**UNIVERZITET CRNE GORE**

**Prirodno-matematički fakultet – Podgorica**

Nikola Darmanović

*Data Mining* tehnike nad podacima UCG-a

ZAVRŠNI RAD

Analitička obrada podataka

Mentor: Savo Tomović

Podgorica, jun 2018.

Sadržaj

[**1.** **Uvod** 3](#_Toc516394777)

[**2.** **Data warehouse sistemi** 3](#_Toc516394778)

[**2.1 Data model** 3](#_Toc516394779)

[**2.2 Dimenzije** 4](#_Toc516394780)

[**2.3 Fakti** 5](#_Toc516394781)

[**2.4**  **Baza Data Warehouse sistema** 5](#_Toc516394782)

[**2.5. Data Warehouse operacije** 6](#_Toc516394783)

[**2.5.1. ETL** 6](#_Toc516394784)

[**2.5.2. Čišćenje podataka** 6](#_Toc516394785)

[**2.6 Meta podaci** 7](#_Toc516394786)

[**2.8 Upotreba Data Warehouse Sistema** 7](#_Toc516394787)

[**3.** **OLAP (On Line Analytical Processing**) 8](#_Toc516394788)

[**3.1 OLAP kocke (Analitičke kocke)** 8](#_Toc516394789)

[**3.2 Olap server** 8](#_Toc516394790)

[**3.1 MDX (Multidimensonal Expression)** 8](#_Toc516394791)

[**4.** **Pentaho, Mondrian OLAP server i Saiku** 10](#_Toc516394792)

[**4.1 Uvod** 10](#_Toc516394793)

[**4.2 Pentaho Data Integration – Kettle** 10](#_Toc516394794)

[**4.2 Schema Workbench** 10](#_Toc516394795)

[**4.3 Mondrian OLAP server** 10](#_Toc516394796)

[**4.4 Saiku** 11](#_Toc516394797)

[**5.** **Primjena Pentaho alata nad podacima UCG** 11](#_Toc516394798)

[**5.1 Zahtjevi** 11](#_Toc516394799)

[**5.2 Pentaho Data Integration – Kettle** 11](#_Toc516394800)

[**5.3 Schema Workbench – kreiranje OLAP kocke** 20](#_Toc516394801)

[**5.4 Saiku** 27](#_Toc516394802)

[**6. Zaključak** 34](#_Toc516394803)

[**Literatura** 35](#_Toc516394804)

1. **Uvod**

Ovaj rad je napisan kao tutorijal koji može koristiti i za kurseve analitičke obrade poadataka. Logički se može podijeliti na dva dijela. U prvom dijelu opisani su ukratko *Data Warehouse* sitemi kao i razlog njihovog nastanka i istorijat, slijede OLAP kocke, Pentaho alati za analitičku obradu i Saiku a sve to prati softverska realizacija nad podacima o upisanim studentima Univerziteta Crne Gore. Softverska realizacija nalazi se na virtuelnoj mašini na kojoj je instaliran set potrebnog softvera a za koju je potrebno 8GB radne memorije i 60GB prostora na disku. Rad je rađen na računaru sa procesorom osme Intelove generacije i5 8600k, 16GB radne memorije i 480GB Solid State diska.

1. ***Data warehouse* sistemi**

*Data warehouse* je kolekcija komponenti i alata za dovlačenje podataka iz različitih sistema i učitavanje tih podataka u bazama podataka dizajniranim za analizu i reporte. Čuvaju se trenutni i istorijski podaci što omogućava korisniku naprednu statističku analizu. Zahvaljujući svom dizajnu, *data warehouse* sistemi mogu uzimati podatke iz različitih međusobno nepovezanih sistema, što ih čini odličnim alatom da se premoste razlike kada se stari sistem poboljšava ili se instalira novi. Pri tome ako se na primjer isključi stari sistem svi podaci iz starog sistema ostaju učitani u data warehouse u odgovarajućem obliku podogdnom za neke vrste statistika.

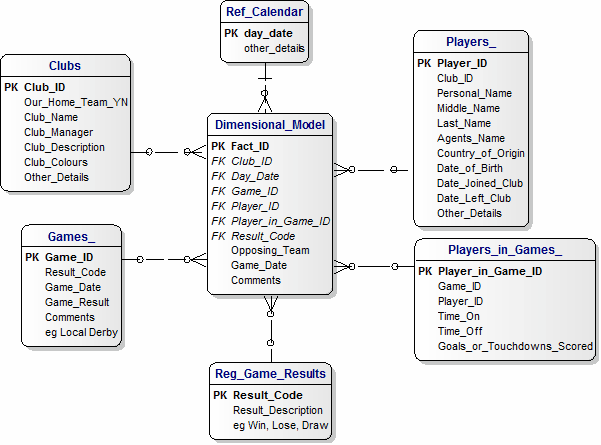
Ideja *data warehouse* sistema nastala je 1988. godine kada su IBM istraživači Beri Devlim i Pol Marfi (*Devlim & Murphy*, 1988) skovali izraz ,,*information warehouse*’’ poslije čega su IT kompanije počele da prave svoje eksperimentalne *data warehouse* sisteme. Bil Inmon je 1992. godine učinio praktičnim data warehouse sisteme objavljivanjem knjige ,,*Building the Data Warehouse*’’. Zatim je Ralf Kimbol 1996. objavio knjigu ,,*The Data Warehouse Toolkit*’’ koja sadrži mnogo praktičnih primjera. Inmon i Kimbol se smatraju očevima data warehouse sistema. Inmonov pristup je sa vrha ka dnu (*top-down*) centralizovani pogled, dok je Kimbolov pristup sa dna ka vrhu (*bottom-up*) konfiguracija zvijezde.

Odluka o pravljenju *data warehouse* sistema treba biti posmatrana iz poslovne perspektive ali ne i iz tehničke. Ako se sistem pravi samo iz tehničkih razloga, onda je velika vjerovatnoća da neće imati koristi za poslovnu logiku jer sistem nije pravljen prema poslovnim potrebama. Dakle, moraju se ispoštovati svi poslovni zahtjevi. Prema tome, poslovni zahtjevi odlučuju koji podaci se moraju nalaziti u data warehouse sistemu, kako da se organizuju, koliko često će se ažurirati i kako će se podaci izvlačiti.

**2.1. Data model**

Relacioni model je veoma zastupljen u *OLTP* (*On Line Transaction*) sistemima. Relacioni modeli su kolekcije entiteta i relacija među njima. Ovi entiteti su dizajnirani da eliminišu redudantnost podataka i da sam proces transakcije učine jednostavnim i brzim. Moderni poslovni sistemi obično sadrže hiljade entiteta koji su mapirani u tabelama baza podataka. Korišćenje modela direktno za analizu ili izvještaje je neefikasno jer krajnji korisnik ne može da se orijentiše i da brzo shvati model.

Dimenzioni model je tehnika dizajna (organizacije) koja predstavlja podatke u standardnom okviru za *data warehouse* siteme. Dimenzioni model se sastoji od fakt tabele i skupa dimenzionalnih tabela. Fakt tabele su numerički dio modela dok su dimenzionalne tabele deskriptivni dio. Ovo je karakteristika Kimbolovog pristupa koji je predložio konfiguraciju zvijezde. Pošto se Data warehouse sastoji od mnogo modela gdje fakt tabele daju dimenzije, onda se fakt tabele mogu povezati preko zajedničkih dimenzija. Ovaj dizajn je mnogo razumljiviji za izvještaje kranjem korisniku jer se vrši ,,grupisanje’’ podataka po nekim dimenzijama.



Slika 1: Primjer dimenzionalnog modela

**2.2. Dimenzije**

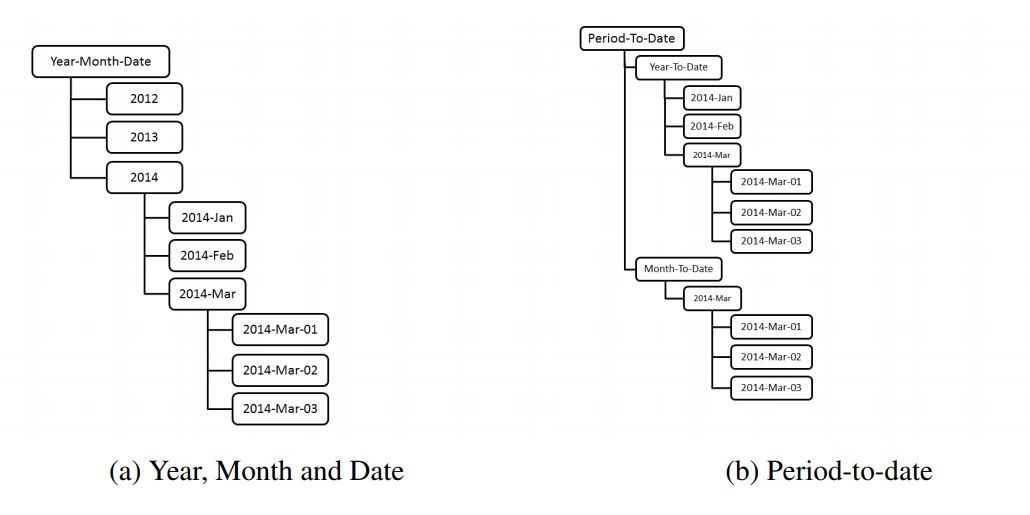
Dimenzija je kolekcija atributa (tekstualnih) koji su u međusobnoj korelaciji. U školskom data warehouse sistemu to su na primjer dimenzije student, predmet, semestar, smjer, vrijeme… Dimenzija student može sadržati atribute kao što su ID, ime, adresa, država, datum rođenj i slično. Atributi su većinom tekstualni, datumski ili numerički. Ti atributi nisu kvantitativni kao što su recimo broj predmeta ili broj svršenih studenata. Ove se informacije izlače naknadno iz data warehouse agregacijom fakt tabela.

Atributi dimenzija se vremenom mogu promijeniti, kao na primjer opis predmeta. SCD(*Slowly Changing Dimension*) definiše različite tipove mogućih izmjena:

* tip 0 kada je vrijednost atributa konstantna i nikada se ne mijenja
* tip 1 kada se vrijednost atributa mijenja
* tip 2 kada se vrijednost atributa mijenja ali se čuvaju sve stare (istorijske) vrijednosti atributa

Moguće je definisati još ovih tipova za neke specifične potrebe industrije. Atributi jedne dimenzije mogu imati različite SCD tipove.

Neke dimenzije mogu imati mnogo članova iz čega proističe da je nerealno ili nepraktično da se svi članovi drže u jednoj velikoj listi radi pretrage. Dobar primjer za ovaj problem je vremenska dimenzija. Da bi se što optimalnije organizovale liste po vremenskoj dimenziji obično se koriste hijerarhijske strukture jer se time jasno uređuju podaci na primjer po godinama, po mjesecima, po danima itd. pa lako i optimalno možemo pretraživati podatke po zahtijevanom vremenskom periodu.



Slika 2: primjeri hijerarhijskih struktura

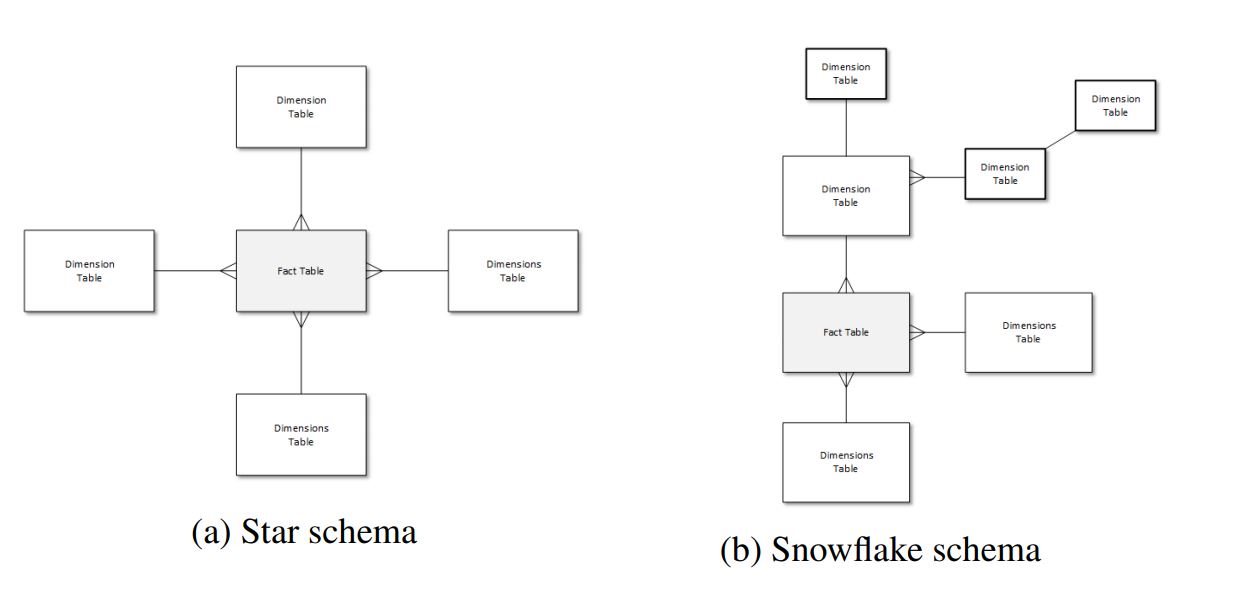
**2.3. Fakti**

Fakti su podaci koji nisu poznati unaprijed i najčešće su numerički. Ako recimo gledamo neki tipični transakcioni sistem kao što je student menadžment sistem, onda su ocjene studenata kao i ETSC krediti predmeta fakti u *data warehouse* sistemu. Ocjena studenta će postati zapis u fakt tabeli sa referencama na dimenzionalne tabele. Prilikom izvlačenja podataka iz data warehouse Sistema često se javlja potreba da se nad podacima izvrše neke agregatne funkcije kao što su prebrojavanje, suma, minimalna vrijednost, maksimalna vrijednost, grupisanje i slično. Da bi se to postiglo, fakt tabele se spajaju preko dimenzija, rekordi se grupišu zajedno i zatim se izvrši odgovarajuća agregatna funkcija.

**2.4. Baza *Data Warehouse* sistema**

Konfiguracije zvijezde je dimenzionalni model sastavljen od fakt tabele i od niza manjih tabela koje se nazivaju dimenzionalne tabele. Data warehouse sistem se sastoji od više takvih modela gdje fakt tabele dijele dimenzije.

Projektanti baza za *data warehouse* sisteme često pokušavaju da sačuvaju prostor u bazi time što razbijaju dimenzije u manje tabele. Radeći to, konfiguracije zvijezde se fizički mijenja pa dobijamo konfiguraciju pahulje. Konfiguracija pahulje se dobija kada jedan ili više atributa jedne dimenzije se razmjeste u odvojenim dimenzijama i onda se te dimenzije povežu u originalnu dimenziju. Primjer može biti dimenzija korisnik gdje se demografske infomarcije male kardinalnosti smještaju u odvojenim dimenzijama. Demografska dimenzija ima manje članova od dimenzije korisnika i te dvije dimenzije se ne učitavaju istovremeno. Preveliko razbijanje u konfiguraciju pahulje nije preporučljivo jer često komplikuje korišćenje data warehouse sitema.



Slika 3: konfiguracije zvijezde (lijevo) i pahulje (desno)

## **2.5. Data Warehouse operacije**

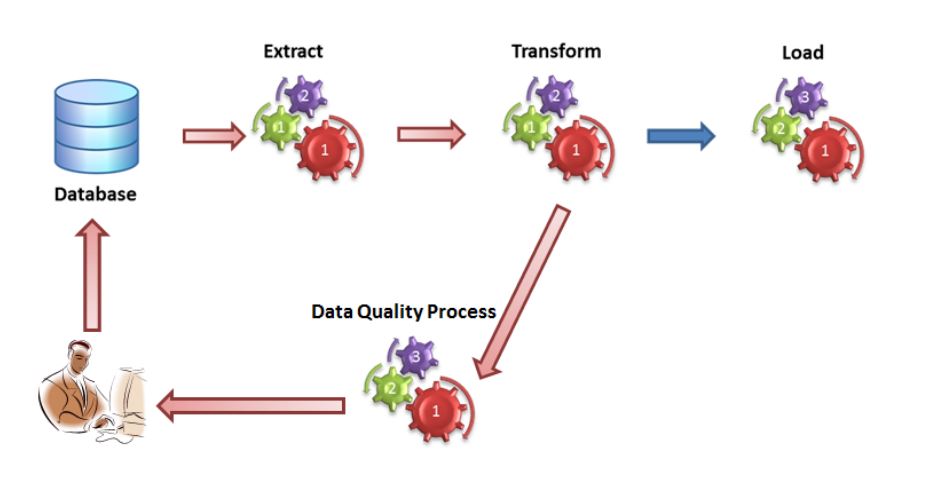
Da bi napravili data warehouse sistem moramo napraviti dimenzije i fakt tabele. U dobro dizajniranom data warehouse sistemu gdje postoji mnogo fakt tabela, često se dešava to da se dijele dimenzije. Praveći manji broj dimenzija i pri tome ih dobro definisati, znači lakše korišćenje podataka za krajnjeg korisnika. Prazan data warehouse nije od koristi, pa se podaci moraju dohvatiti, očistiti i transformisati iz drugih baza ili sistema i onda učitati u data warehouse sistemu.

**2.5.1. ETL**

Proces pravljenja data warehouse sistema se naziva ETL (Extract, Transform and Load). U fazi izlvačenja (extract) vrši se povezivanje sa različitim izvorima iz kojih se izvlače podaci. U fazi transformacije vrši se poravnavanje sličnih podataka kao što je recimo pol koji se u nekim sistemima može predstaviti sa M i Ž ili M i F ili u nekima sa 1 i 0. Na kraju u fazi učitavanja čuvaju se očišćeni podaci.

**2.5.2. Čišćenje podataka**

Kvalitet podataka je jedan od najvažnijih faktora data warehouse sistema. Ako je kvalitet dovučenih podataka loš onda je *data warehouse* sistem napravljen uzalud. Prije nego što se podaci smjeste u data warehouse sistemu moraju proći kroz nekoliko procesa čišćenja i pravila preslikavanja da bi se pronašli i riješili problem kao što su nedostajuća vrijednost, pogrešnja vrijednost ili više različitih vrijendnosti za jedan atribut koje treba preslikati u potvrđenu konvencionalnu vrijednost. Takođe, administratori zaduženi za podatke se obavještavaju o netačnim podacima da ih mogu ispraviti, da bi se izbjegla greška prilikom automatizacije ispravljanja, pa se narednim ažuriranjem učitavaju tačni podaci.



Slika 4: primjer čišćenja podataka

**2.6. Meta podaci**

Meta podaci su podaci o podacima: ,,Meta podaci su sve informacije koje definišu i opisuju sadržaj, strukture i operacije *data warehouse sistema*’’(*Mundy,* *Thronthwaite* *& Kimball, 2008*). Meta podaci opisuju na primjer tip svakog atributa u dimenziji ili koje dimenzije i fakti postoje u sistemu. Repozitorijum meta podataka se obično skladišti u posebnoj bazi podataka i koristi ga neki spoljašnji softver da dobije podatke o strukturi data warehouse sistema u svrsi izvršavanja nekih operacija.

**2.8. Upotreba *Data Warehouse* Sistema**

Kada se kreiraju dimenzije i učitaju se fakt tabele se u data warehouse system onda se sve vrste izvještaja i analitičkih operacija se mogu izvršiti nad učitanim podacima. Fakt tabele se mogu spojiti kroz dimenzije, grupisati i urediti na mnogo različitih načina. *Data warehouse* može čuvati istorijske promjene, pa korisnik može vršiti upite retrospektivno. Specifični djelovi *data warehouse* sistema mogu biti dostupni različitim grupama korisnika ili određenim aplikacijama definisanjem pogleda nad dimenzijama i fakt tabelama.

1. **OLAP (*On Line Analytical Processing***)

## **3.1. OLAP kocke (Analitičke kocke)**

Analitičke kocke ili OLAP kocke su multidimenzionalni nizovi podataka koji se mogu posmatrati na različite načine. Kocke se mogu napraviti od podataka u data warehouse sistemu i one omogućavaju korisniku da upoređuje različite aspekte podataka zajedno, imaju mogućnost da prikažu sve istorijske promjene ili da predvide neke promjene u budućnosti. OLAP je akronim za *On Line Analytical Processing*, što je metod za analizu poslovnih podataka. Operacije koje se mogu izvršiti nad kockom su sječenje, bacanje, bušenje ka dnu, bušenje ka vrhu i pivot.

OLAP kocke sadrže sledeće operacije

* **Sječenje** je operacija kojom se uzima jedna dimenzija kocke, odnosno kocka se siječe po jednoj dimenziji čime se dobija podkocka.
* **Bacanje** je operacija slična bušenju, pri čemu uzimamo više dimenzija kocke (na primjer vrijeme, lokacija i cijena) i tako kreiramo novu podkocku.
* **Pivot** (ili rotacija) je operacija koja rotira kocku podataka po nekoj osi da bi se obezbijedio alternativni prikaz pokdataka.
* **Bušenje ka vrhu** je operacija agregacije nad kockom podataka kojom se redukuje broj dimenzija. Time na primjer ako imamo dimenzije lokacija i vrijeme i ako ovom operacijom uklonimo dimenziju vrijeme, onda se podaci agregiraju samo po lokaciji.
* **Bušenje ka dnu** je operacija kojom se povećava broj posmatranih dimenzija. Ovom operacijom dobijamo više detalja i složeniju kocku.

Neki OLAP sistemi nude dodatne operacije bušenja, kao što su **bušenje preko** koja izvršava upite nad više od jednom fakt tabelom ili **bušenje kroz** operacija koja koristi relacione SQL mehanizme da buši od donjih nivoa kocke ka krajnjim relacionim tabelama. Takođe postoje i dodatne operacija kao što su rangiranje, računanje prosjeka, računanje kamata, konverzije valuta...

## **3.2. Olap server**

Glavna komponenta OLAP-a je OLAP server koji je zapravo spona između klijenta i menadžment sistema baza podataka (DBMS). OLAP server razumije na koji način su podaci organizovani u bazi podataka i ima specijalne fukncije za analizu podataka.

## **3.3. MDX (*Multidimensonal Expression*)**

Multidimenzionalni izrazi ili MDX (*Multidimensional expression*), je upitni jezik namijenjen za definisanje i upravljanje multidimenzionalnim objektima i podacima. Ovaj upitni jezik je veom sličan strukturiranom upitnom jeziku (SQL). Sve funkcionalnosti koje postoje u MDX-u, mogu se ostvariti i pomoću SQL-a, ali ne na tako jednostavan, intuitivan i brz način. Strukture pomoću koji se multidimenzionalni podaci mogu prikazati u više od dvije dimenzije se nazivaju kocke, a skup podataka koji se nalazi u presjeku dimenzija kocke, naziva se mjera. Mjera se može sastojati od jednog ili više podataka. Jedna kocka se sastoji od dimenzija, svaka dimenzija podijeljena je u različite nivoe, a svaki od nivoa dalje je podijeljen na članove. Kocka može da sadrži i jedan ili više atributa mjere koji se nalaze u privatnoj dimenziji mjera, na osnovu koje se referencira određena mjera u upitima. Svaki MDX upit sastoji se od:

* *SELECT* klauzule, koja se koristi da odredi dimenzije i članove koji će biti prikazani, odnosi se na dimenziju ose (*axis*), po kojoj se podaci dobijaju za više članova. Sadrži specifikacije za ose (dimenzije), u presjeku ovih dimenzija nalazi se tražena vrijednost, a podržano je do 128 dimenzija. Prvih pet osa zovu se kolone, redovi, strane, sekcije i poglavlja, preporučljivo je koristiti nazive axis(0), axis(1), itd. zato što neki alati ne podržavaju pomenute nazive osa, takođe je neophodno ispratiti redosled osa da ne bi doslo do greske.
* *FROM* klauzule koja određuje odakle se podaci dovlače. Moguće je selektovati samo jednu kocku u jednom upitu.
* *WHERE* klauzule koja filtrira podatke, i nije obavezna, služi da definiše restrikcije prema nekim dimenzijama ili članovima, odnosi se na slicer dimenziju koja predstavlja dio koji ulazi u rezultat, tj. onaj dio koji se mapira na ose definisane u select klauzuli. Nije dozvoljeno da u jednom upitu članovi iste dimenzije definišu različite ose ili i osu i where klauzu.

Ćelija (*cell*) multidimenzionalne strukture se nalazi u presjeku po svim članovima te strukture, a iz ćelije se dalje dobijaju željeni podaci. Da bi se podaci dobili, koriste se torke (*tuples*), koje predstavljaju sistem referenci. Pomoću torki, mogu da se identifikuju kako jedinstvene ćelije, tako i čitav skup njih. Skup (*set*) je uređena kolekcija torki. Skup torki se piše između vitičastih zagrada („ { “, „ } “), torke unutar skupa se odvajaju zarezom („ , “).

MDX funkcije:

* **Members** vraća skup svih članova koji pripadaju jednoj dimenziji ili nivou.
* **Children** može da se koristi za dohvatanje sve djece neke dimenzije, a takođe i za dohvatanje sve djece nekog člana određenog nivoa dimenzije. Fukciju children nije moguće koristiti za dohvatanje svih članova određenog nivoa, kao što je to slučaj kod funkcije members.
* **Non empty** ukoliko u rezultatima ima praznih polja, pomoću ove funkcije oni se eliminišu. Ova funkcija se može upotrijebiti nad jednom ili nad više dimenzija.
* **Filter** pomoću koje prikazuju se samo oni rezultati koji ispunjavaju zadati uslov.
* **Head koristi** se za selekciju određenog broja podataka.
* **Crossjoin(set1, set2)** vraća proizvod dva ili više navedenih skupova.

1. **Pentaho, Mondrian OLAP server i Saiku**

**4.1. Uvod**

Pentaho je open source softver za poslovnu inteligenciju, razvijen od strane Pentaho Corporation, koji pruža usluge integracije podataka, OLAP servisa, kopanja podataka kao i ETL-a. Od 19. septembra 2017. godine Pentaho je postao dio kompanije Hitachi Ventara koja objedinjuje Pentaho, Hitachi Data Systems i Hitachi Insight Group. Pentaho za čitanje i organizaciju podataka koristi Mondrian OLAP open source server razvijen takođe od strane Pentaho Corporation. U ovom radu koristi se desktop aplikacija Pentaho Data Integration sa kodnim nazivom Kettle. Za lakši i interaktivni prikaz podataka i analizu može se koristiti bilo koja klijentska aplikacija koja radi sa Mondrian OLAP serverom, pa je za tu svrhu pogodna Saiku klijentska web aplikacija.

**4.2*. Pentaho Data Integration – Kettle***

Pentaho Data Integration (kodni naziv *Kettle*) je Pentaho desktop aplikacija koja se sastoji od ETL funkcija i naprednog grafičkog interfejsa zvanog *Spoon*. Kettle omogućava lako kreiranja, ažuriranje i održavanje *data warehouse* sistema. Glavni alati koje *Kettle* posjeduje su *Pan* i *Kitchen*. *Pan* sadrži mnoštvo funkcija za čitanje podataka iz različitih izvora (SQL, XML, CSV...), manipulaciju podataka i upis podataka. *Kitchen* je zadužen za izvršavanje zadataka koji su kreirani uz pomoć *Spoon*-a. Zadaci se obično planiraju da se izvršavaju u paketima i to automatski u određenim intervalima. Transformacije i zadaci su obično opisani preko XML fajla ili se nalaze u *Kettle* repozitorijumu.

**4.2. *Schema Workbench***

Sechema Workbebench je dizajnerski interfejs koje omogućava vizuelno kreiranje i testiranje šema Mondrian OLAP kocki. Mondrian vrši procesiranje MDX zahtjeva sa ROLAP šemama. Te šeme se čuvaju kao XML fajlovi i predstavljaju meta modele koji su kreirani u specifičnim strukturama koje koristi Mondrian platforma. Ti XML modeli se mogu posmatrati kao kuboidne strukture koje koriste postojeće fakt i dimenzionalne tabele koje se nalaze u RDBMS (Relacioni menadžment sistem baza podataka). To znači da nije potrebno da se pravi i održava stvarna kocka već samo da se kreira meta model.

**4.3. Mondrian OLAP server**

Mondrian OLAP server je open source OLAP server napisan u programskom jeziku Java. Podržava MDX upite, XML za analize i *olap4j* *interface* specifikacije. Ovaj server čita podatke iz SQL ili nekog drugog izvora i agregira podatke u Mondrian keš memoriji (ovo nije keš memorija CPU-a već jedna memorija koju Mondrian OLAP server koristi).

Mondrian se koristi za:

* Visoko performantne interaktivne analize ogromnih ili malih količina podataka
* Analize podataka po dimenzijama
* Parsiranje MDX upita u SQL upite koji odgovaraju dimenzionalnim upitima
* Brze upite nad agregatnim tabelama u RDBMS
* Napredno računanje koristeći MDX izraze za izračunavanja

**4.4. Saiku**

Saiku je tanki klijent. Prvobitno ovo je bila aplikacija *Pentaho Analysis Tool* nastala 2008. godine ali je vremenom evoluirala u veliki projekat koji je 2010. nanovo napisan i nazvan Saiku. Saiku omogućava poslovnim korisnicima da analiziraju kompleksne podatke koristeći interaktivni interface. Podaci se mogu prikazati tabelarno ili grafički. Kada se dobiju odgovarajući podaci, ti rezultati mogu da se čuvaju ili da se izvezu u obliku PDF-a ili Excel-a.

1. **Primjena Pentaho alata nad podacima UCG**

**5.1. Zahtjevi**

Za ovaj rad potreban je računar sa makar 4GB radne memorije. Od programa potrebno je imati instalirano:

* *Java JRE*
* *XAMPP (MySQL, Apache, Tomcat)*
* *Kettle*
* *Schema Workbench*
* *Saiku*

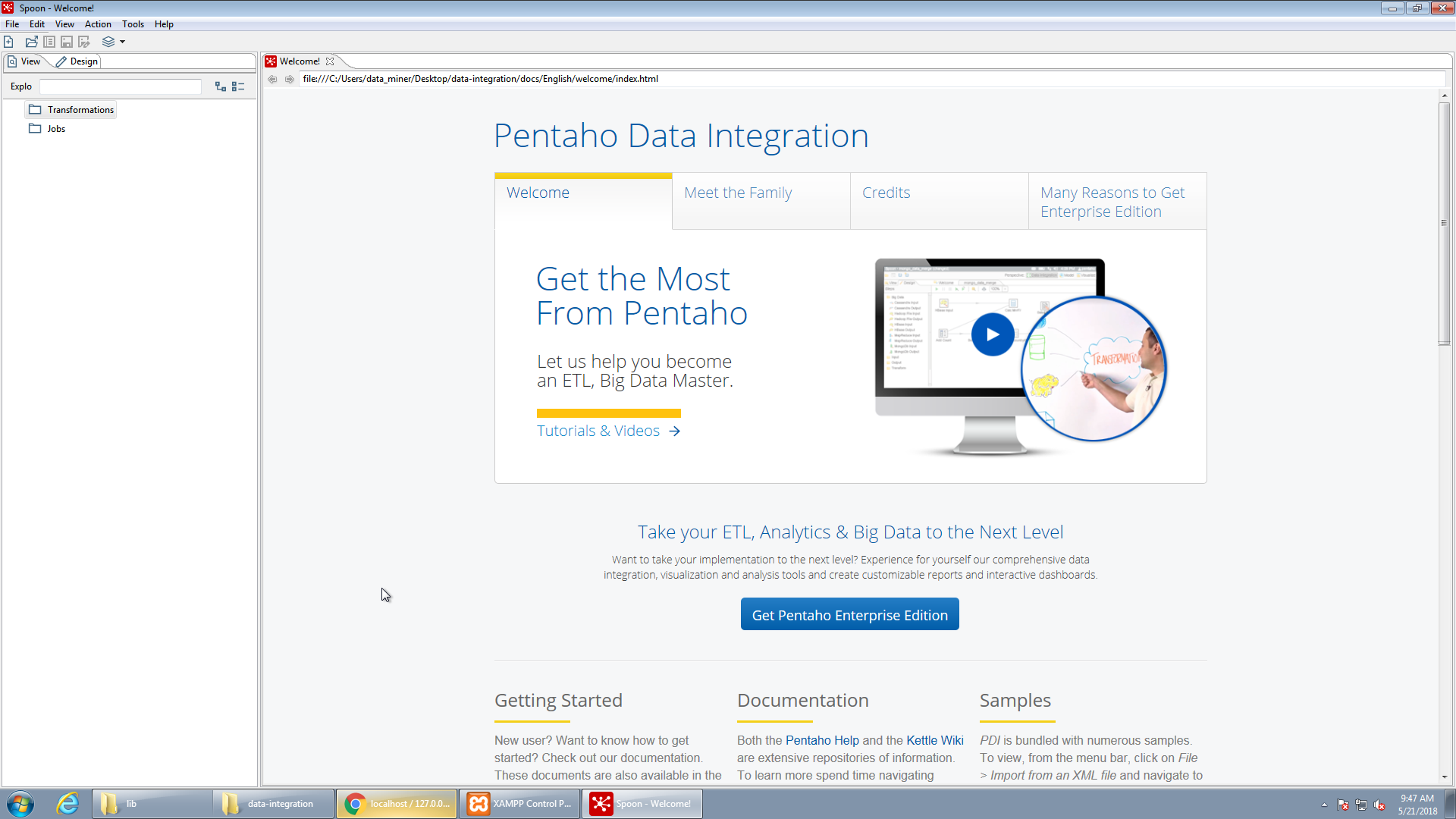
**5.2. *Pentaho Data Integration – Kettle***

Prvi korak koji je potreban da se uradi je da se sa preuzme *mysql-connector-java-\** JDBC drajver za *MySQL* sa <https://dev.mysql.com> i da se smjesti u lib folderu. Putanje do ovog foldere su obično:

PC: C:\Program Files\pentaho\design-tools\data-integration\lib

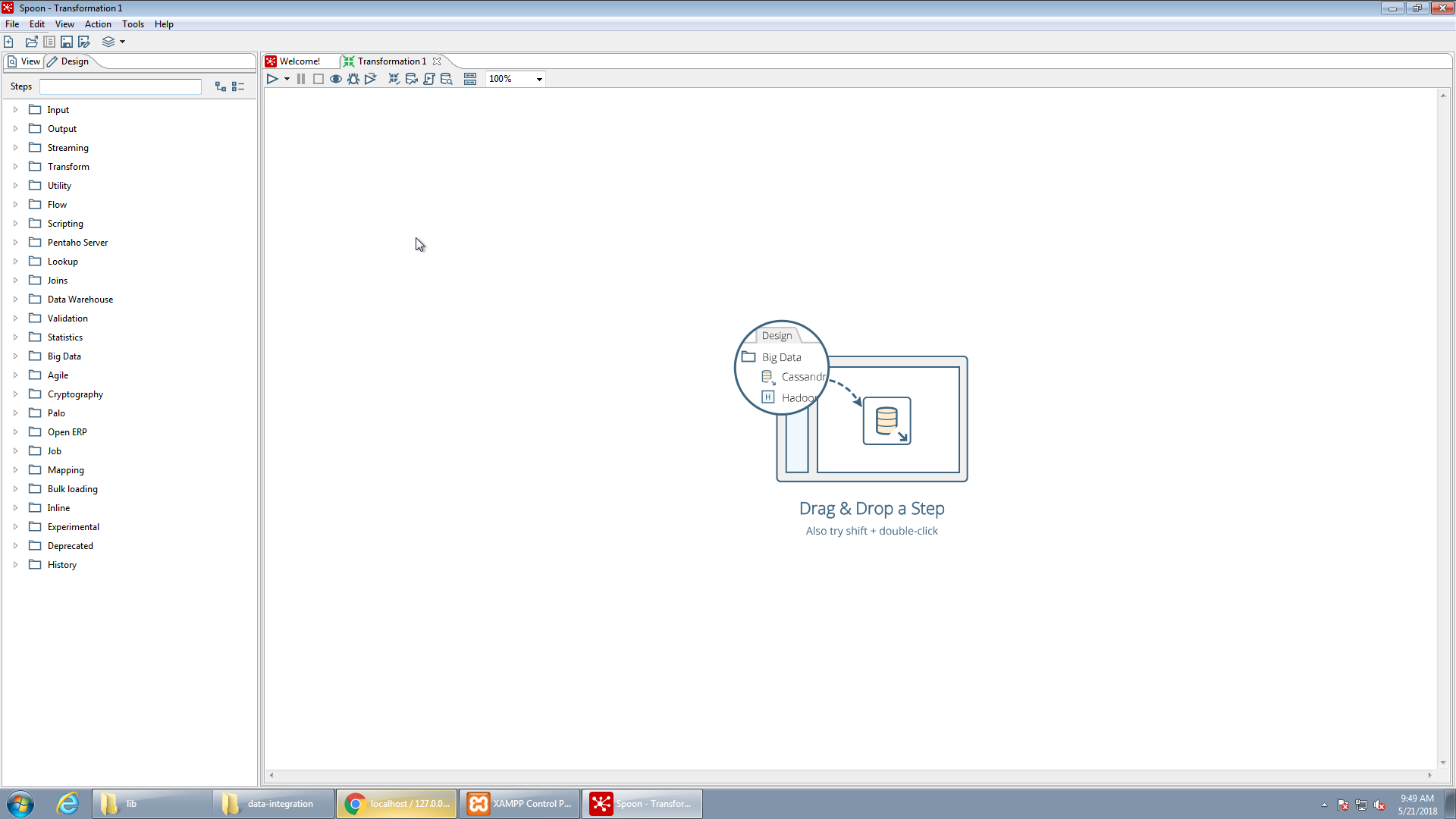
MAC: /Applications/data-integration/lib

Zatim potrebno je kreirati MySQL bazu. Nazovimo je ,,studenti\_ucg''. Sada treba pokrenuti fajl *Spoon.bat* koji pokreće *Kettle GUI*. Često se dešava da na Windows operativnim ne postoji sistemska varijabla java\_home koja sadrži putanju do foldera trenutne instalirane verzije Jave, pa je potrebno dodati jer u suprotnom nemoguće je pokrenuti *Spoon*.



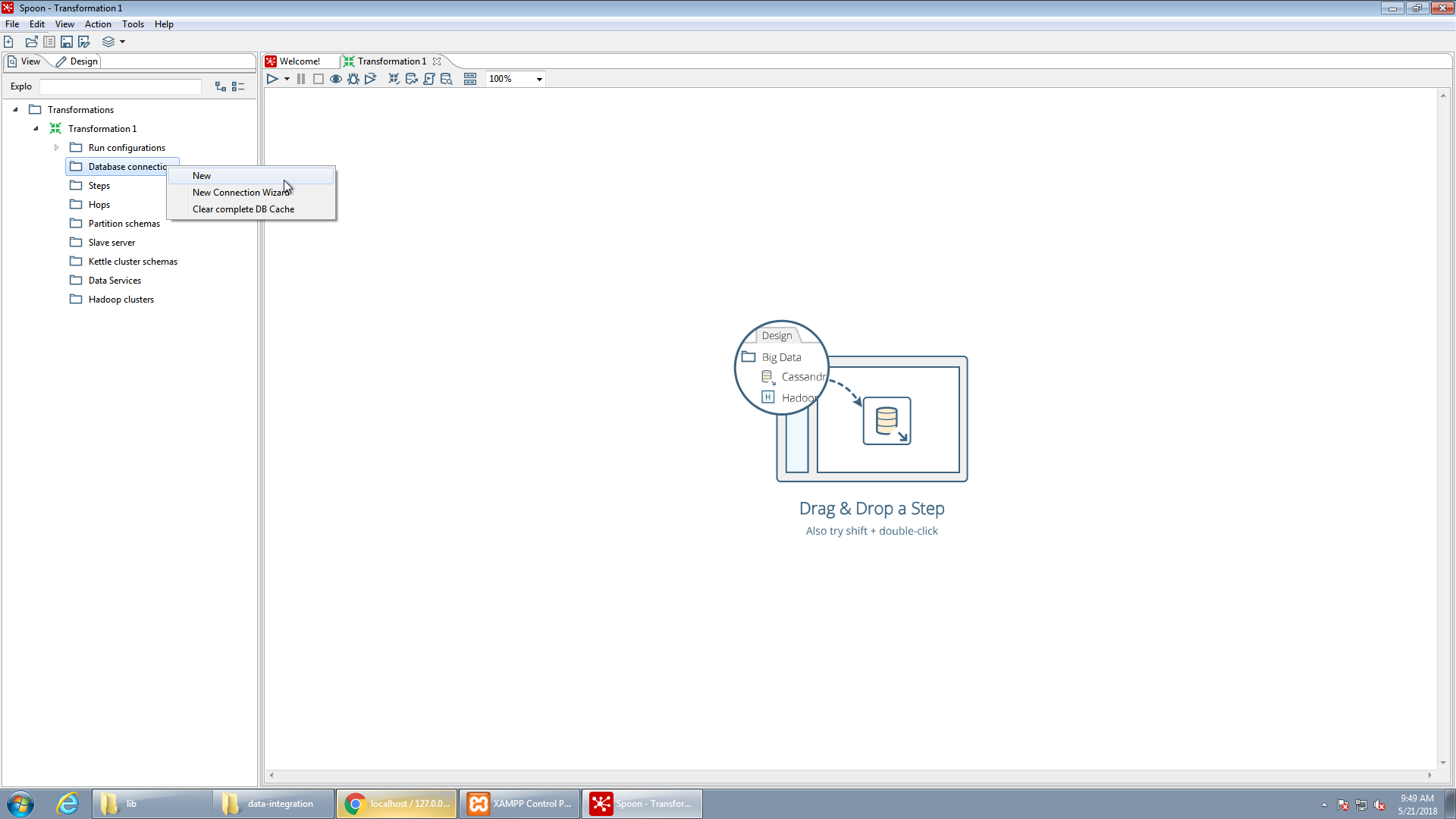
Slika 5: Izgled *Spoon GUI*-a

Kada smo pokrenuli *Spoon* prvi korak je da se kreira nova transformacija i sačuva. Moguće je samo duplim klikom na stavku *Transformatons*.



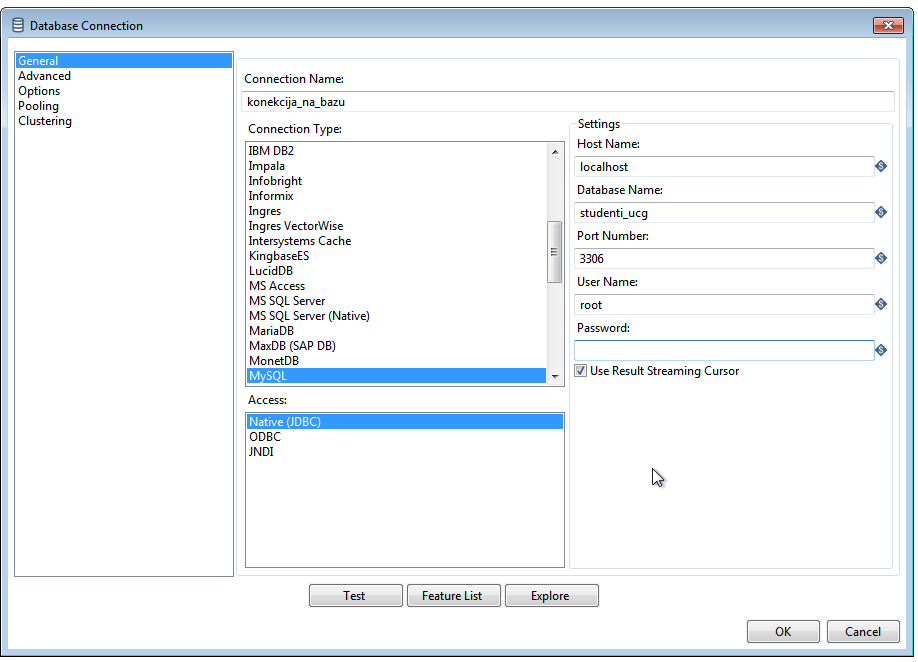
Slika 6: Kreirana transformacija

Sada je potrebno uspostaviti konekciju sa bazom studenti\_ucg. U tabu view na stavki *Database connections* desnim klikom pa zatim sa *add new connection* otvaramo prozor za dodavanje konekcije.



Slika 7: Otvaranje prozora za kreiranje konekcije

Konekcija se može imenovati bilo kako, za tip konekcije potrebno je odabrati MySQL a za pristup *Native(JDBC)*. Pošto se koristi XAMPP, predefinisani host name je localhost, username je root a lozinka je prazna. Za ime baze treba upisati ime baze koju smo već kreirali, u našem slučaju studenti\_ucg.



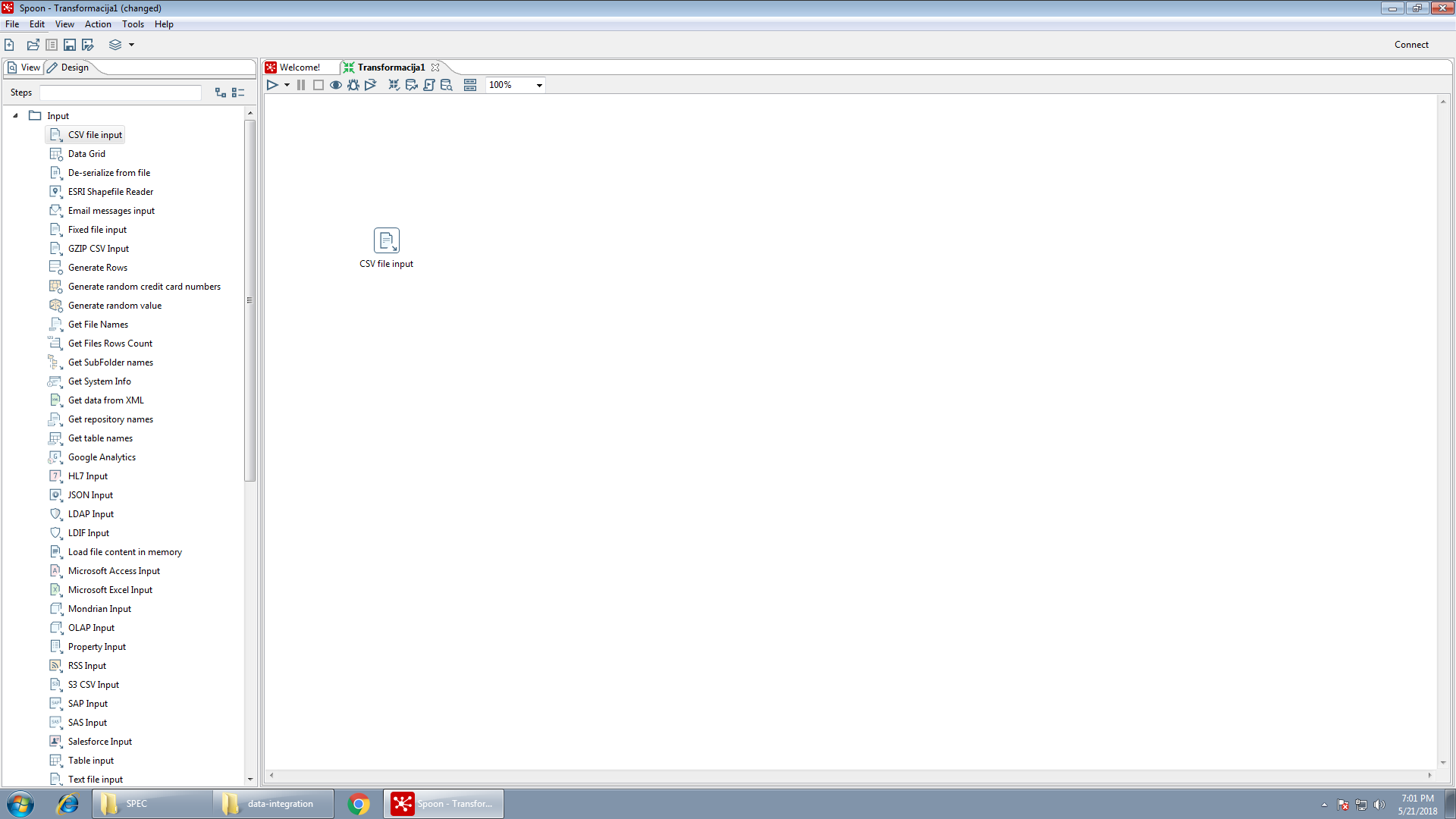
Slika 7: Kreiranje konekcije

Kada se unesu ovi podaci, preporučuje se da se testira konekcija preko dugmeta *Test*. Ako je sve u redu pojaviće se sledeća poruka, a u suprotnom potrebno je provjeriti da li je skinut odgovarajući JDBC drajver za konekciju na MySQL



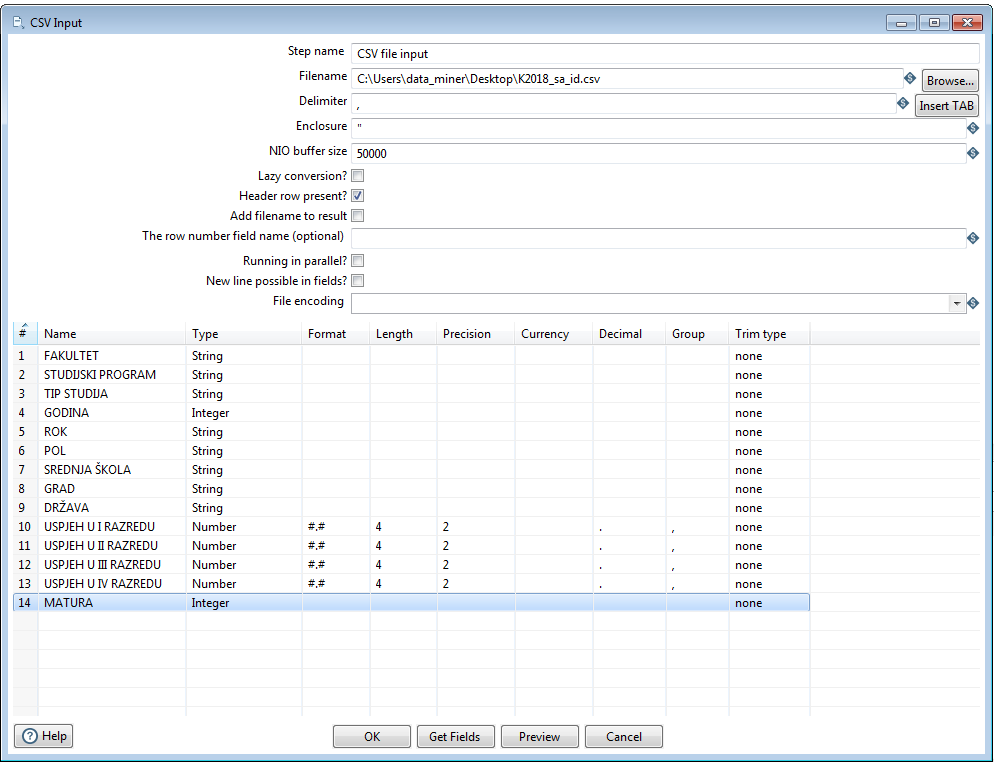
Slika 8: Uspješno kreirana konekcija

Nakon uspješne konekcije sa bazom, sve je spremno za prvi konkrenti korak, a to je učitavanje podataka. Podatke učitavamo iz CSV fajla K2018\_sa\_id.csv koji se nalazi na vrituelnoj mašini. Taj fajl sadrži podatke sa kolonama fakultet, studijski program, tip studija, godina (upisa), rok, pol, srednja škola, grad, država, uspjeh u I razredu, uspjeh u II razredu, uspjeh u III razredu, uspjeh u IV razredu i matura. Da bi se omogućio unos potrebno je iz taba za dizajn ubaciti stavku *CSV File Input* koja se nalazi pod *Input*. To je moguće duplim klikom ili operacijom drag and drop kao i za sve ostale operacije iz lijeve sekcije.



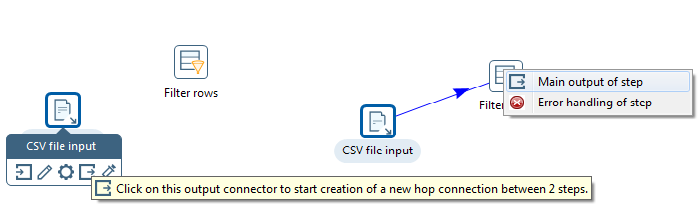
Slika 9: CSV file input

Sada duplim klikom otvorimo stavku koju smo dodali. Izaberemo lokaciju fajla. Zatim preko dugmeta *Get Fields* aplikacija sama pokupi kolone iz CSV fajla ali je potrebno ručno podesiti da paramatre da bude kao sa slike 10. Nakon unešenih kolona na dugme *Preview* provjerava se da li se podaci uspješno učitavaju. Ovom akcijom obezbijedili smo izvor podataka za rad.



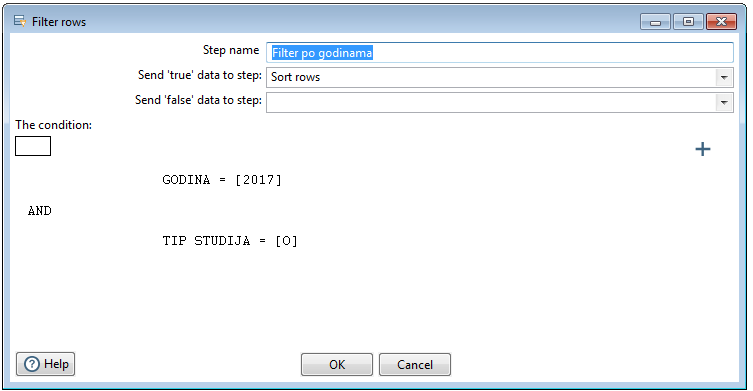
Slika 10: Kolone izvora

Nakon što se obezbijedio izvor podataka, može se dodati filter koji će filtrirati podatke po godinama. Filter se dodaje isto kao i za unos podataka. Taj korak se naziva *Filter rows* i nalazi se pod *Steps*. Kada se doda potrebno je povezati izvor sa filterom. To se radi tako što se zadrži kursor na izvoru i onda će se pokazati dodatne opcije a u ovom slučaju potreban nam je *file outpup*. Zatim dođe do filtera i odabere se *Main outpup of step*.



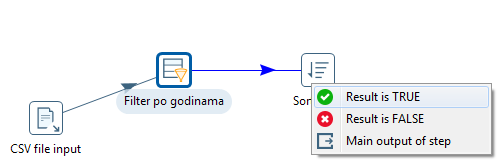
Slika 11: Spajanje izvora i filtera

Sad je potrebno podesiti filter. Povezivanje je potrebno prije podešavanja filtera da bi se u *Filteru* prepoznala polja, odnosno kolone, izvora. U filteru moguće su razne logičke manipulacije.

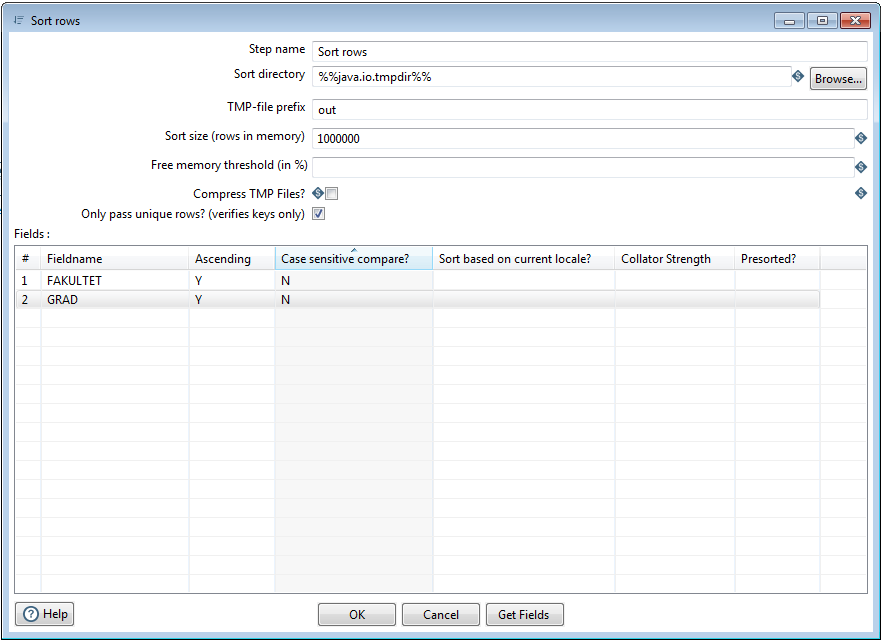


Slika 12: Filter po koloni ,,GODINA''

Sledeći korak je sortiranje redova. Ovo je veoma jednostavna operacija koja sortira redove po jednoj ili više kolona. Prvi tok podataka filtriraće se po gradovima i fakultetima. *Sort rows* nalazi se pod *History*. Prilikom spajanja moguće je odabrati da li želimo podatke koji zadovoljavaju filter, ne zadovoljavaju ili sve. Potrebno je odabrati sve.

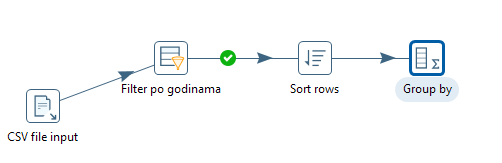


Slika 13: Izlaz podataka iz filtera

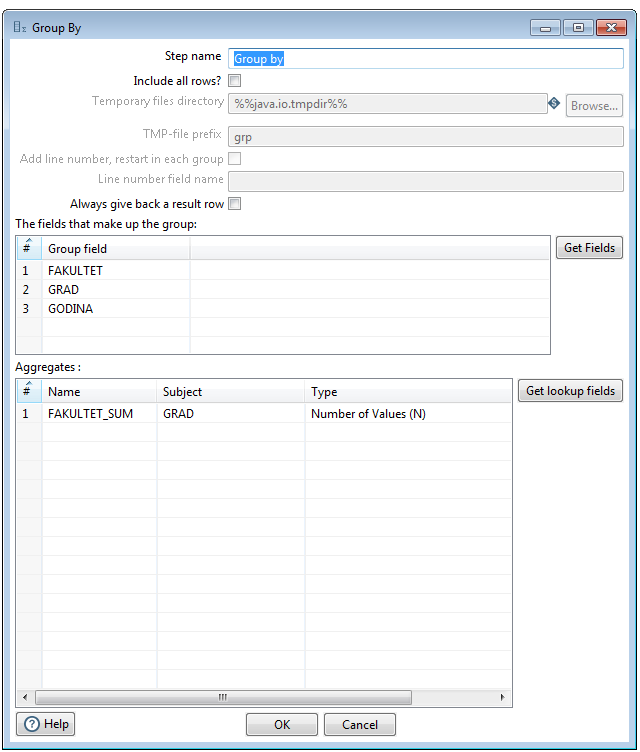


Slika 14: Sortiranje

Sledeći korak je grupisanje podataka. Group by nalazi se pod Statistics. U ovom primjeru za grupisanje uzimaju se polja FAKULTET, GRAD i GODINA. Dodaje se agregatna funkcija sumiranja po gradovima za fakultete.

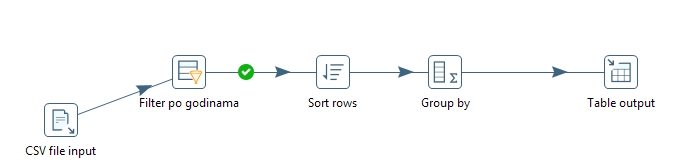


Slika 15: Dodavanje grupisanja



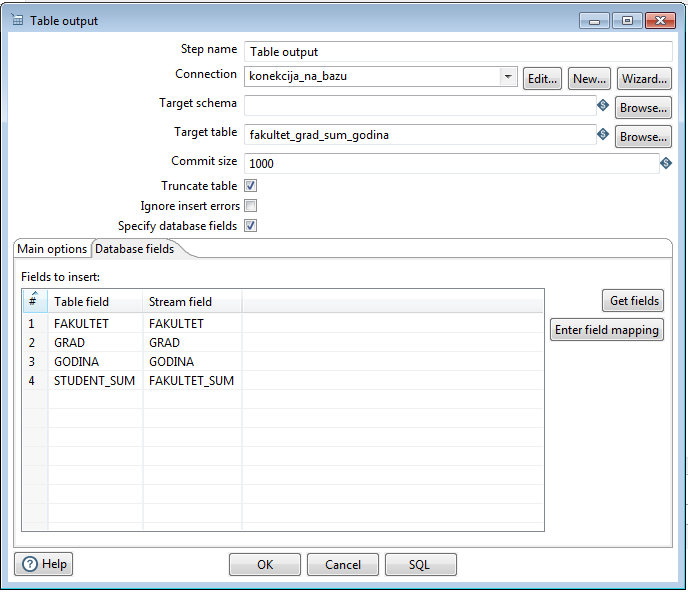
Slika 16: Grupisanje (ispravit)

Sada je potrebno kreirati tabelu u već kreiranoj bazi. Neka je ime tabele bude FAKULTET\_GRAD\_GODINA sa kolonama FAKULTET, GRAD, FAKULTET\_SUM, GODINA, ID\_STUDENT. Potrebno je dodati novu stavku Table output koja se nalazi pod Output.



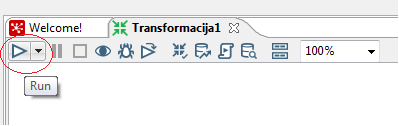
Slika 17: Dodavanje izlaza

U table output potrebno je izabrati konekciju koja ja kreirana na početkum, odgovarajuću bazu i izmapirati polja iz aplikacije u polja iz baze. Svaki naredni put kad se bude koristio *Kettle* potrebno je označiti *Truncate table* da bi se brisali stari podaci iz *MySQL* tabele, ali prvi put ostaje neoznačen jer postoji problem kada je prazna baza!



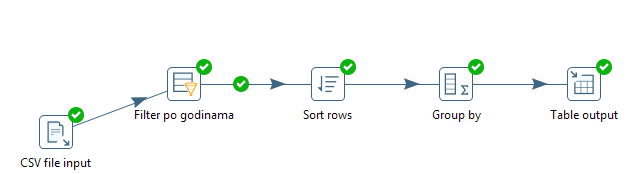
Slika 18: Izlaz

Sada se može pokrenuti punjenje baze.



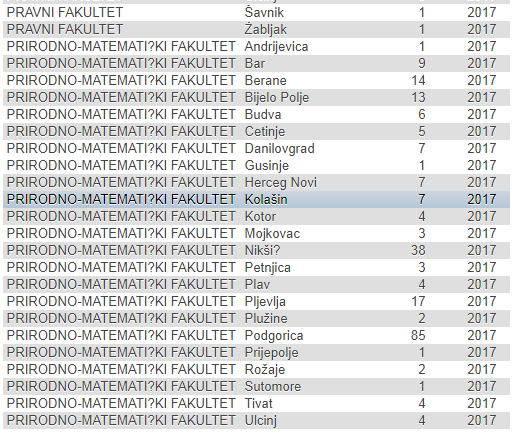
Slika 19: Dugme za izvršavanje

Ako je sve prošlo uspješno prošlo izgledaće kao na slici 20, u suprotnom pojaviće se crvena ikonica na mjestu gdje je došlo do problema.



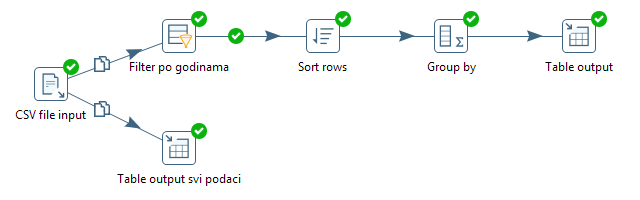
Slika 20: Uspješno izvršena operacija

Kada se pogledaju podaci u bazi jasno je da već imamo lakše podatke za analizu.



Slika 21: Dio podataka iz tabele fakultet\_grad\_sum\_godina

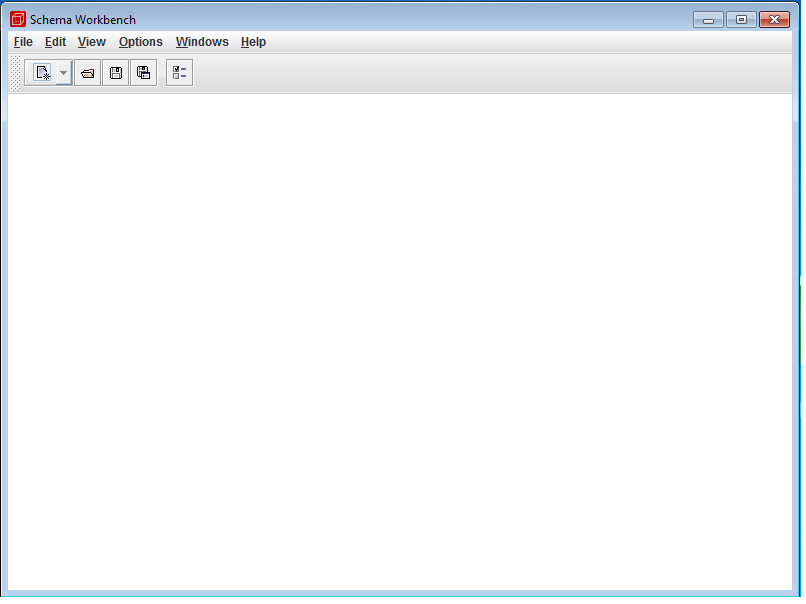
Zbog specifičnosti podataka u ovom radu, nije potrebno podijeliti sve po tabelama i napraviti fizičku zvijezdu, pa je poslednji korak da se svi podaci iz fajla učitaju u jednoj tabeli. Tu tabelu koju cijelu učitamo ćemo kasnije koristiti za rad i uz pomoć Saikua moći ćemo da dobijemo isti rezultat kao u prethodnom primjeru. Za ovaj poslednji korak potrebno je kreirati novu tabelu u bazi, neka se zove SVI\_PODACI koja sadrži sve kolone kao u CSV fajlu K2018\_sa\_id i kolona ID je primarni ključ. Nakon toga potrebno je dodati novu izlaz (Table output), povezati originalni izvor pri čemu je potrebno odabrati opciju kopiranja podataka a ne distribucije i na kraju podesiti u izlazu SQL tabelu SVI\_PODACI i izmapirati kolone, u slučaju da program sam ne može da prepozna ili se drugačije zovu u *SQL* tabeli.



Slika 22: Finalni izgled transformacije nakon presipanja svih podataka

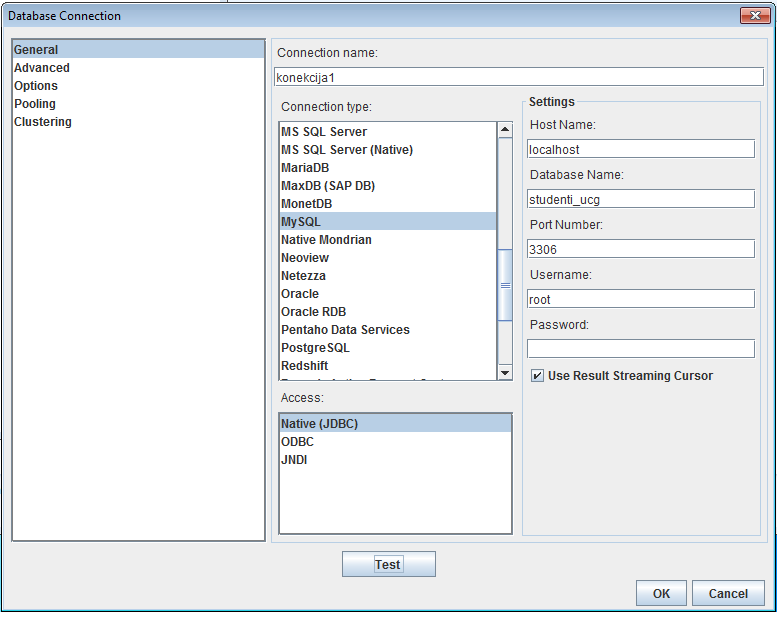
## **5.3. Schema Workbench – kreiranje OLAP kocke**

Prvi korak koji je potreban da se uradi je isti kao i za Kettle a to je da se sa preuzme *mysql-connector-java-\* JDBC* drajver za konekciju sa MySQL bazom i da se smjesti u folderu *lib*. Pokreće se duplim klikom na *workbench.bat*.



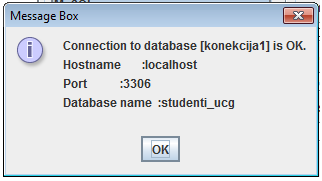
Slika 23: GUI programa Schema Workbench

Takođe je potrebno kreirati konekciju sa postojećom bazom i to je prvi korak nakon pokretanja programa. Stavka za kreiranje konekcije se nalazi u gornjem meniju pod stavkom *Options*, pa nakon toga se ide na *Connections*. Potrebno je odabrati MySQL konekciju i unijeti neophodne parametre.



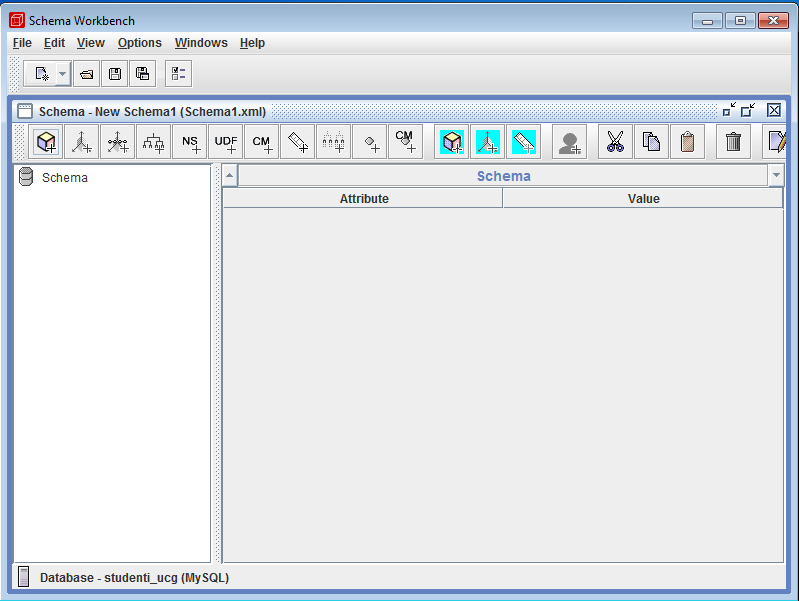
Slika 24: Kreiranje konekcije

Nakon unešenih parametara potrebno je testirati konekciju. Ako je sve u redu prikazaće se poruka o uspješnoj konekciji kao na slici 24 u suprotnom potrebno je provjeriti da li je izabran odgovarajući drajver za konekciju i da li je pokrenut MySQL server.



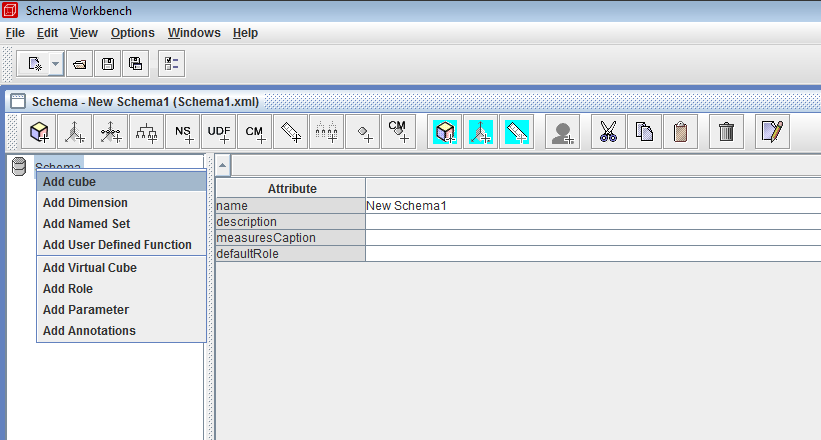
Slika 25: Uspješna konekcija

Sledeći korak je da se kreira šema. Opcija za kreiranje šeme se nalazi pod *File*, *New* pa *Schema*. Potrebno je imenovati i sačuvati.



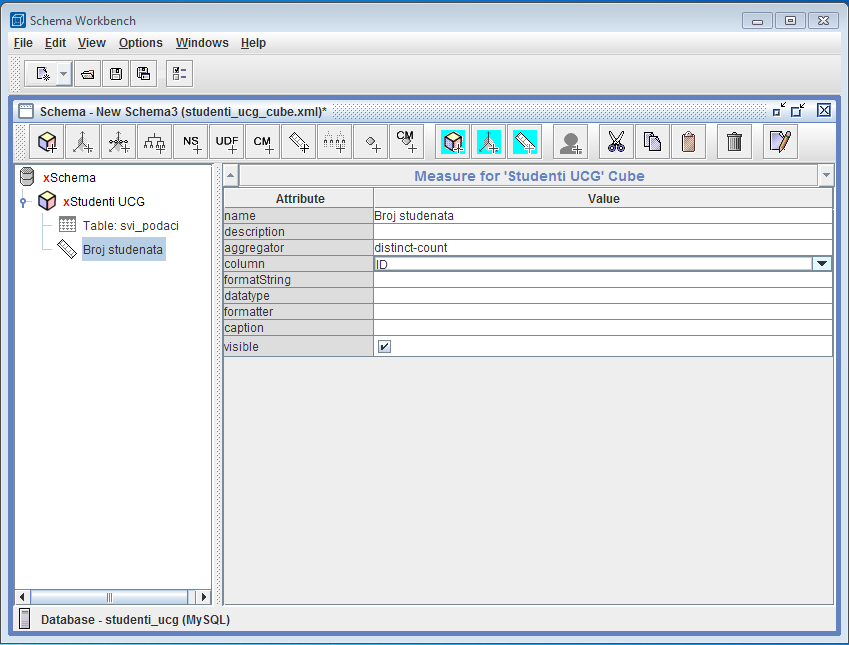
Slika 26: Izgled programa Schema Workbench nakon kreiranja šeme

Sada se može krenuti sa kreiranjem kocke. Desnim klikom na *Schema* pa *Add* *cube*.



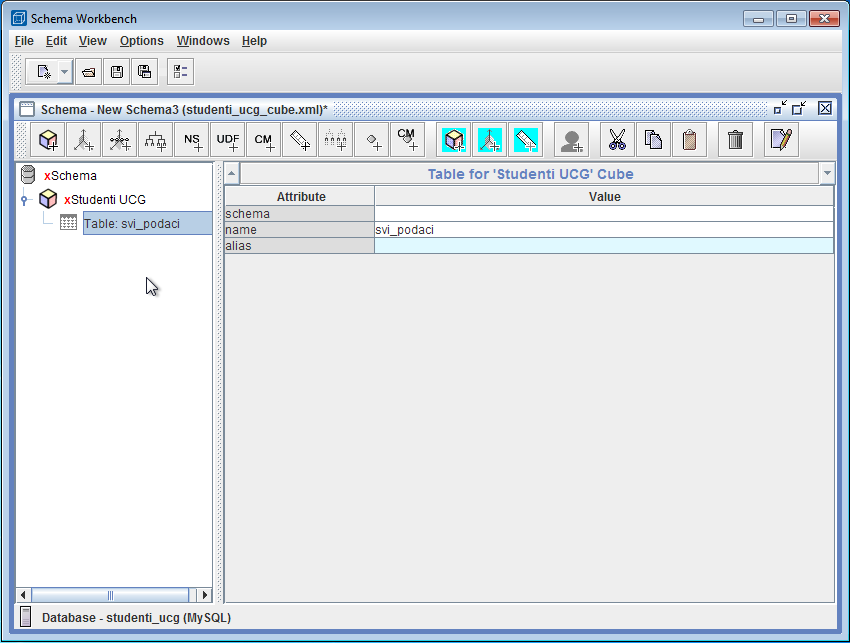
Slika 27: Izgled programa Schema Workbench nakon kreiranja šeme

Sledeći korak je dodavanje tabele. Dodavanje tabele se obavlja na isti način kao i dodavanje kocke, dakle desni klik na kocku pa *Add table*. Za tabelu odabrati tabelu svi\_podaci.



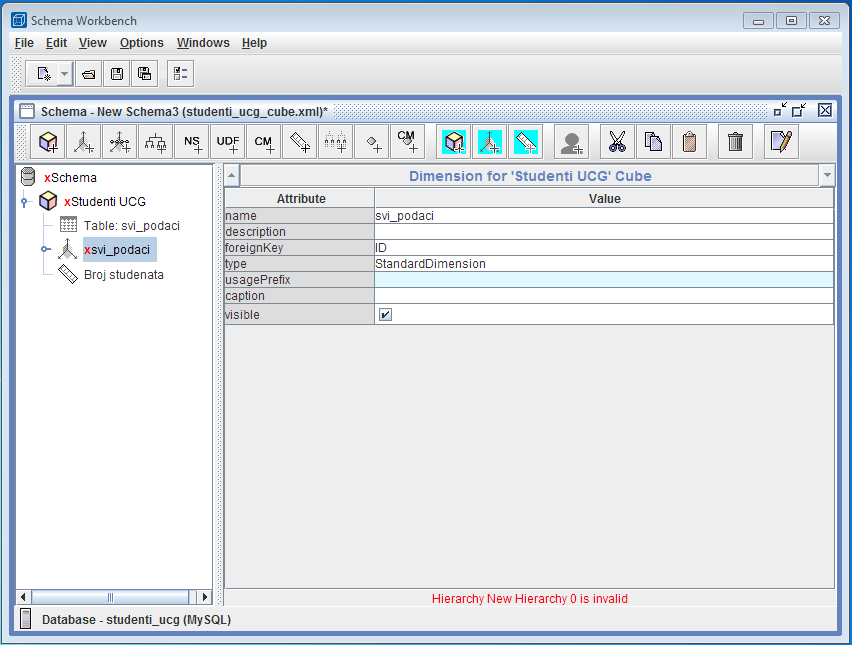
Slika 28: Dodavanje tabele

Neka se kocka zove Studenti UCG. Sledeći korak je dodavanje mjere. Dodaje se desnim klikom na kocku pa *Add measure*. Neka se mjera zove *Broj studenata*. Za kolonu je potrebno odabrati ID a za agregator *distinct-count*.



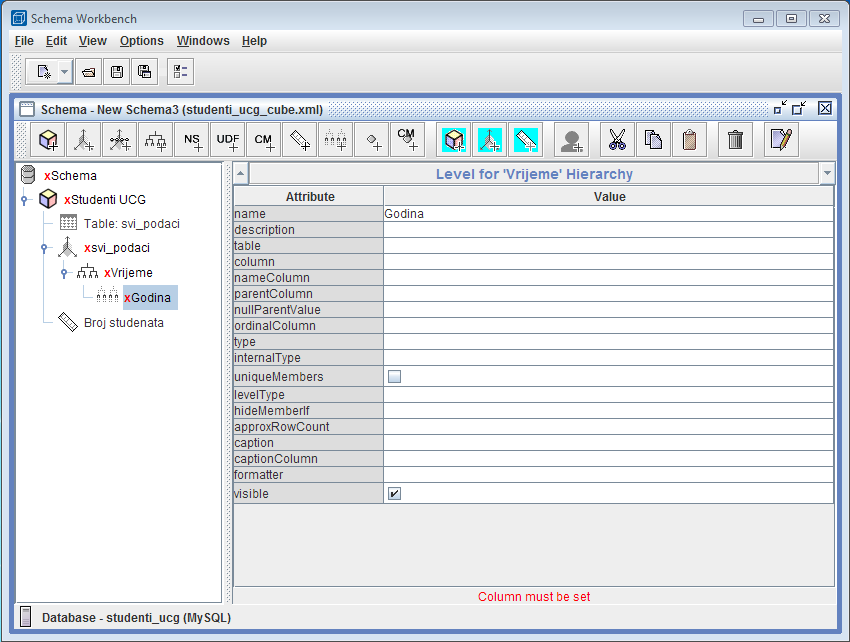
Slika 29: Dodavanje mjere

Nakon mjere potrebno je dodati dimenziju. Pošto je zbog novije verzije došlo do razlike u XML formatu mora se naknadno ručno izmijeniti fajl da bi se dodali atribut XML elementi. Dodaje se desnim klikom na kocku pa *Add dimension*. Neka se dimenzija zove svi\_podaci.



Slika 30: Dodavanje dimenzije

Nakon dimenzije dodaje se hijerarhija. Neka se hijerarhija zove *Vrijeme*. Za hijerarhiju dodaje se *level* (nivo) i neka se taj *level* zove *Godine*. Cilj je hijerarhijski prikaz po godinama.



Slika 31: Dodavanje hijerarhije i nivoa

**NAPOMENA**: Pošto je XML element *Attribute* dodat u *Mondrian schema* verziji 4.0 a *Schema Workbench* je još uvijek u verziji 3.x, potrebno je ručno editovati XML fajl. Potrebno je ručno izmijeniti i podatke o tabeli.

Zbog navedenih razlika u verzijama *Mondrian schema* *i Schema Workbench* nastavak kocke radiće se ručno. Trenutni rad u *Schema Workbench* potrebno je sačuvati i neka bude sačuvan pod imenom studenti\_ucg\_cube.xml. Prvi korak je premještanje XML elementa *table* u novom XML elementu zvanom *PhysicalSchema*. Takođe potrebno je dodati i ključ za tabelu svi\_elementi. Ovaj dio XML fajla treba da bude sledeći:

<**PhysicalSchema**>

<**Table** name="*svi\_podaci*" schema="">

<**Key**>

<**Column** name="*ID*" />

</**Key**>

</**Table**>

</**PhysicalSchema**>

Zatim se dimenzije premještaju pod novim XML elementom zvanim *Dimensions*. Za postojeću dimenziju atribut *foreignKey* se preimenuje u *key* i za atribut *type* postavlja se ,,*TIME''*.

<**Dimensions**>

<**Dimension** key="*ID*" name="*svi*\_*podaci*" table="*svi\_podaci*" visible="*true*" type="*TIME*">

</**Dimension**>

</**Dimensions**>

Pod dimenzijom svi\_podaci potrebno je dodati novi XML element *Attributes* koji treba da sadrži sve kolone iz tabele svi\_podaci (ID, GODINA, FAKULTET, GRAD, TIP\_STUDIJA, STUDIJSKI\_PROGRAM, ROK, POL, DRZAVA). Oblik novih elemata je sledeći:

<**Attributes**>

<**Attribute** name="*ID*" keyColumn="*ID*" hasHierarchy="*false*" />

<**Attribute** name="*Godina*" hasHierarchy="*true*">

<**Key**> <**Column** name="*GODINA*" /> </**Key**>

<**Name**> <**Column** name="*GODINA*" /> </**Name**>

</**Attribute**>

...

</**Attributes**>

Nakon dimenzije potrebno je prilagoditi Hijerarhiju. Hijerarhija se ubacuje pod postojećom dimenzijom i pod novim XML elementom *Hierarchies* koji se nalazi na istom nivou kao i *Attributes*.

<**Hierarchies**>

<**Hierarchy** name="*Vrijeme*" hasAll="*true*">

<**Level** attribute="*Godina*">

<**Annotations**>

<**Annotation** name="*AnalyzerDateFormat*">[yyyy]</**Annotation**>

</**Annotations**>

</**Level**>

</**Hierarchy**>

</**Hierarchies**>

Posldenja stvar koju treba izmijeniti su mjere. Potrebno je kreirati novi XML element *MeasureGroups* u istom nivou kao i *Dimensions*. Pod novokreiranim elementom potrebno je ubaciti novi XML element *MeasureGroup* i pod njim staviti postojeću mjeru. U istom nivou sa mjerom potrebno je dodati još jedan novi XML element *DimensionLinks* koji sadrži fakt vezu sa dimenzijom nad kojom ćemo da radimo.

<**MeasureGroups**>

<**MeasureGroup** name="*Studenti*" table="*svi\_podaci*">

<**Measures**>

<**Measure** aggregator="*distinct-count*" column="*ID*" name="*Broj studenata*" visible="*true*" />

</**Measures**>

<**DimensionLinks**>

<**FactLink** dimension="*svi*\_*podaci*" />

</**DimensionLinks**>

</**MeasureGroup**>

</**MeasureGroups**>

Nakon ovih izmjena konačni sadržaj XML fajla je sledeći:

<?**xml** version="*1.0*" encoding="*UTF-8*" standalone="*yes*"?>

<**schema** metamodelVersion="*4.0*" name="*UCG\_studenti*">

<**Cube** name="*Studenti UCG*">

<**Dimensions**>

<**Dimension** key="*ID*" name="*svi\_podaci*" table="*svi\_podaci*" visible="*true*" type="*TIME*">

<**Attributes**>

<**Attribute** name="*ID*" keyColumn="*ID*" hasHierarchy="*false*" />

<**Attribute** name="*Godina*" hasHierarchy="*true*">

<**Key**> <**Column** name="*GODINA*" /> <**/Key**>

<**Name**> <**Column** name="*GODINA*" /> <**/Name**>

<**/Attribute**>

<**Attribute** name="*Fakultet*" hasHierarchy="*true*">

<**Key**> <Column name="*FAKULTET*" /> <**/Key**>

<**Name**> <Column name="*FAKULTET*" /> <**/Name**>

<**/Attribute**>

<**Attribute** name="*Grad*" hasHierarchy="*true*">

<**Key**> <**Column** name="*GRAD*" /> <**/Key**>

<**Name**> <**Column** name="*GRAD*" /> <**/Name**>

</**Attribute**>

<**Attribute** name="*Tip studija*" hasHierarchy="*true*">

<**Key**> <**Column** name="*TIP\_STUDIJA*" /> </**Key**>

<**Name**> <**Column** name="*TIP\_STUDIJA*" /> </**Name**>

</**Attribute**>

<**Attribute** name="*Studijski program*" hasHierarchy="*true*">

<**Key**> <**Column** name="*STUDIJSKI\_PROGRAM*" /> </**Key**>

<**Name**> <**Column** name="*STUDIJSKI\_PROGRAM*" /> </**Name**>

</**Attribute**>

<**Attribute** name="*Rok*" hasHierarchy="*true*">

<**Key**> <**Column** name="*ROK*" /> </**Key**>

<**Name**> <**Column** name="*ROK*" /> </**Name**>

</**Attribute**>

<**Attribute** name="*Pol*" hasHierarchy="*true*">

<**Key**> <**Column** name="*POL*" /> </**Key**>

<**Name**> <**Column** name="*POL*" /> </**Name**>

</**Attribute**>

<**Attribute** name="*Drzava*" hasHierarchy="*true*">

<**Key**> <**Column** name="*DRZAVA*" /> </**Key**>

<**Name**> <**Column** name="*DRZAVA*" /> </**Name**>

</**Attribute**>

</**Attributes**>

<**Hierarchies**>

<**Hierarchy** name="*Vrijeme*" hasAll="*true*">

<**Level** attribute="*Godina*">

<**Annotations**>

<**Annotation** name="*AnalyzerDateFormat*">[yyyy]</**Annotation**>

</**Annotations**>

</**Level**>

</**Hierarchy**>

</**Hierarchies**>

</**Dimension**>

</**Dimensions**>

<**MeasureGroups**>

<**MeasureGroup** name="*Studenti*" table="*svi\_podaci*">

<**Measures**>

<**Measure** aggregator="*distinct-count*" column="*ID*" name="*Broj studenata*" visible="*true*" />

</**Measures**>

<**DimensionLinks**>

<**FactLink** dimension="*svi\_podaci*" />

</**DimensionLinks**>

</**MeasureGroup**>

</**MeasureGroups**>

</**Cube**>

<**PhysicalSchema**>

<**Table** name="*svi\_podaci*" schema="">

<**Key**>

<**Column** name="*ID*" />

</**Key**>

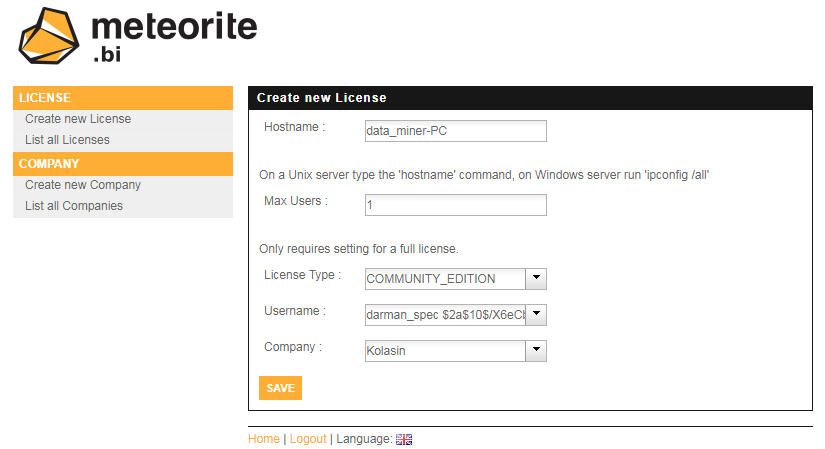
</**Table**>

</**PhysicalSchema**>

</**schema**>

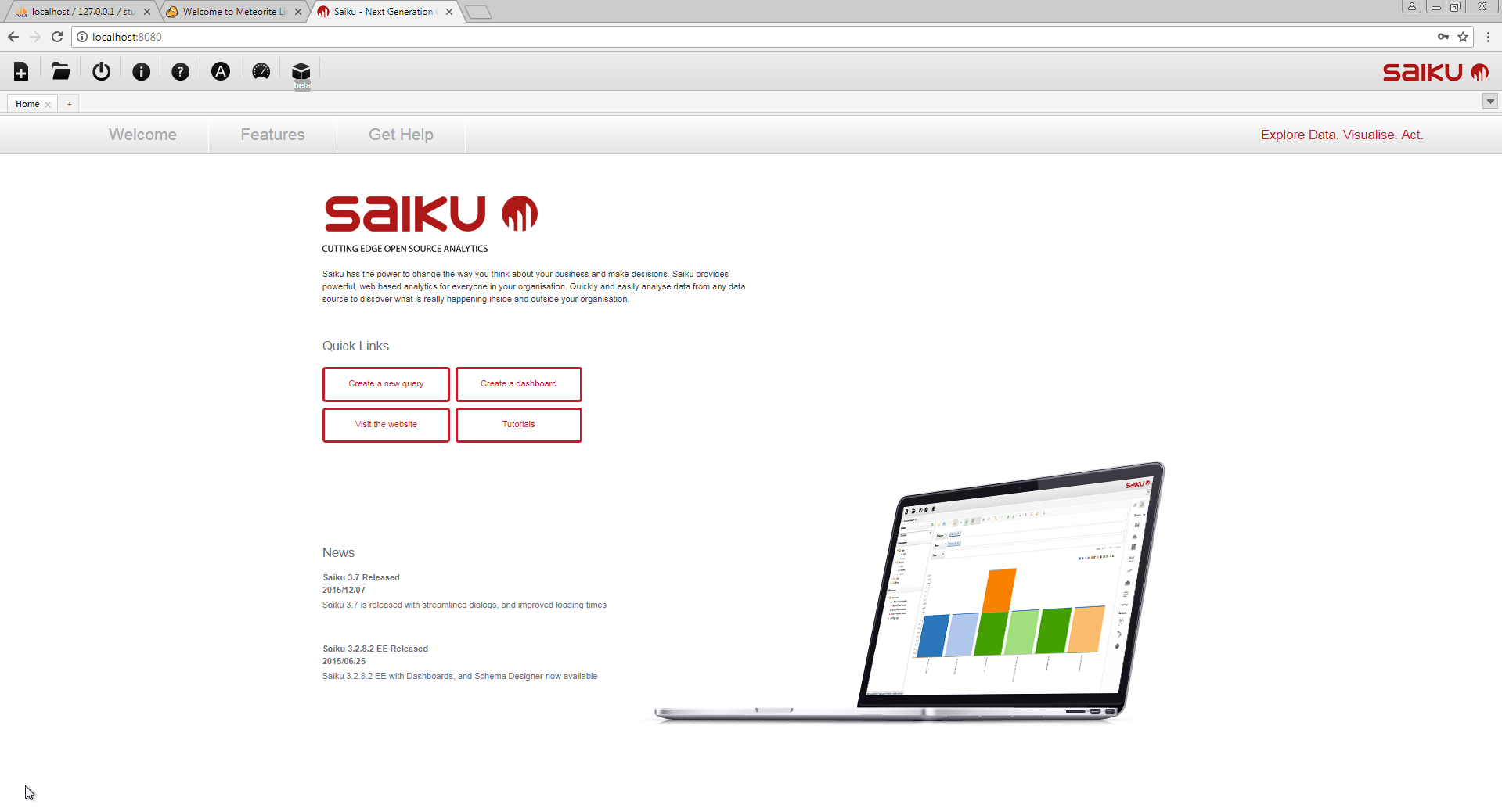
## **5.4. Saiku**

Saiku se može preuzeti sa hitachivantara.com. Prije instalacije potrebno je registrovati se i izabrati licencu. Licenciranje se vrši na adresi *licensing.meteorite.bi*. Za potrebe ovog rada i sličnih dovoljna je *community* verzija koja je besplatna. Prilikom kreiranja licence potrebno je obratiti pažnju na polje *Hostname* jer tu mora da bude precizno navedeno ime računara. Nakon kreiranja licence potrebno je sačuvati na računaru jer prilikom instalacije biće zahtijevana.



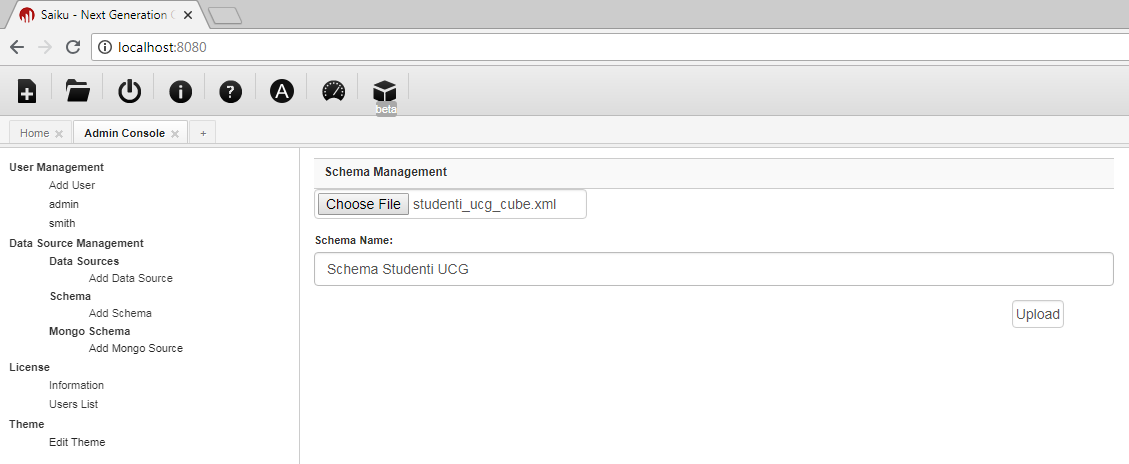
Slika 32: Kreiranje licence

Instalacija Saikua sadrži i upakovan tomcat server. Instalacija se obično nalazi pod C:\Program Files\Saiku Enterpise. Pokreće se duplim klikom na *start-saiku.bat* u folderu server. Treba sačekati dok se u konzoli ne javi poruka ,,*INFO: Server startup in XXXXX ms*''. Nakon toga u internet pretraživaču se pokreće Saiku klijent i obično se nalazi na *localhost:8080*. Standardni korisnik je admin kao i lozinka.



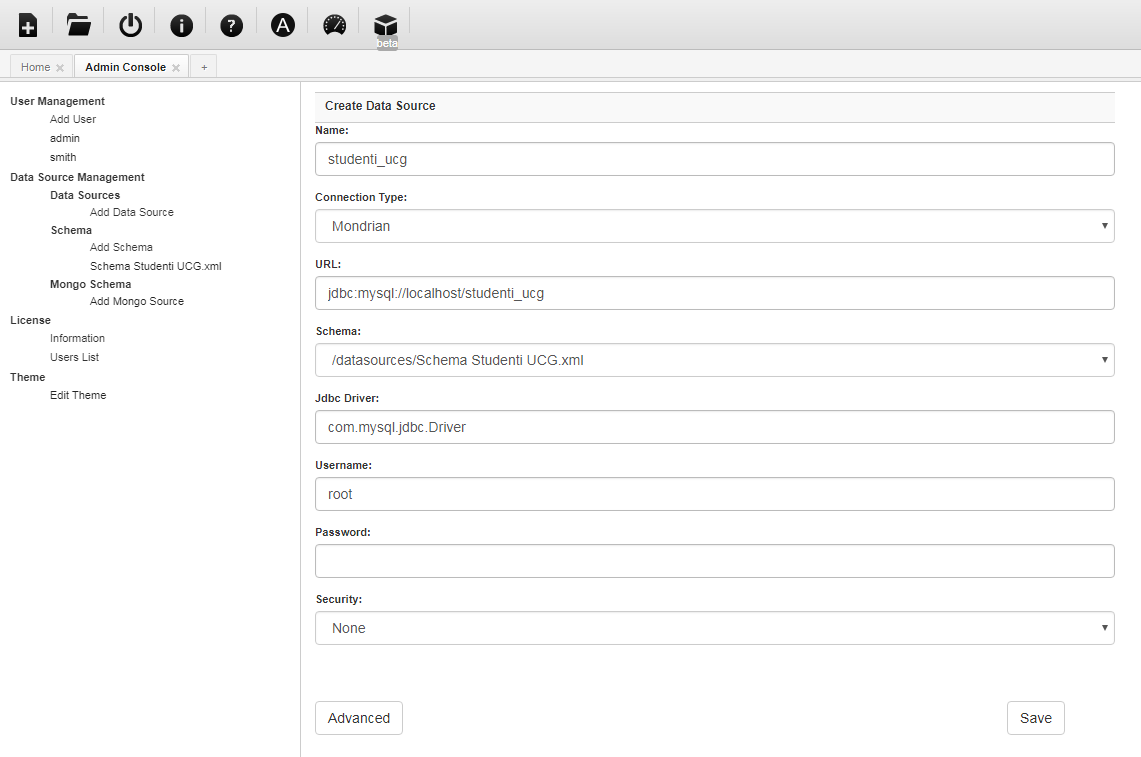
Slika 33: Saiku

Sledeći korak je dodavanje OLAP kocke, tj XML prethodno kreiranog fajla. To se radi u administratorskom dijelu koji se pokreće preko dugmeta sa oznakom ,,*A*'' (treće sa desna u meniju na vrhu). Kocka se dodaje tako što se prvo klikne na *Add Schema*. Potrebno je odabrati prethodno kreiranu kocku i nazovimo je *Schema Studenti UCG*.



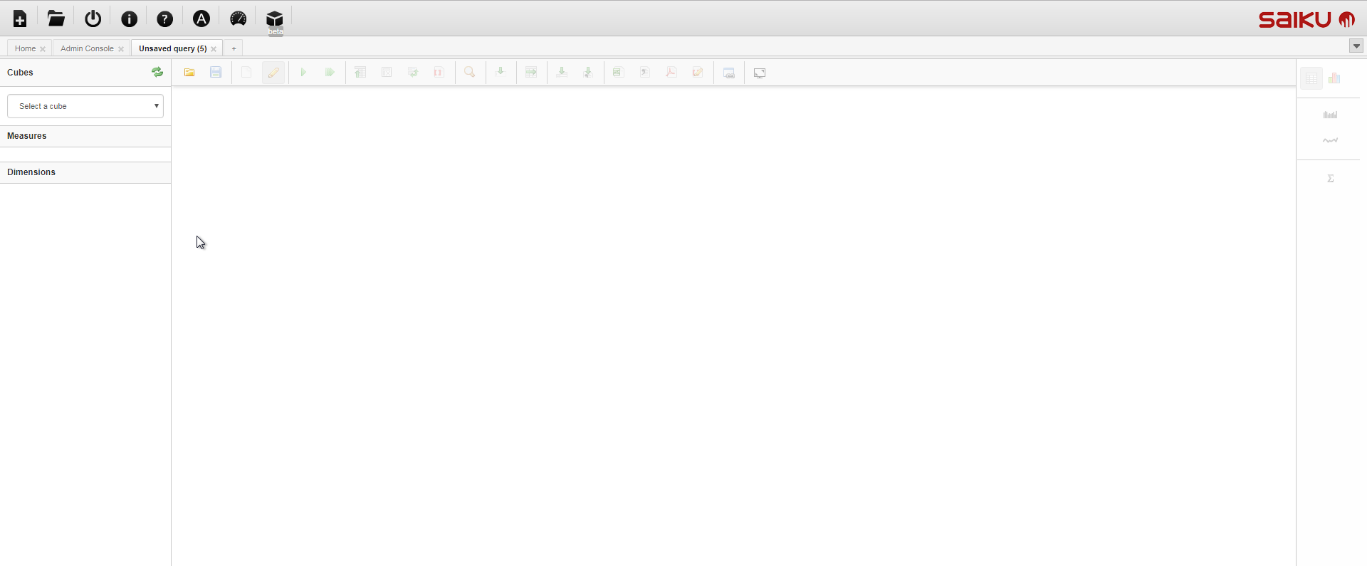
Slika 34: Dodavanje kocke u Saiku

Nakon uspješno dodate kocke potrebno je dodati izvor podataka. Izvor podatak se dodaje klikom na *Add Data Source* stavku i upisivanjem odgovarajućih parametara. Neka se izvor zove studenti\_ucg, tip konekcije je Mondrian, putanja do baze se upisuje sa *jdbc:mysql://localhost/studenti\_ucg* jer koristimo mysql bazu, zatim se selektuje prethodno dodata kocka, za drajver je potrebno upisati *com.mysql.jdbc.Driver* i na kraju za korisnika i šifru potrebno je unijeti iste parametre kao za već postojeći mysql.



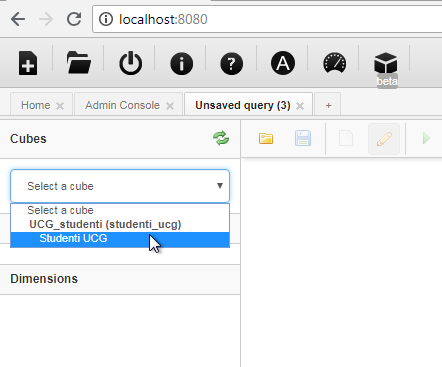
Slika 35: Dodavanje izvora podataka

Nakon uspješno dodatog izvora podataka i kocke može se početi sa korišćenjem. Nakon dodavanje nove kocke preporučljivo je da se osvježi pretraživač i izbriše keš kako bi mogli da izaberemo kocku. U slučaju da se kocka ne pojavi, potrebno je ispitati *saiku.log* fajl koji se nalazi pod Saiku Enterpise/server/tomcat/logs. U tom log fajlu sadržane su sve informacije o greškama i da bi se kocka prikazala potrebno je imati spojenu kocku nad izvorom podataka. Klikom na prvo dugme u glavnom meniju dolazimo do dijela za izvršavanje upita i prikaz podataka.



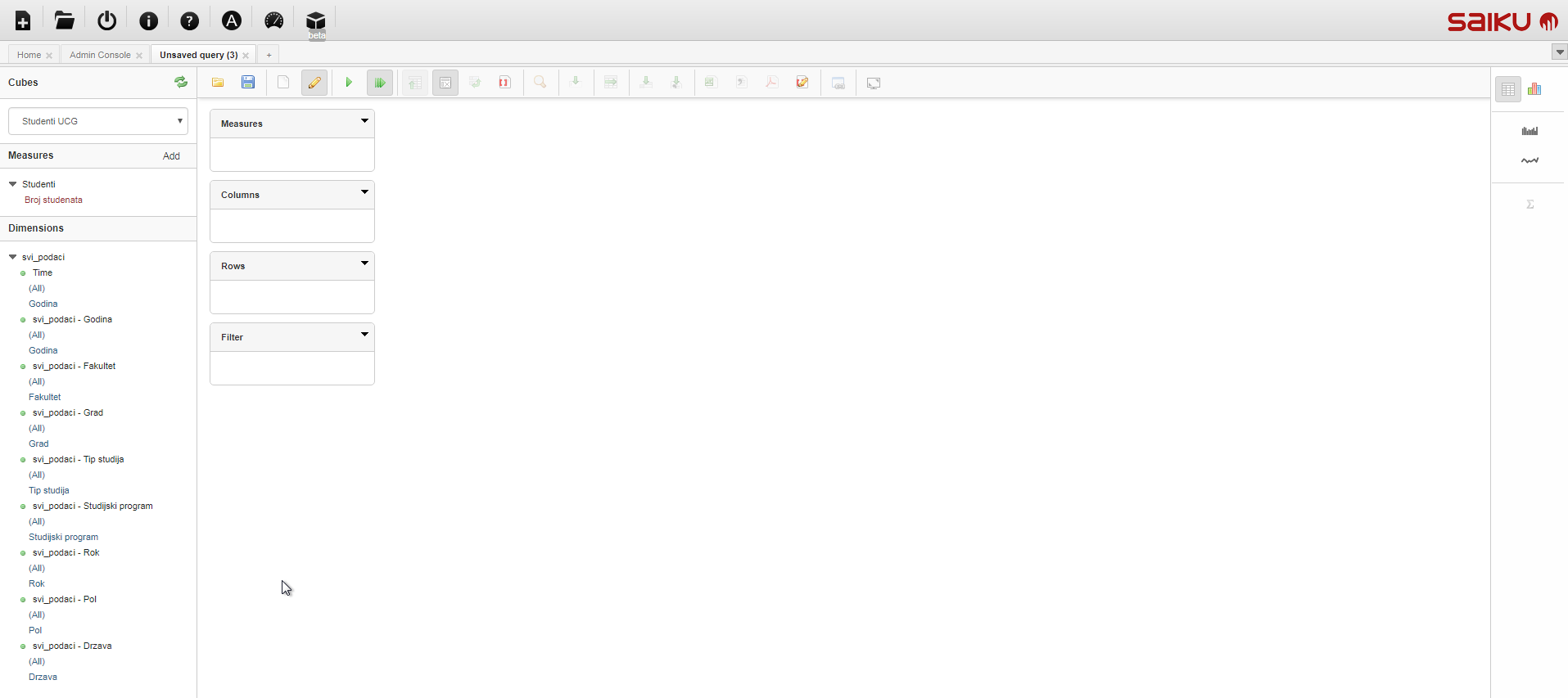
Slika 36: Izgled dijela aplikacije za izvršavanje upita i prikaz podataka

Sada je potrebno odabrati već kreiranu kocku koja je povezana sa izvorom podataka.



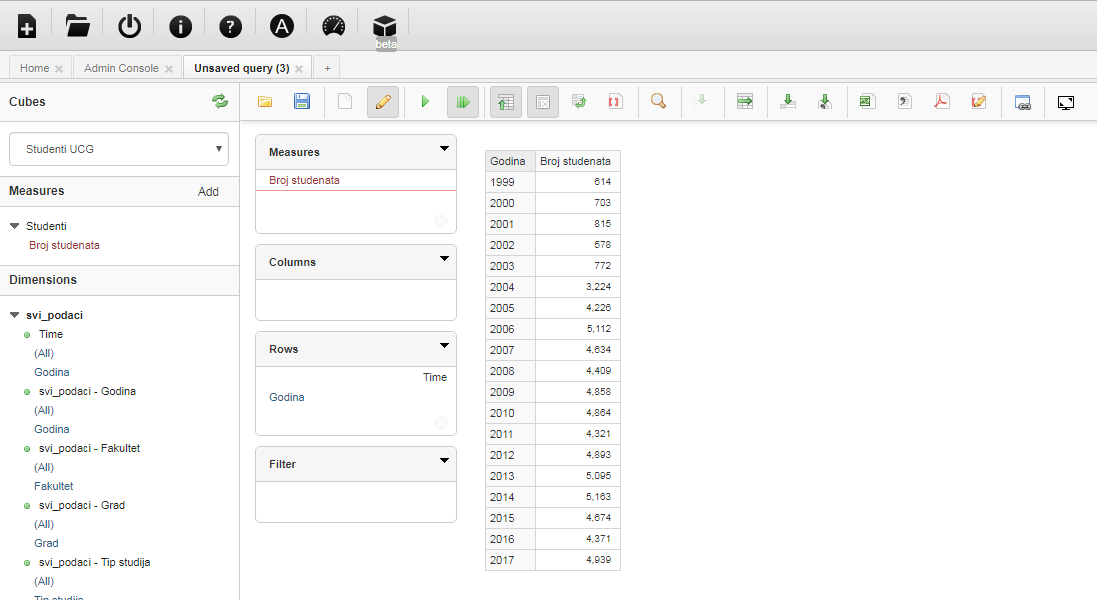
Slika 37: Odabir kocke

Nakon selekcije kocke biće vidljive sve mjere i dimenzije koje se nalaze u njoj.



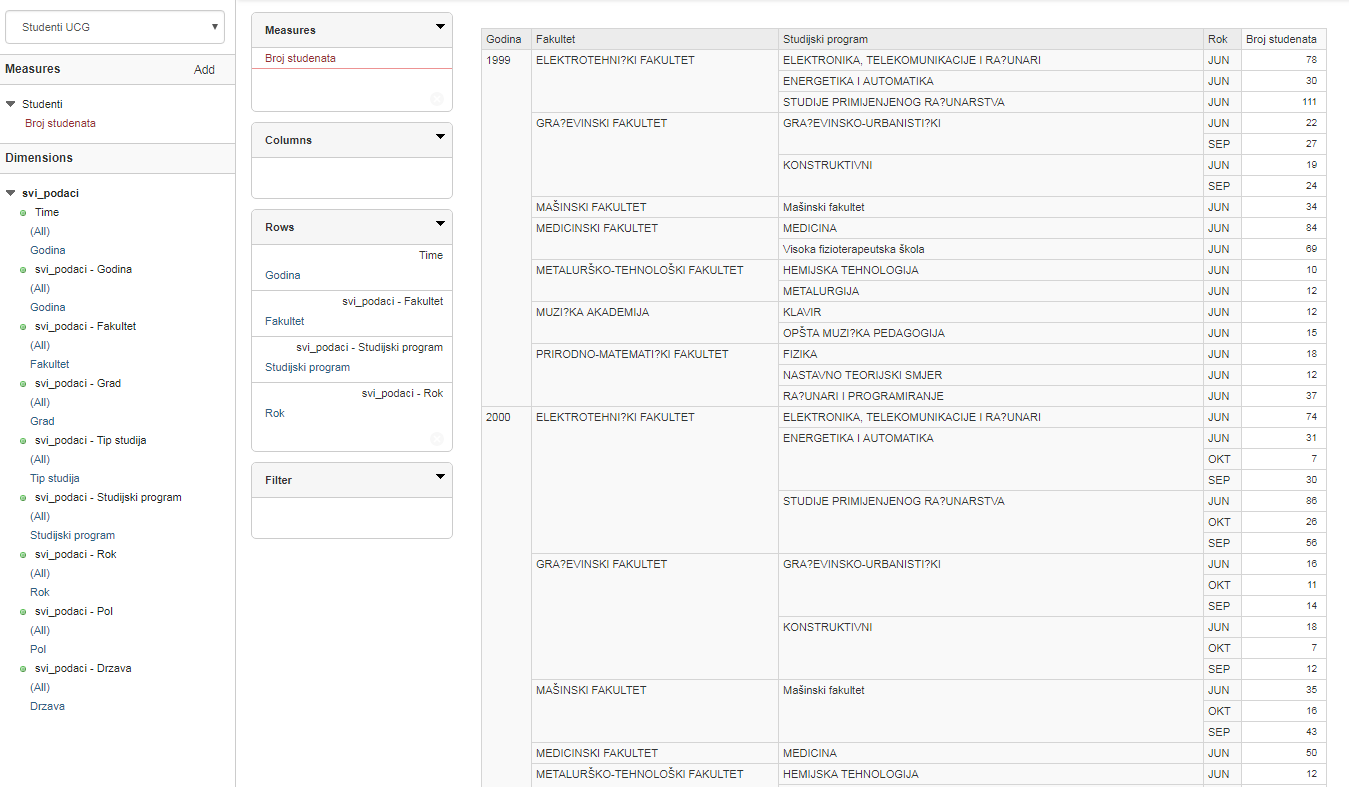
Slika 38: Saiku nakon selekcije kocke sa prikazom mjera i dimenzija

Da bi bilo prikaza podataka potrebno je izabrati jednu mjeru i makar jedan red. Dodavanjem mjere Broj studenata i dimenzije Godine, prikazaće se broj studenata po godinama. Dodavanje mjera i dimenzija moguće je operacijom *drag and drop*.



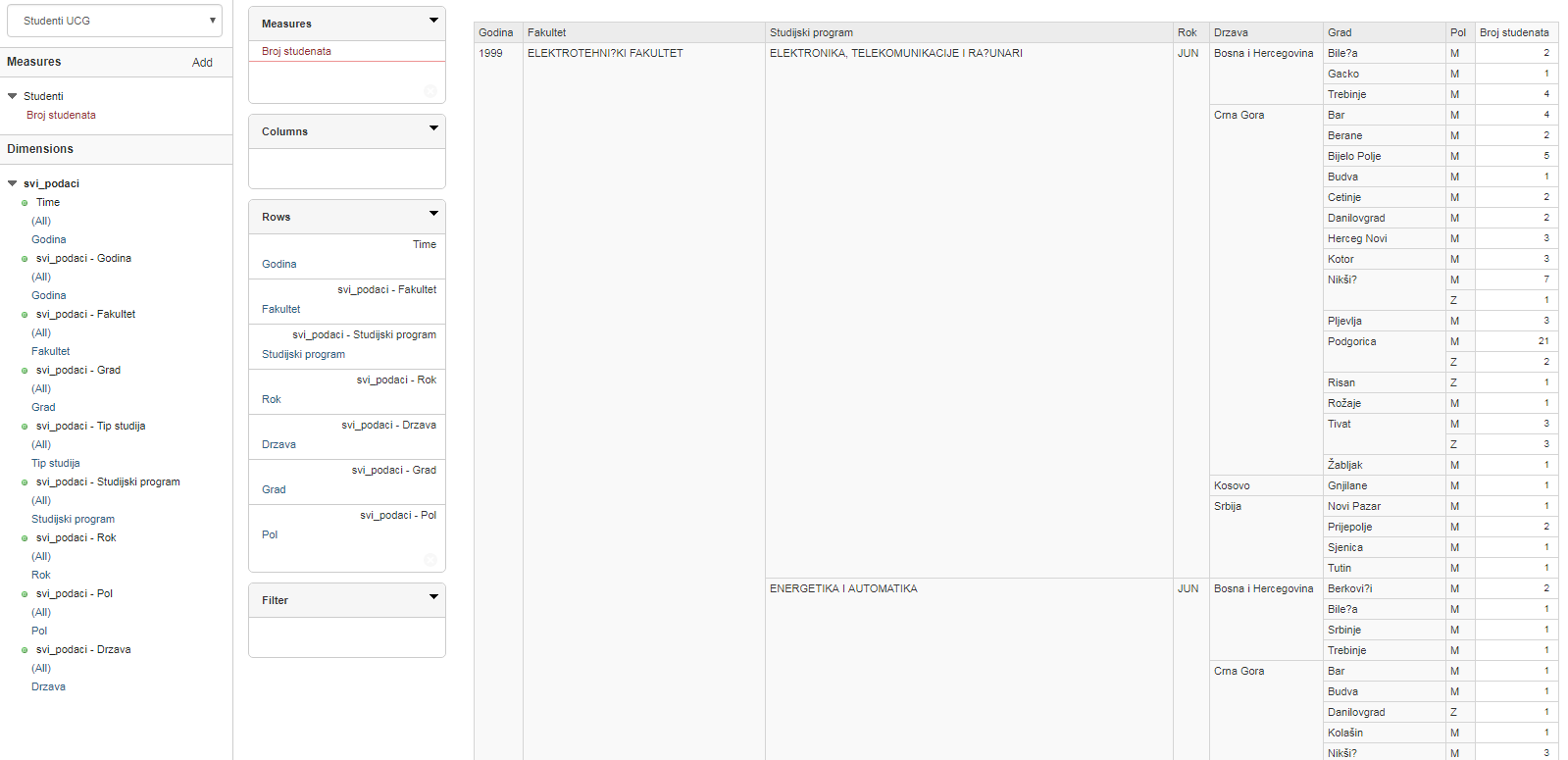
Slika 39: Prikaz broja studenata po godinama

Nakon godina potrebno je dodati dimenzije Fakultet, Studijski Program i Rok. Time se dobijaju informacije o broju upisanih studenata za svaku godinu na kom fakultetu, za koji studijski program i u kojem upisnom roku.



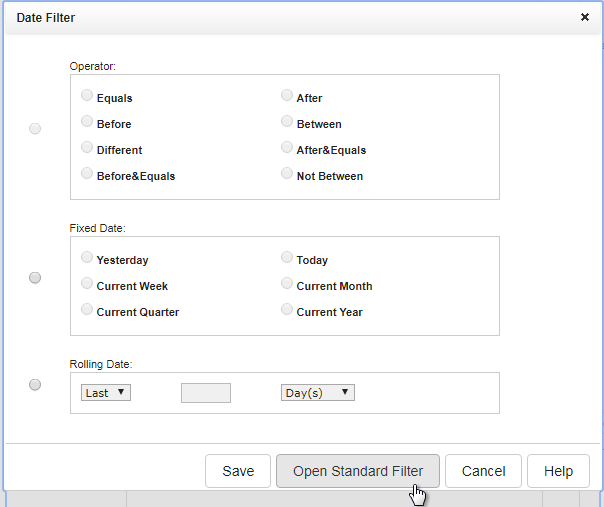
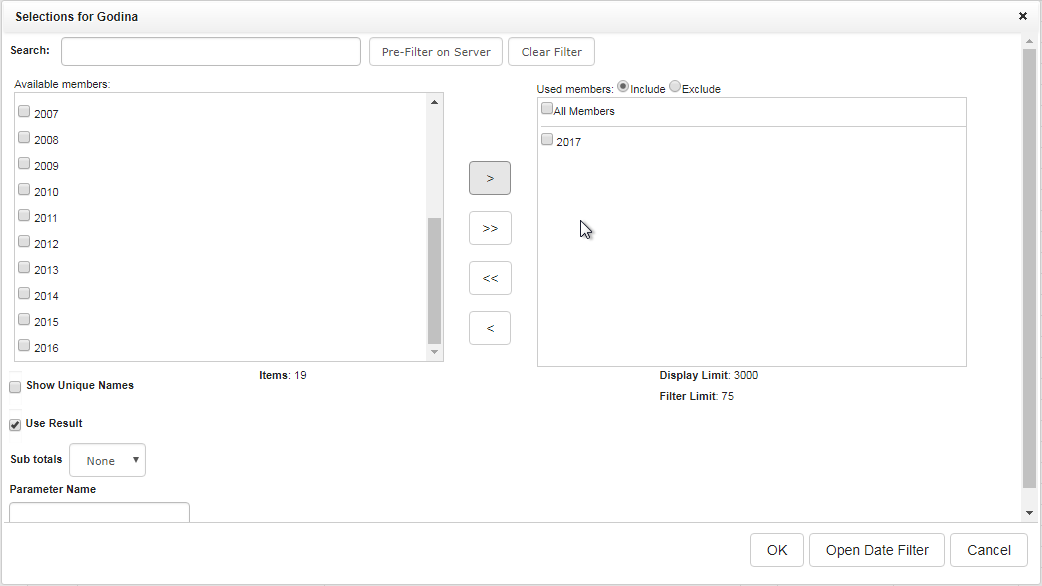
Slika 40: Prikaz broja upisanih studenata po godinama, fakultetima, studijskom programu i upisnom roku

Nakon trenutnih dimenzije potrebno je dodati i dimenzije Država, Grad i Pol čime se prethodnim podacima dodjeljuju još informacije iz koje države, kojeg grada i kojeg su pola upisani studenti.



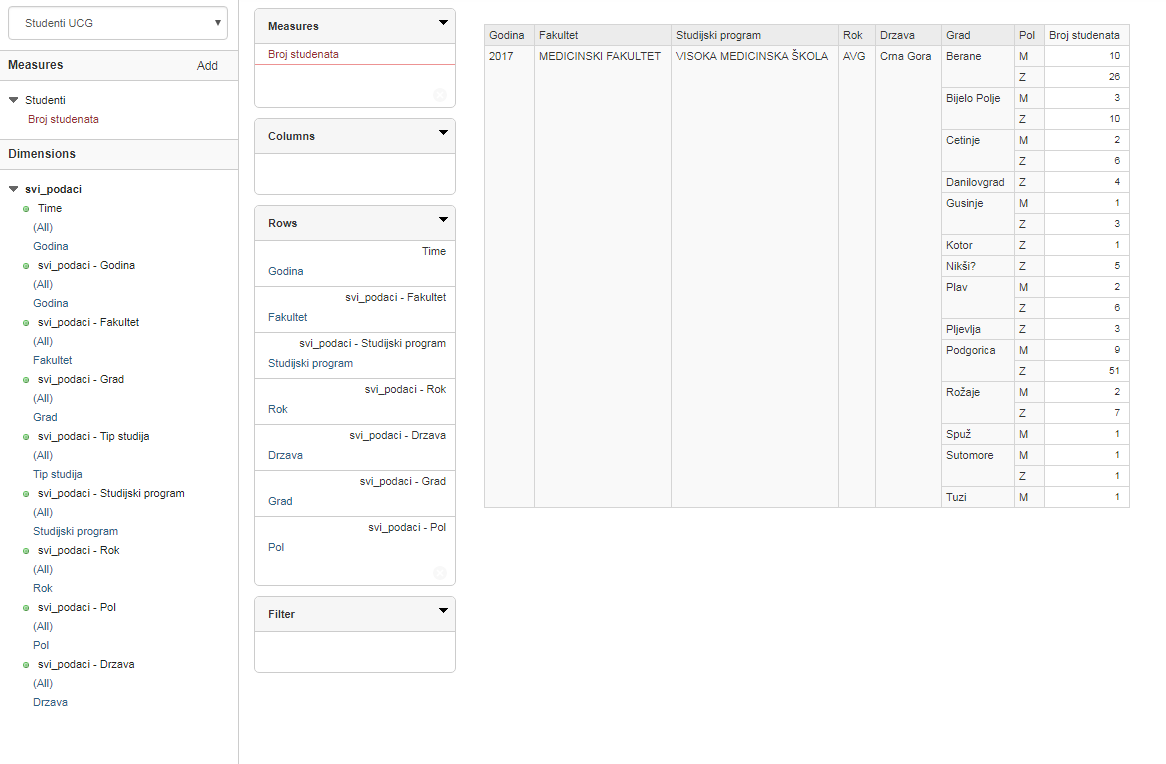
Slika 41: Prikaz broja upisanih studenata po godinama, fakultetima, studijskom programu, upisnom roku, državi, gradu i polu

Podatke je moguće filtrirati. Najbolji način za filtriranje je da se u odgovarajućoj dimenziji izaberu odgovarajući parametri. Duplim klikom na dimenziju otvoriće se filter i klikom na dugme *Open Standard Filter* otvara se standardni filter. U ovom primjeru neka budu izabrani podaci samo za 2017. godinu.



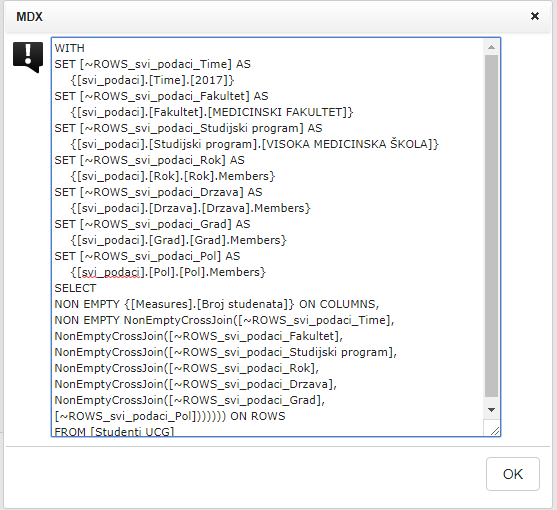
Slika 42: Filter Slika 43: Standradni filter i odabir 2017 godine

Nakon godine potrebno potrebno je odabrati Medicinski fakultet za dimenziju Fakultet i Visoka medicinska škola za dimenziju Studijski program. Ovim smo dobili podatke o Visokoj medicinskoj školi.



Slika 44: Prikaz podataka upisanih studenata za Visoku medicinsku školu za 2017. Godinu

Klikom na dugme *Show MDX* moguće je vidjeti generisani MDX upit.



Slika 45: Prikaz MDX upita za prethodni primjer

Za sledeći primjer potrebno je prikazati broj upisanih studenata PMF-a preko grafika od 1999. godine do 2017. godine. Za mjeru se uzima Broj studenata, za dimenziju se uzima Godine i dimenizija Fakultet je potrebna da se stavi pod sekcijom za kolone, radi vizuelno privljačnijeg prikaza. Za dimenziju fakultet potrebno je postaviti filter za Prirodno-matematički fakultet. Klikom na *Chart Mode* podaci se prikazuju grafički i postoje različiti načini grafičkog prikazivanja. Na dobijenom prikazu na slici 46 jasno se vidi porast popularnosti Prirodno-matematičkog fakulteta.



Slika 46: Grafički prikaz broja upisanih studenata na Prirodno-matematičkom fakultetu od 1999. do 2017. Godine.

Saiku omogućaca izvoz i tabelarnog i grafičkog prikaza u PDF, XLS i XLS fajlovima.

# **6. Zaključak**

Primjenom moćnih Data Mining alata i dobro definisanim strukturama i kockom, prilagođenom samo za potrebe krajnjeg korisnika moguće je omogućiti čak i laicima da izvlače kompleksne izvještaje i vrše analizu bez poznavanja SQL-a. Rad je rađen nad podacima Univerziteta Crne Gore o upisanim studentima. Cijela programska realizacija se nalazi na virtuelnoj mašini i može koristiti na kursevima Analitičke obrade podataka kao i radnicima Univerziteta za izvlačenje i analizu podataka.

Jedini problem sa izabranim alatima je razlika u verzijama od *Schema Workbench* koja je još uvijek na 3.x dok je *Mondrian schema* verzija 4.0 pa je potrebno ručno doraditi kocku. To je razlika koja postoji u vremenu pisanja rada, pa pošto su Pentaho alati od skoro vlasništvo Hitachi grupe može se očekivati da će se u skorijoj budućnosti riješiti.

# **Literatura**

* Kumar, Steinbach, Tan – Introduction to Data Mining
* Sæmundur Melstað - Reykjavik University Data Warehouse
* Ardhendu Tripathy, Kaberi Das - A Descriptive Approach towards Data Warehouse and OLAP Technology: An Overview
* <https://www.hitachivantara.com>
* <https://www.meteorite.bi>