TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kristjan Henri Roots
213450IACB

Kodune ülesanne 2

Programmeerimine II
(IAX0584)

Juhendaja: Lebit Jürimägi
Magister

Tallinn 2021

Autorideklaratsioon

Kinnitan, et olen koostanud antud kodutöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Kristjan Henri Roots

08.05.22

Sisukord

Jooniste loetelu

1 Ülesande püstitus

Käesoleva kodutöö ülesandeks oli luua andmebaasi rakendus kasutades C keeles programmikoodi ning SQL skripti tabelite struktruuri ja andmetega. Lahendatav variant 2 määras ülesandeks luua filmiarvustuste portaali. Soovituslikult olid vajalikud andmed jagatud nelja tabelisse: Režissöörid, Filmid, Arvustajad ning Arvustused.

Loodud rakendus pidi olema võimeline lahendama järgmisi ülesandeid:

- Kuvama sisse loetud algandmeid.
- Võimaldama režissööri lisamist läbi kasutajaliidese.
- Lisama, muutma ning kustutama filmide ning arvustustega seotud infot.
- Otsima filme väljalaske aasta alusel.
- Kuvama viis parimalt hinnatud filmi kasutajate ning kriitikute meelest.
- Kuvama publikatsioonide keskmised hinded iga aasta kohta.

2 Andmemudeli kirjeldus

Tabelite koostamisel võeti aluseks soovituslik andmemudel, kuid lisati ka juurde mõni tabel rakenduse huvitavamaks tegemise eesmärgil. Loodud lahenduses leidub seitse tabelit ning lahenduse SQL skript on loodud kasutades PostgreSQL'i pgAdmin 4 tarkvaras. SQL serverina kastutati personaalset arvutit (vt Joonis 12, lk 24).

Tabeliandmete identifitseerimiseks otsustati kasutada primaarvõtmena *serial* andmetüübiga identifikatsiooni numbrit (järgnevalt kasutusel lühendina *ID*), mis kasvab andmete lisamisel automaatselt ning on lokaalses tabelis unikaalne. Kõik välisvõtmed on seatud kustutamise korral *Cascading* tüüpi, mis tähendab, et näiteks filmi kustutamisel kustutatakse ka sellega seotud tegelased, žanrid ning arvustused.

Loodud rakenduses kasutatavate filmide andmed ning arvustused pärinevad veebilehekülgedelt IMDb[1] ning Metacritic[2].

2.1 Filmi andmed

Filmi andmete jaoks on soovitustele vastavalt loodud eraldi tabel *movies*, kuid tabeli tulbad on mõnevõrra keerulisemad. *Movies* tabelis leiduvad tulbad *ID*, filmi nime, aasta, kestvuse (minutites), stuudio, keele, vanusepiirangu, IMDb keskmise hinde ning lühikirjeldusega.

Piirangutena peavad tabelis olema unikaalsed filmide nimed ning IMDb keskmise hinde korral on kasutusel andmetüüp *numeric* pikkusega 3 ning täpsusega 1, mis võimaldab hindeid 0.0 kuni 10.0.

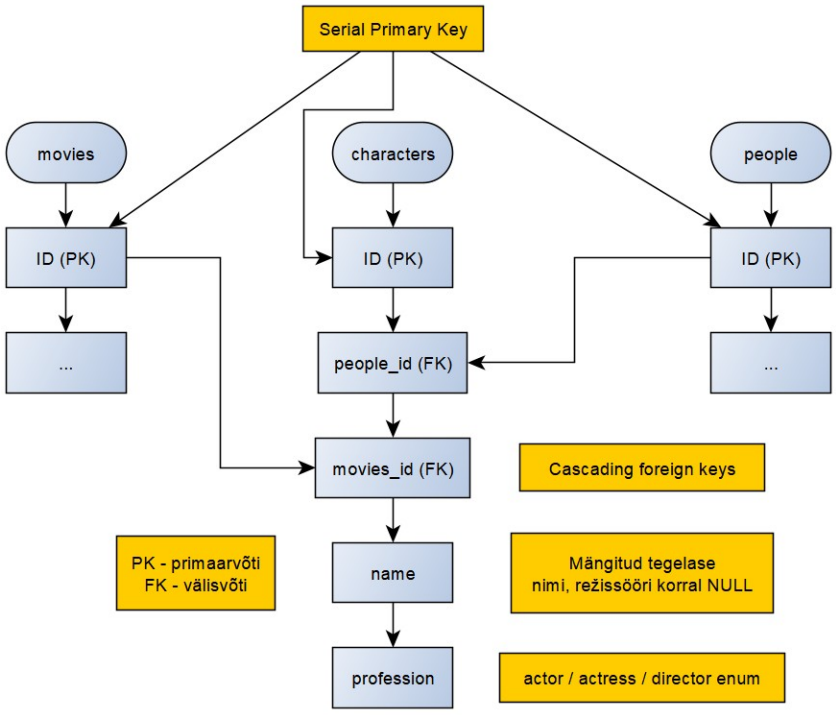
Soovituslikust tabeli struktuurist on välja jäetud filmi žanr ning režissöör. Mõlemale neist on loodud hoopiski keerulisem struktuur.

2.2 Filmiga seotud isikud

Filmiga seotud isikute andmeid hoitakse eraldi filmide ning isikute andmetest tabelis *characters,* mis seob andmebaasis oleva filmi tabelis *movies* isikuga tabelis *people* (vt Joonis 1, lk 7). Lisa tulpadena leiduvad tabelis ka näitleja mängitud tegelase nimi ning roll filmis, mis on seadistatud andmebaasis *enum* tüübiga, mille väärtusteks on *actor, actress* või *director.* Otsus lisada filmidele juurde ka tegelased ning näitlejad oli puhtalt vabatahtlik.

Piirangutena on seatud unikaalseteks tegelase nimi, seotud film, isik ning roll filmis. Antud lahendus võimaldab samal isikul mängida filmis mitut rolli, selleks isikuks võib olla ka kasvõi filmi režissöör.

Isikud ise on tabelis *people*, mis sisaldab vaid *ID*'d, nime ning rahvust.



Joonis 1. *Characters* tabeli struktuur.

2.3 Seotud žanrid ning arvustused

Žanrite ning arvustuste jaoks loodud lahendused on struktuurilt sarnased tegelaste tabeliga ehk viited filmile ning žanrile või isikule on tabelis välisvõtmetena.

Arvustuste jaoks loodud tabel *review* hoiab peale viidete ka filmi arvulist hinnangut, samuti *numeric* andmetüübina, ja ka lühikest tekstilist hinnangut. Filmi žanrite jaoks on loodud tabel *movie_genres,* mis hoiab vaid välisvõtmeid. Mõlemas tabelis peavad olema unikaalsed viited filmile ning žanrile või isikule, samal filmil ei tohi olla mitu hinnangut sama isiku poolt.

Žanrite jaoks eraldi loodud tabel *genres* sisaldab lihtsalt *ID*'d ning žanri nime, arvustajate tabel *reviewer* lisaks nimele ka publikatisooni (tavakasutaja puhul tühi).

3 Lahenduse kirjeldus

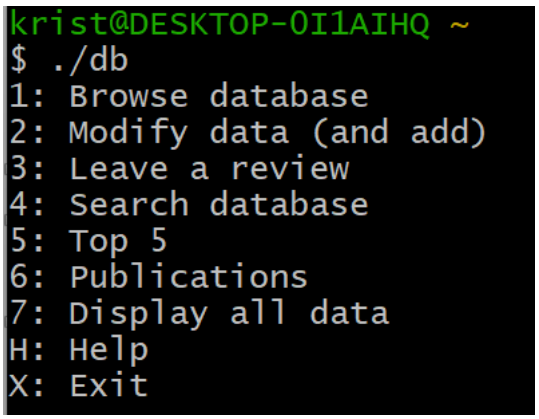
Vabatahtlikult lisatud keerulisus nii andmestruktuuri kui ka lisatud vidinate poolest läks lõpplahenduses juba pisut "üle käte". Valmis lõpplahendus sisaldab endas üle 1500 rea koodi ning 25 alamprogrammi, mille tõttu keskendun kirjeldamisel vaid kõige olulistematele hetkedele ning lähtun püstitatud nõuetest.

Lahenduse mahukuse tõttu oli suur rõhk programmikoodi kommenteerimisel ning lahti seletamisel. Lõpplahenduses on loodud igale suurele alamfunktioonile eelnev kommentaar, mis seletab lahti eesmärgi ning sisendmuutujad. Muidugi on alamprogrammid kommenteeritud ka siseselt.

3.1 Põhiprogramm

Kasutajaga suheldakse läbi menüüde kogumiku; *main* funktsiooni ülesandeks on luua ning lõpetada ühendus andmebaasidega ning suunata kasutaja õigete funktsioonide juurde (vt Joonis 2, lk 10). Ühendust andmebaasis hoitakse loodud andmestruktuuris *db_status_t DB,* mis sisaldab ka kõikide andmebaasis leiduvate tabelite nimesid, mida läbi *enum*iga indekseerides alamfunktsioonidele edastatakse. Antud lahendus annab SQL käskudele modulaarsuse ning väldib igale tabelile erineva käskluse manuaalset kirjutamist.

Lisa nõudena oli vajalik funktsioonis kasutada dünaamilisi massiive, kuid enamustel olukordadel oli käskluste hoidmiseks vaja vaid puhvreid, mis said loodud staatilisteks. Programmi loomisel on ka eeskuju võetud 10. nädala tunniülesandest *prod41.c*, millest ka esimesed viis alamprogrammi pärinevad.



Joonis 2. Rakenduse avamenüü.

3.2 Andmete kuvamine

Funktsionaalsetest nõuetest oli rakendusel vajalik vaid kuvada kõik sisseloetud informatsiooni, mida teeb ka alamfunktsioon *printAll* mõne käsklusega, kuid palju huvitavamalt on loodud alamfunktsioon *display_movie_list,* mis suudab sisse antud käsklusest moodustada interaktiive nimekirja. Funktsiooni aluseks oli *prod41.c* funktisoon *display*, millele lisati interaktiivsus.

Sisendkäsklusena nõuab funktsioon viimase tulbana filmi *ID*'d, mida kasutajale ei kuvata. Filmid on paigutatud järjestikuliste numbritega nimekirja sorteeritud keskmise hinnangu alusel, millest filmi valides pöördutakse omakorda funktsiooni *main_movie_view* poole, mille ülesandeks on valitud filmi põhjaliku ülevaate kuvamine.

Funktsioon näitab kasutajale kõiki filmiga seotud andmeid: režissööri, tegelasi ning näitlejaid, žanried ja ka arvustusi. Andmete puudumisel on vastav lahter lihtsalt tühi (vt Joonised 3 – 5, lk 11).

Loodud lahenduse eeliseks on jällegi modulaarsus, sest funktsioone kasutatakse lihtsalt kogu filminimekirja kuvamisel ning ka andmebaasi otsingu vastusena, mida on kirjeldatud detailsemalt vastavas peatükis. Lõpp tulemusena on lahendus igati efektiivne ja võib olla isegi mõne voogedastusteenusele sarnane.

```
    name                              | year                              | avg_rating                        |
 1. Tulnukas                          | 2006                              | 10.1                              |
 2. Back to the Future                | 1985                              | 10.0                              |
 3. Back to the Future Part II        | 1989                              | 9.0                               |
 4. The Truman Show                   | 1998                              | 8.8                               |
 5. Interstellar                      | 2014                              | 8.7                               |
 6. The Mask                          | 1994                              | 8.6                               |
 7. The Batman                        | 2022                              | 7.1                               |
 8. El Camino                         | 2019                              | 7.1                               |
 9. Ace Ventura: Pet Detective        | 1994                              | 6.8                               |
10. Back to the Future Part III       | 1990                              | 6.5                               |
11. Yes Man                           | 2008                              | 5.8                               |
12. The Room                          | 2003                              | 1.0                               |

To see more data about the film select the film number or any other number to return to the main menu
Select movie:
```

Joonis 3. Filminimekirja avavaade.

```
Select movie: 2
Info:
name                  : Back to the Future
year                  : 1985
duration_in_minutes   : 116
studio                : Universal Pictures
director              : Robert Zemeckis (American)
language              : English
age_rating            : PG
imdb_score            : 8.5
```

Joonis 4. Filmi lähivaade.

```
Genres: Adventure, Sci-Fi, Comedy
Characters:
            Marty McFly played by Michael J. Fox          (Canadian)
         Dr. Emmet Brown played by Christopher Lloyd      (American)

Reviews:
Kristjan Henri Roots
admin
10.0
One of my favorite trilogies ever, timeless.
```

Joonis 5. Filmiga seotud andmed.

3.3 Andmehaldus

Andmehaldus on loodud rakenduses vägagi keeruline, võimaldades kasutada ka rekursiooni. Käsklused serverile on antud enamjaolt *PQexec* kasutusel, mis oli igati mugav, kuid korrektsem oleks olnud anda parameetreid eraldi *PQexecParams* abil. Mõne lihtsama pahatahtliku käskluse eest päästab *PostgreSQL* ise, kuid täieliku kaitse jaoks tuleks rakendus teha ümber (vt Joonis 13, lk 24).

Andmehalduse põhivastutus on alamfunktsioonil *traffic_officer*, mis sisendmuutujate abil teeb selgeks, kas andmeid on vaja kirjutada, kustutada või uuendada, millise tabeliga on tegemist, ning pöördub vastava andmetöötlusega tegeleva funktsiooni poole. Alamfunktsiooniga suhtleb ka kasutaja soove edastav *modify_data* ning ka muud andmetega tegelevad funktsioonid ise ilma kasutaja sekkumiseta.

3.3.1 Andmete kirjutamine

Andmete lisamine andmebaasi on jagatud kolme alamfunktsiooni *create_movie, create_with_single_named* ning *linked_data* vahel. Kõigis neis kasutatakse kasutaja sisendi hoidmiseks dünaamilist indeksite massiivi *params*, mis samuti *prog41.c*'st pärit on.

Esimene funktsioonidest nagu nimigi ütleb tegeleb vaid filmide lisamisega. Sisendi kontroll kehtib vaid filmi nime ning sisestatud aasta korral, nimi peab olema unikaalne ning mitte tühi ja aasta ei tohi olla negatiivne ega ületada täisarvu maksimumi *INT_MAX* teegist *limits.h*.

Teine alamfunktsioon neist, *create_with_single_named,* kutsutakse esile tabelite korral, mis ei sisalda välisvõtmeid ehk žanrid, inimesed ja arvustajad. Kõigis nimetatud tabelites on esimeseks väärtuseks nimi, mida saab sisestada ühtselt ning kontrollida unikaalsust. Inimeste ja arvustajate korral on veel vajalik rahvuse ning publikatsiooni sisestamine, mille järgselt koostatakse vajalikud SQL päringud.

Viimane funktsioon *linked_data* on keeruliseim andmeid haldav funktsioon, kuna tegeleb välisvõtmetega tabelitega nagu konkreetsed filmiga seotud tegelased, žanrid ning arvustused. Andmete lisamiseks on vajalik teada filmi ning sisendist sõltuva teise tabeli *ID*'sid, millele küsida ka juurde andmeid sõltuvalt sisendist ning teha unikaalsuskontroll (vt Joonis 1, lk 7). Kokku on funktsioonil kaheksa sisend väärtust, mille tõttu ei ole lõplikus rakenduses realiseeritud seotud andmete uuendamine, ainult lisamine ning kustutamine on lubatud.

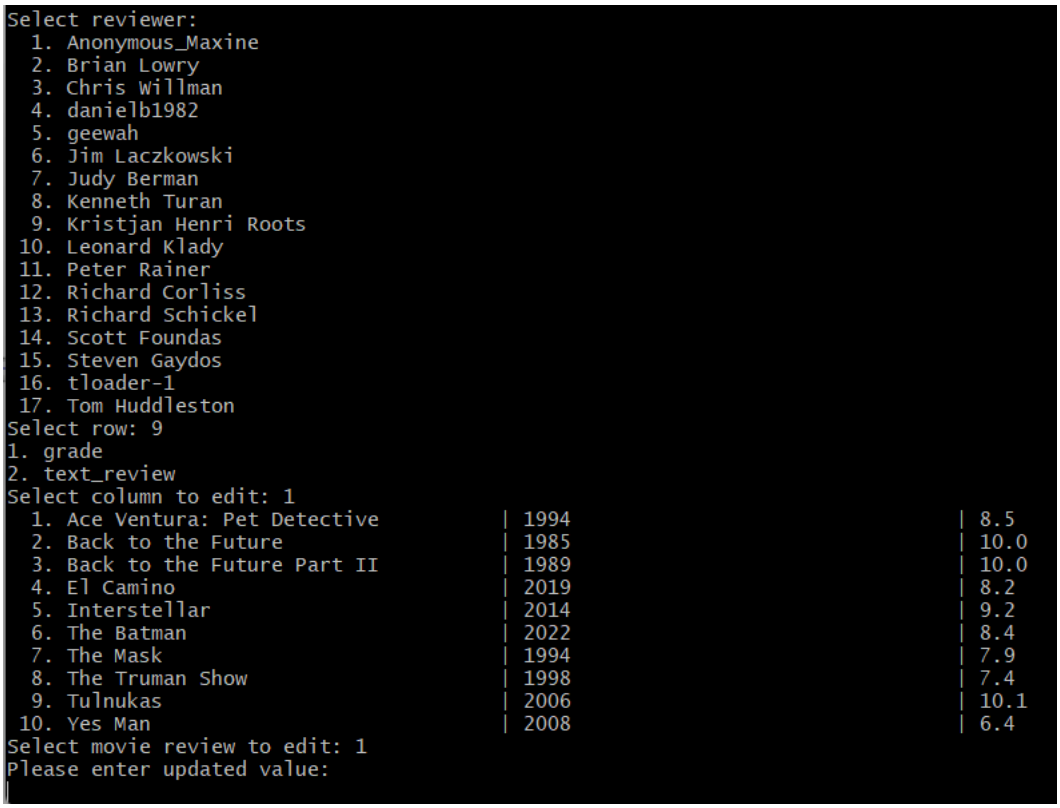
3.3.2 Andmete uuendamine

Andmete uuendamine käib rakenduses funktsioonis *update_db_value,* mis läbi sisendina saadud tabeli tüübina teeb kindlaks, millist tulpa kasutaja soovib muuta ning pöördub omakorda saadud infoga andmete loomise funktsioonide poole.

Nüüd on kasutusel aga lisamuutujana massiiv new_value. Sisendmuutujate abil on võimalik andmete loomise funktsioonidele ette öelda täpselt millise tulba väärtus on vajalik ning funktsioonide vastena kopeeritakse loodud massiivi kasutaja antud uus väärtus.

Lahenduse eelis on kompaktsus. Andmete kirjutamise massiivid on küll keerulisemad, kuid uuendamise programm ei vaja iga tabeli tüübi ning tulba jaoks eraldi käsklusi. Ilma andmete loomise funktsioonide poole pöörumiseta peaks uuendamise programm samuti teadma, millistele tabeli tulpadele kehtivad millised nõuded ja piirangud, funktsioonide poole pöördumine säästab identse andmete kontrolli.

Erandina on aga loodud eraldi käsud arvustuste jaoks, mida luuakse funktsiooniga linked_data, mille täielik ümber kirjutamine uuendamise jaoks oleks osutunud liiga ajamahukaks. Läbi spetsiaaljuhuks kirjutatud koodijupi on kasutaja võimeline muutma arvustusel antud hinnangut ning lühihinnangut teksti kujul. Esmalt küsitakse kasutajalt arvustaja ning seejärel kuvatakse kasutajale kõik filmid, millele valitud arvustaja hinnangu jätnud on (vt Joonis 6, lk 14). Andmete puudumisel teavitatakse kasutajat ning naastakse peamenüüsse.



Joonis 6. Arvustuse uuendamine.

3.3.3 Andmete kustutamine

Andmete kustutamine käib võrdlemisi lihtsa, kuid sellegipoolest pika funktsiooni *database_delete* kaudu. Funktsiooni sisendiks on tabeli tüüp kus kohast kasutaja soovib väärtuse kustutada ning kõikide erinevate variantide poole pöördumiseks on kasutusel lihtne *switch case* iga erineva tabeli tüübiga.

Isikute ning filmiga seotud tegelaste korral tuleb kasutajat hoiatada, sest ülesandele seatud nõudena peab olema igal filmil režissöör ning isiku kustutamisel kustutatakse ära ka kõik filmid, milles on ta seotud režissöörina. Filmiga seotud tegelase kui režissööri korral kustutatakse ära vaid seotud film, *cascading* tüüpi välisvõtmete tõttu kustutatakse seotud andmed ära ise. Režissööri olemasolu tehakse kustutamisel kindlaks funktsiooni *is_director* abil, mis tagastab väärtuse 1 juhul, kui tegemist on režissööriga ning 0, kui ei ole.

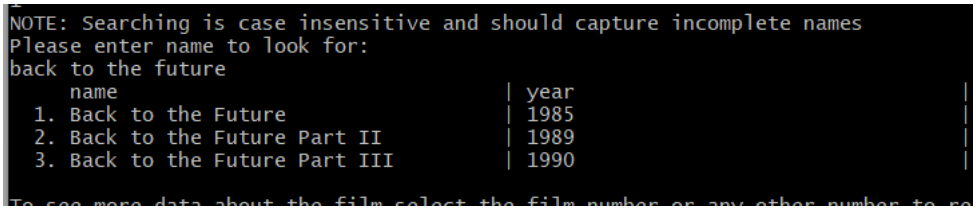
3.4 Andmebaasi otsing

Seatud nõudena pidi rakendus olema võimeline otsima andmebaasist filme aasta alusel ning filmid kuvama keskmise hinnangu põhjal. Lõplikus lahenduses on peale aasta järgi otsingu realiseeritud veel otsing filmi nime, näitleja, režissööri, arvustaja, žanri ning tegelase järgi. Veelgi enam on enamustel kategooriatel veel võimalik valida tekstiotsingu ning nimekirja vahel.

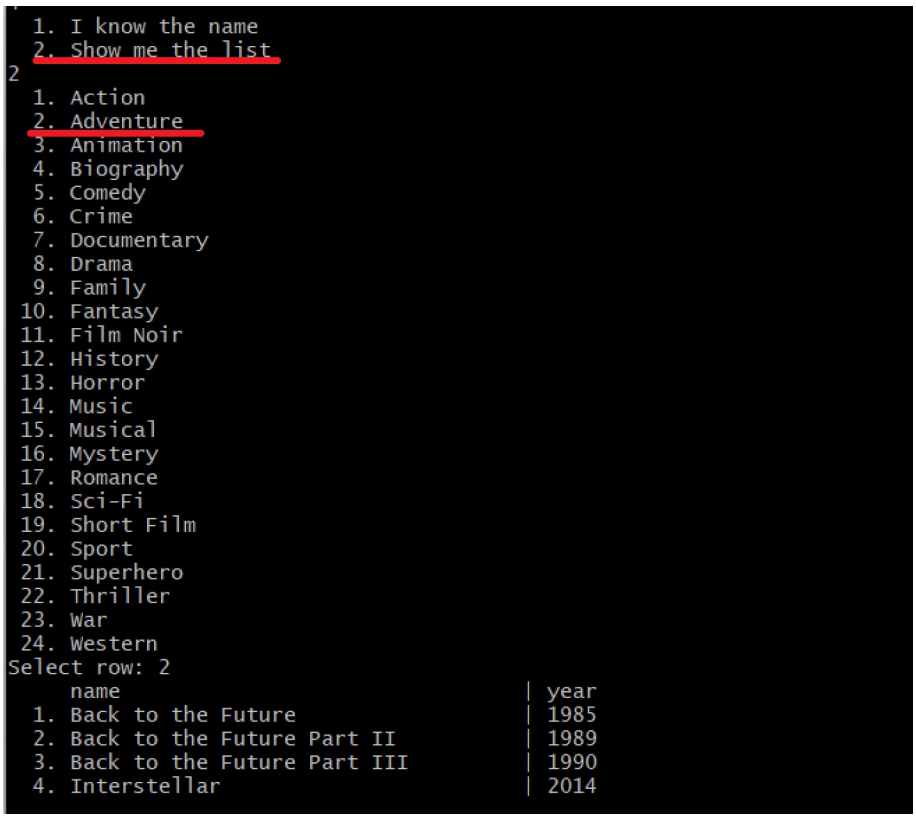
Tekstiotsing on realiseeritud *ILIKE* käsuga, mis teeb suurte tähtede olemasolu ebaoluliseks ning otsitav termin on ka paigutatud protsentide vahele, mille tõttu suudab rakendus leida ka ainult osaliselt õige termini korral korrektseid vasteid (vt Joonis 7, lk 15).

Kasutajal on võimalik tekstiotsingu asemel valida ka nimekiri, mille korral kuvatakse kõik andmebaasis leiduvad terminid. Näiteks žanri järgi otsides kuvatakse kasutajale kõik žanrid, mille seast saab kasutaja valida rea numbri alusel ning otsing toimub sel juhul termini *ID,* mitte nime järgi (vt Joonis 8, lk 16).

Vastete kuvamiseks on kasutusel juba tuttav funktsioon *display_movie_list,* mis teeb vastetest interaktiivse nimekirja ning filmi rea numbri sisestades näidatakse kasutajale jällegi filmi andmeid detailselt.



Joonis 7. Filmi nime järgi otsing.



Joonis 8. Žanri järgi otsing nimekirjast.

3.5 Viis parimat ning publikatisoonide keskmine

Võrreldes ülejäänud loodud programmiga olid viimase paari nõude täitmine kerge töö. Nõuetena pidi rakendus väljastama viis parimat filmi kasutajate ning kriitikute arvates. Eesmärgi saavutab alamprogramm top_five, mis küsib kasutajalt, kas paremik väljastatakse kasutajate või kriitikute arvamuse järgi ning seejärel koostatakse SQL käsklus liites kokku filmide tabeliga arvustused ning arvustajad, kes on vastavalt sisendile kriitikud või kasutajad ning pöördutakse funktsiooni *display_movie_list* poole (vt Joonis 9, lk 17).

Viimase nõudena väljastab rakendus publikatsioonide keskmised hinded aasta vaates. Just nii käsklust luues on aga ridade arv üsnagi suur, kuna valitud filmid on välja tulnud enamalt eri aastatel. Lahendusena otsustati väljastada üleüldised keskmised ning selle kõrvale veel publikatsiooni poolt hinnatud filmide arv (vt Joonis 10, lk 17). Saadud tulemus ei ole küll täpne vaste nõuetele, mis tõttu on lisades kirjutatud käsklus, mis tagastab täpse ette seatud vaste koos veel samal aastal hinnatud filmide arvuga (vt Joonis 14, lk 25).

```
According to:
1. Critics
2. Users
1
    name                          | year | avg_rating
  1. Back to the Future           | 1985 | 10.00
  2. The Mask                     | 1994 | 10.00
  3. Back to the Future Part II   | 1989 | 8.00
  4. The Batman                   | 2022 | 5.00
```

Joonis 9. Parima hinnanguga filmid.

```
publication          | avg_rating | movies_reviewed
TimeOut              | 10.00      | 1
users                | 8.25       | 4
Director's Club      | 8.00       | 1
TIME                 | 7.83       | 6
admin                | 7.76       | 10
Los Angeles Times    | 7.40       | 5
Variety              | 4.75       | 4
```

Joonis 10. Publikatsioonide keskmine.

3.6 Rekursiivne andmehaldus

Andmetöötlusega tegelevate funktsioonide lühitutvustusel jäi mainimata hädavajalik alamfunktsioon, mida kasutavad pea kõik nendest. Seotud andmete lisamisel või andmete üleüldisel uuendamisel on vajalik teada tabeli elementide *ID*'sid. Ülesande nõuetes oli andmete uuendamise nõudeks teha seda läbi *ID'*de, kuid suvaliste numbrite meeles pidamine on äärmiselt ebamugav.

Funktsioon true_id, mis oli loodud täielikuks lisavidinaks on lõpp programmis andmetöötluse keskmeks. Funktsiooni eesmärgiks on sisendmuutujate abil kuvada kasutajale kõik valikuvõimalused ning tagastada kasutaja valitud väärtuse *ID.*

Näiteks kui kasutaja soovib luua arvustust küsitakse esmalt arvustaja nime, mille kuvab kasutajale *true_id.* Nimekirjas rea numbri valides tagastatakse funktsioonist arvustaja *ID* ning sama korratakse ka filmiga ning alles siis küsitakse kasutaja hinnangut.

Suurimaks lisa funktsionaalsuseks kui ka nõrkuseks on funktsiooni võime luua uusi andmeid rekursiivselt. Suure ringiga on kasutaja võimeline sisestama uue tegelase, siduma selle isikuga keda ei olnud varem andmetes, filmiga mida samuti ei eksisteeri režissööriga keda ka andmetes ei leidu. Rekursioon toimub funktsiooni *traffic_officer* uuesti kutsumisel *true_ID* sees, sõltuvalt sisenditest võidakse *true_ID* ise rekursiivselt esile kutsuda.

Kasutajale pakub antud lahendus mugavust, sest andmete sisestamise järjekorral ei ole enam suurt rolli. Enne tegelase lisamist ei pea kõigepealt eraldi vastava menüü läbi looma filmi, režissööri ning näitlejat, kuid andmete lisamise ebaõnnestumisel tekivad eraldi probleemid.

Lahendatud on küll lõpmatud lisamise tsüklid ning täielikud väljumised, kuid kui näiteks viimasel sammul tegelase loomine ebaõnnestub jäävad loodud andmed filmi, režissööri ning näitleja kohta ikkagi andmebaasi. Sõltuvalt arvamusele ei pruugi olukord olla ka halb, sest uuesti tegelast looma minnes on alginfo juba olemas, vaja läheb vaid tegelasega seotud infot.

Kontrollida tuleb ka millal kasutaja käest küsida andmete lisamise võimalust. Näiteks ei ole mõistlik andmete kustutamisel küsida kasutaja käest, et ega juhul, kui oodatud andmeid tabelis ei ole soovitakse need äkki sinna lisada täpselt enne kui need sealt ka kustutatakse.

4 Eriolukorrad

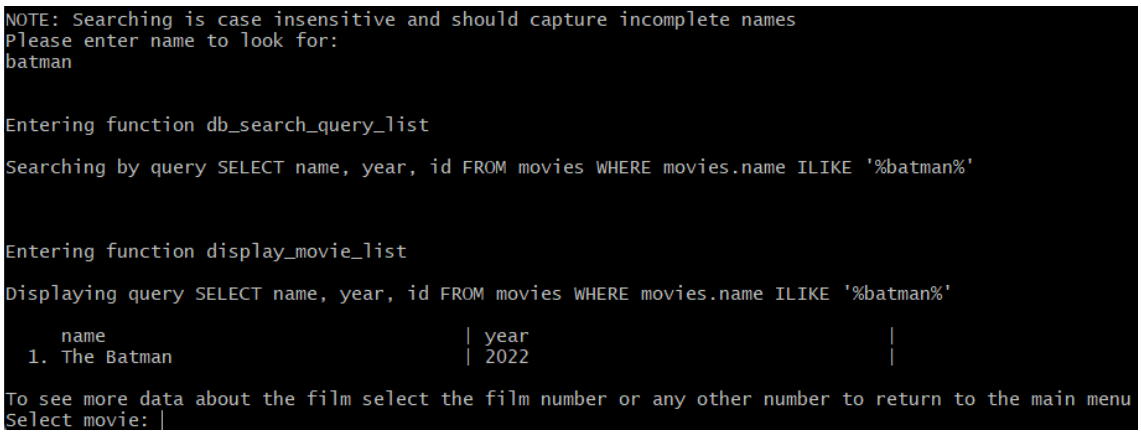
Loodud programmi mahust tulenevalt on eriolukordade hulk samuti meeletu. Teada tuleb täpselt millised kasutaja sisendeid kontrollida ja kuidas teada, et näiteks rekursiooni käigus andmete lisamine ei õnnestunud. Kahjuks ei saa kinnitada programmi iga eriolukorra korrektset käsitlust, kuid mõnedest lahendatud eriolukordadest on võimalik teha väike nimekiri:

- Negatiivsed aastad, hinnangud ei ole lubatud, ka hiljem uuendades.
- Uuendades inimeste või filmide nimesid kehtivad samad piirangud ning ei ole võimalik uuendada nime korduvaks.
- Funktsioon *true_id'*l ei õnnestu kasutajalt saada korrektset *ID'*d, funktsioon tagastab -1 ning esile kutsuvast funktsioonist väljutakse.
- Andmete otsingul puuduvad tulemused.
- Tühjade väärtuste andmine nimedena.
- Piiridest väljas reanumbrid.
- Sama žanri, tegelase, arvustuse (sama inimese poolt) lisamine filmile.
- Filmiga seotud režissööri kustutamine. Funktsioon *is_director*. Kustutatakse ka seotud film.
- Originaalandmete kustutamine kui andmed on seotud. *Cascading* välisvõtmetega kustutatakse ka seotud info.

Teadaolevad vigased olukorrad:

- Filmile on võimalik lisada mitu režissööri, millest detailse vaate kuvamisel näidatakse vaid ühte ning ühe seose kustutamisel kustutatakse terve film. Lahenduseks saaks lihtsalt režissööri eraldi lisamise ning režissööriks uuendamise valikutest ära võtta. Vastavad menüüd olid loodud enne, kui režissööri küsimine liigutati otse pärast filmi andmete sisestamist.

Programmis esinevate vigade lihtsamaks diagnoosimiseks on makro abil loodud ka *debug* režiim, mis väljastab rohkelt infot hetkel käimas oleva alamprogrammi ning saadetavate SQL päringute kohta. Režiimi aktiveerimiseks tuleb käsureal kompileerimisel defineerida muutuja *VERBOSE* käsuga *-DVERBOSE* või eemaldada kommentaar defineerimiskäsult programmi koodi alguses.



Joonis 11. *Debug* režiim.

5 Kokkuvõte

Töö käigus loodi rakendus C keeles programmikoodiga ning SQL skript kasutades PostgreSQL ning pgAdmin 4 tarkvara. Loodud rakendus vastab seatud nõuetele ja pakub ka omapoolseid edasiarendusi ning lisa funktsionaalsust. Lõpp tulemust võiks pikkuselt rohkem võrrelda küll Tarkvara projektiga, kuid ülesanne oli huvitav ning lisatavaid vidinaid jätkus.

Eriolukordadest lahendati ära kõik testimise käigus välja tulnu, kuid täielikku funktsionaalsust ei suuda mahu tõttu lubada.

Kindlasti oleks edasiarenduseks rohkelt ideid, näiteks muutub andmete mahu suurenemisel nimekirjast valimine äärmiselt tüütuks ning lahenduseks vajalikud alamprogrammid on juba enam vähem saadaval, kuid saavutatud tulemusega võib jääda rahule.

6 Allikad

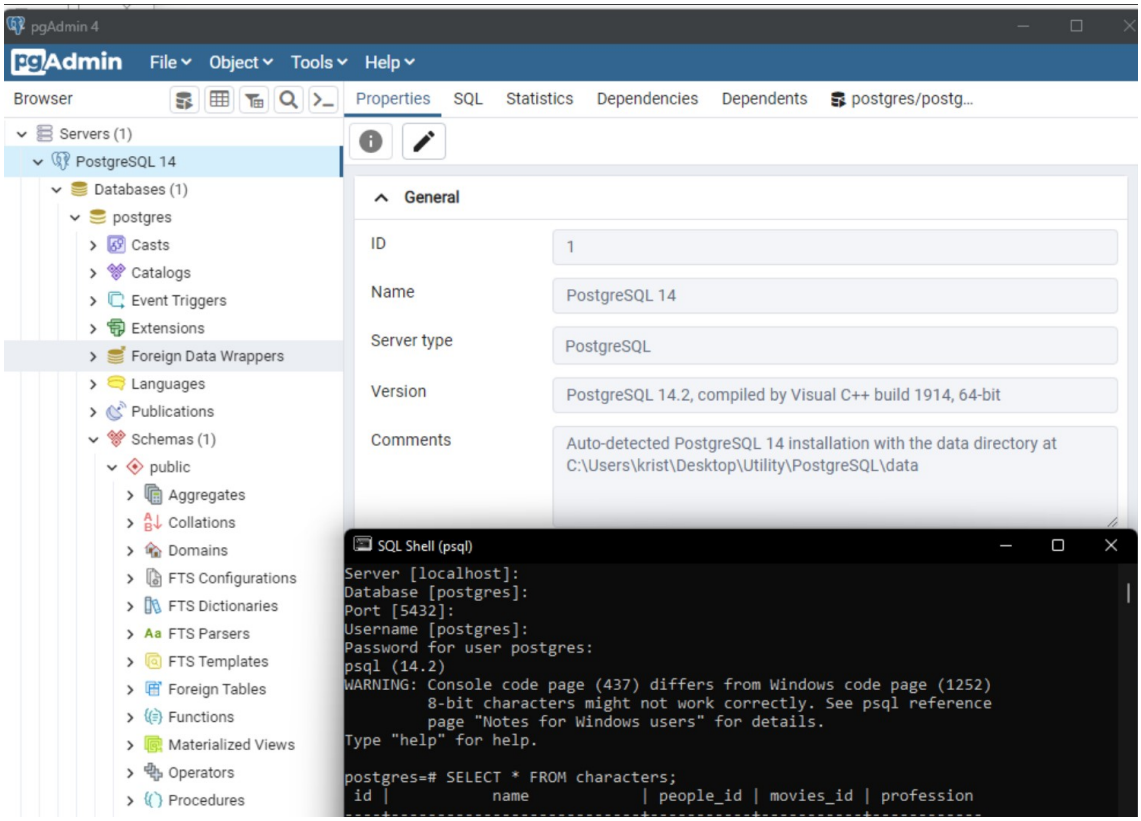
[1] “IMDb,”

https://www.imdb.com/?ref_=nv_home

[2] “Metacritic,”

https://www.metacritic.com/

7 Lisad



Joonis 12. Andmebaasi server.

```
Select row: 1
Please enter short review:
' , NULL, NULL, NULL); TRUNCATE TABLE review;--

Please enter grade 0.0 - 10:
0
Failure (ERROR:  cannot insert multiple commands into a prepared statement
).
Back in main menu, press H for help
X
```

Joonis 13. SQL injection kaitse.

postgres/postgres@PostgreSQL 14

Query Editor

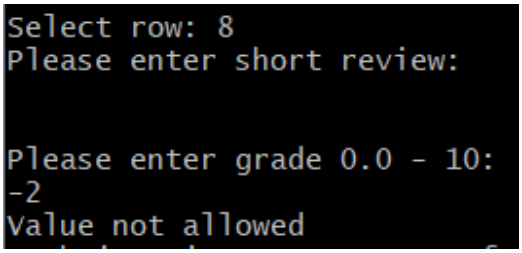
```
SELECT DISTINCT
CASE
WHEN publication IS NULL THEN 'users'
ELSE publication
END AS publication,
year, ROUND(AVG(grade), 2)
AS avg_rating,
COUNT(DISTINCT review.movies_id) AS movies_reviewed
FROM reviewer
LEFT JOIN review ON reviewer.id = review.reviewer_id
LEFT JOIN movies ON movies.id = review.movies_id
GROUP BY publication, year
ORDER BY year DESC, avg_rating DESC NULLS LAST;
```

Data Output | Messages | Explain | Query History | Notifica

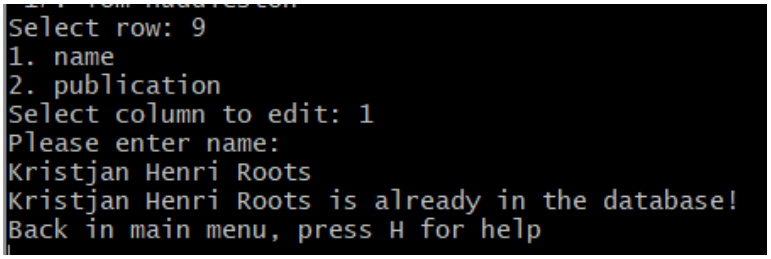
	publication text	year integer	avg_rating numeric	movies_reviewed bigint
1	admin	2022	8.40	1
2	Director's Club	2022	8.00	1
3	users	2022	5.00	1
4	admin	2019	8.20	1
5	TIME	2019	8.00	1
6	Los Angeles Times	2019	5.00	1
7	admin	2014	9.20	1
8	Los Angeles Times	2014	9.00	1
9	TIME	2014	8.00	1
10	admin	2008	6.40	1
11	Variety	2008	6.00	1
12	TIME	2008	5.00	1
13	admin	2006	10.10	1
14	Variety	2003	1.00	1
15	Los Angeles Times	1998	10.00	1
16	TIME	1998	9.00	1
17	admin	1998	7.40	1
18	users	1994	10.00	1
19	Los Angeles Times	1994	8.00	1
20	Variety	1994	6.00	2
21	admin	1994	3.95	2
22	TIME	1990	8.00	1
23	Los Angeles Times	1990	5.00	1
24	admin	1989	10.00	1
25	TIME	1989	9.00	1
26	users	1989	8.00	1
27	admin	1985	10.00	1
28	TimeOut	1985	10.00	1
29	users	1985	10.00	1

Joonis 14. Publikatsioonide aasta keskmine.

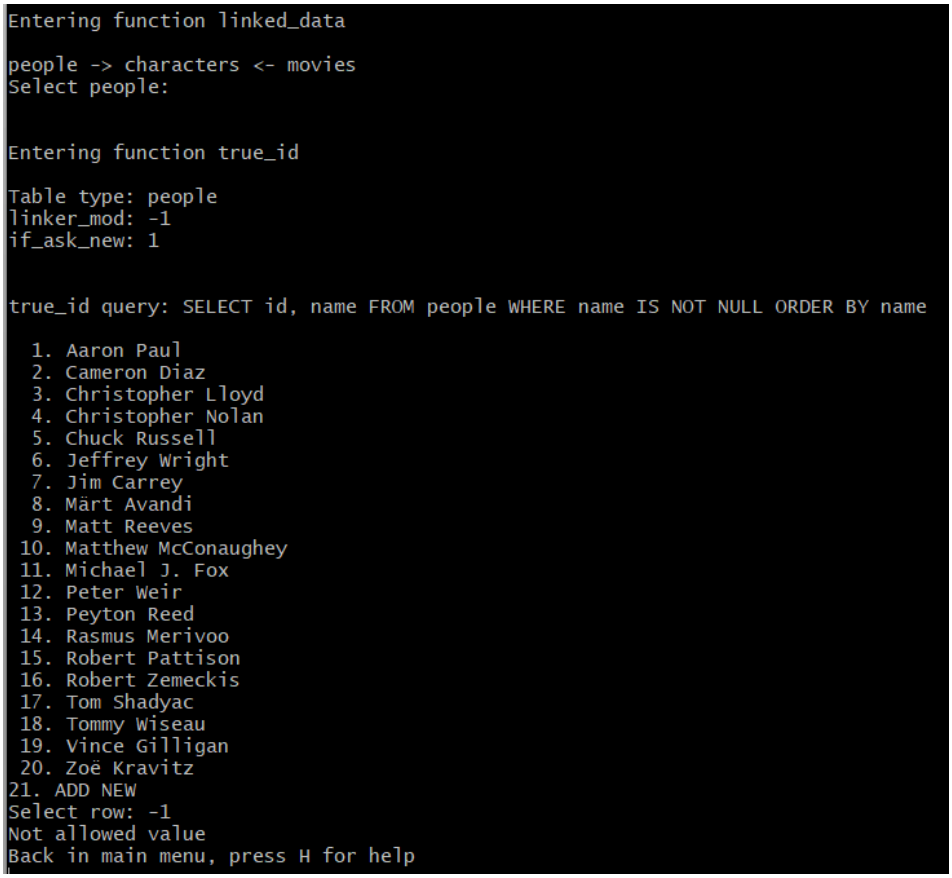
7.1 Eriolukorrad



Joonis 15. Hinnangu väärtuse kontroll.



Joonis 16. Unikaalse nime kontroll.



Joonis 17. *true_id* väljundi kontroll.

```
Search by:
1. Movie name
2. Actor / Director
3. Reviewer
4. Movie genre
5. Character
6. Year
1
NOTE: Searching is case insensitive and should capture incomplete names
Please enter name to look for:
nonexistent
No results
```

Joonis 18. Otsingu tulemuste puudumine.

```
Please enter movie name:

Cannot be empty!
Back in main menu, press H for help
```

Joonis 19. Filmi nime kontroll.

```
Select movies:
  1. Back to the Future
  2. Back to the Future Part II
  3. Back to the Future Part III
  4. El Camino
  5. Interstellar
  6. The Batman
  7. The Mask
  8. The Room
  9. The Truman Show
 10. Tulnukas
 11. Yes Man
12. ADD NEW
Select row: 6
Value already linked to table!!
Back in main menu, press H for help
```

Joonis 20. Juba eksisteeriv seos žanril ning filmil.