa*

  lista_zapisa = *lista_zapisa* **,** *zapis*
  lista_zapisa = *zapis*
  zapis = **(** *lista_vrednosti* **)**
  lista_vrednosti = *lista_vrednosti* **,** "*vrednost*"
  lista_vrednosti = "*vrednost*"

  Izvršavanjem ove naredbe upisuju se zadati podaci u zadatu tabelu. Sve kolone tabele obavezno moraju biti navedene u listi kolona, a *NULL* vrednosti nisu podržane.

- **Naredba za ažuriranje**
  Naredba je oblika: UPDATE *naziv_tabele* SET *lista_izmena* WHERE *lista_uslova*

  lista_izmena = *lista_izmena* **,** *izmena*
  lista_izmena = *izmena*
  izmena = *naziv_kolone* **=** "vrednost"

  Izvršavanjem ove naredbe navedene izmene se sprovode za sve zapise zadate tabele koji ispunjavaju zadate uslove.

- **Naredba za brisanje**
  Naredba je oblika: DELETE FROM *naziv_tabele* WHERE *lista_uslova*

  Izvršavanjem ove naredbe brišu se svi zapisi zadate tabele koji ispunjavaju zadate uslove.

- **Naredba za prikaz liste postojećih tabela**
  Naredba je oblika: SHOW TABLES

Eksportovanje baze podataka i kraj rada

Korisnik može da zahteva da se baza podataka eksportuje pri čemu je potrebno da zada putanju destinacionog fajla. Potrebno je podržati eksportovanje u sledeće formate:

- Format koji studenti samostalno definišu. Ovako eksportovanu bazu podataka je kasnije moguće učitati prilikom pokretanja programa.
- SQL format. Potrebno je bazu podataka eksportovati u .sql fajl koji se sastoji od spiska SQL naredbi za kreiranje svih tabela, a zatim i spiska SQL naredbi kojim bi se svi postojeći podaci upisali u odgovarajuće tabele.

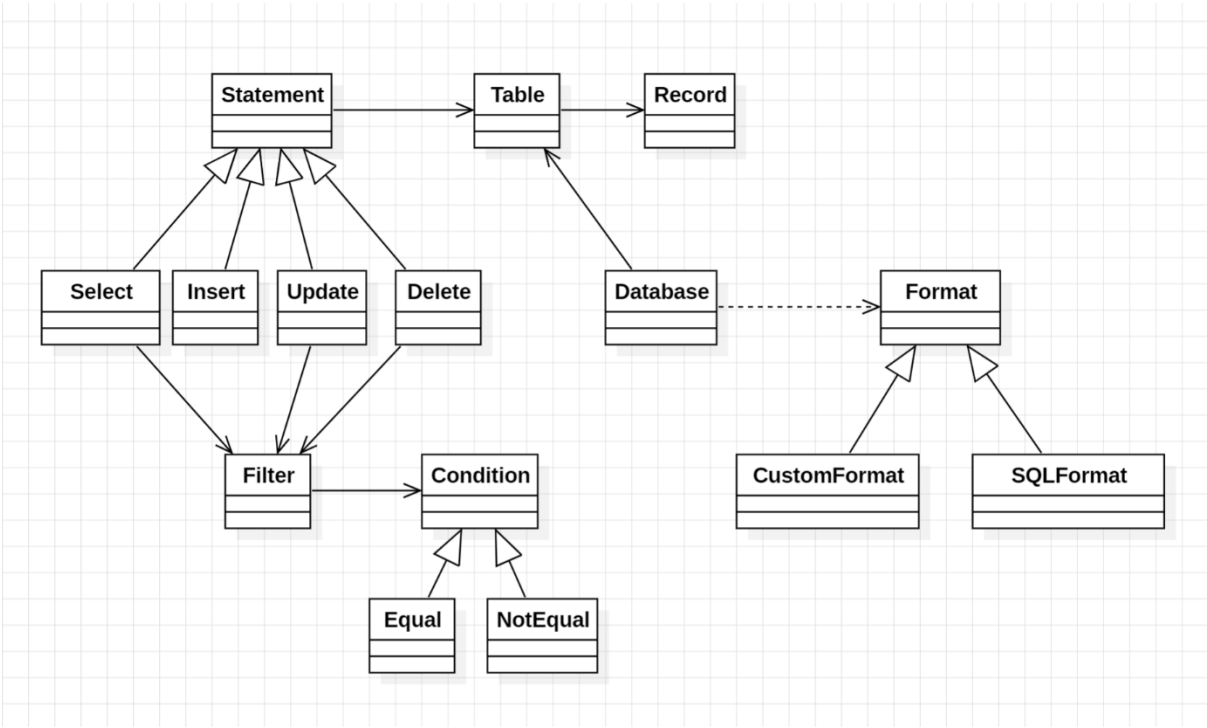
Korisnik može da zahteva kraj rada programa. Od korisnika se traži potvrda za napuštanje programa. Ukoliko korisnik nije sačuvao bazu podataka koja je trenutno otvorena potrebno mu je ponuditi da to uradi pre napuštanja programa.

Testiranje rada programa

Prilikom testiranja rada programa moguće je koristiti postojeće alate za rad sa bazama podataka. Baza podataka napravljena u ovom alatu se može sačuvati u .sql fajlu. Naredbe iz ovog fajla je moguće izvršiti u većiti alata za rad sa bazama podataka i time se može proveriti ispravnost rada.

Dijagram klasa

Na osnovu prethodne funkcionalne specifikacije formiran je dijagram klasa koji je prikazan na slici 1. Dijagram klasa nije detaljan, te ga treba tumačiti kao skicu koja načelno ukazuje na arhitekturu softvera. Studenti mogu da koriste ovaj dijagram kao referencu i, po potrebi, prošire ga da bi ga usaglasili sa eventualnim dopunama specifikacije.



Slika 1. Dijagram klasa

Prilikom implementacije rešenja, **obratiti pažnju na objektno orijentisani dizajn** i intenzivno koristiti kolekcije i algoritme standardne biblioteke jezika C++ i lambda funkcije gde god je to moguće.

Bodovanje projekta

- Osnovne funkcionalnosti (kreiranje/brisanje/prikaz tabela): 5 poena
- SELECT naredba: 5 poena
- INSERT naredba: 5 poena
- UPDATE naredba: 5 poena
- DELETE naredba: 5 poena
- Eksportovanje i učitavanje iz samostalno definisanog formata: 5 poena
- Eksportovanje u SQL format: 5 poena
- INNER JOIN: **BONUS 10 poena**

Na broj osvojenih poena utiče i:

- korišćenje mogućnosti koje pruža STL (šablonske funkcije, kolekcije, algoritmi, regularni izrazi, iteratori, lambda izrazi)
- objektno orijentisani dizajn rešenja,
- čistoća, čitkost i komentarisanje programskog koda