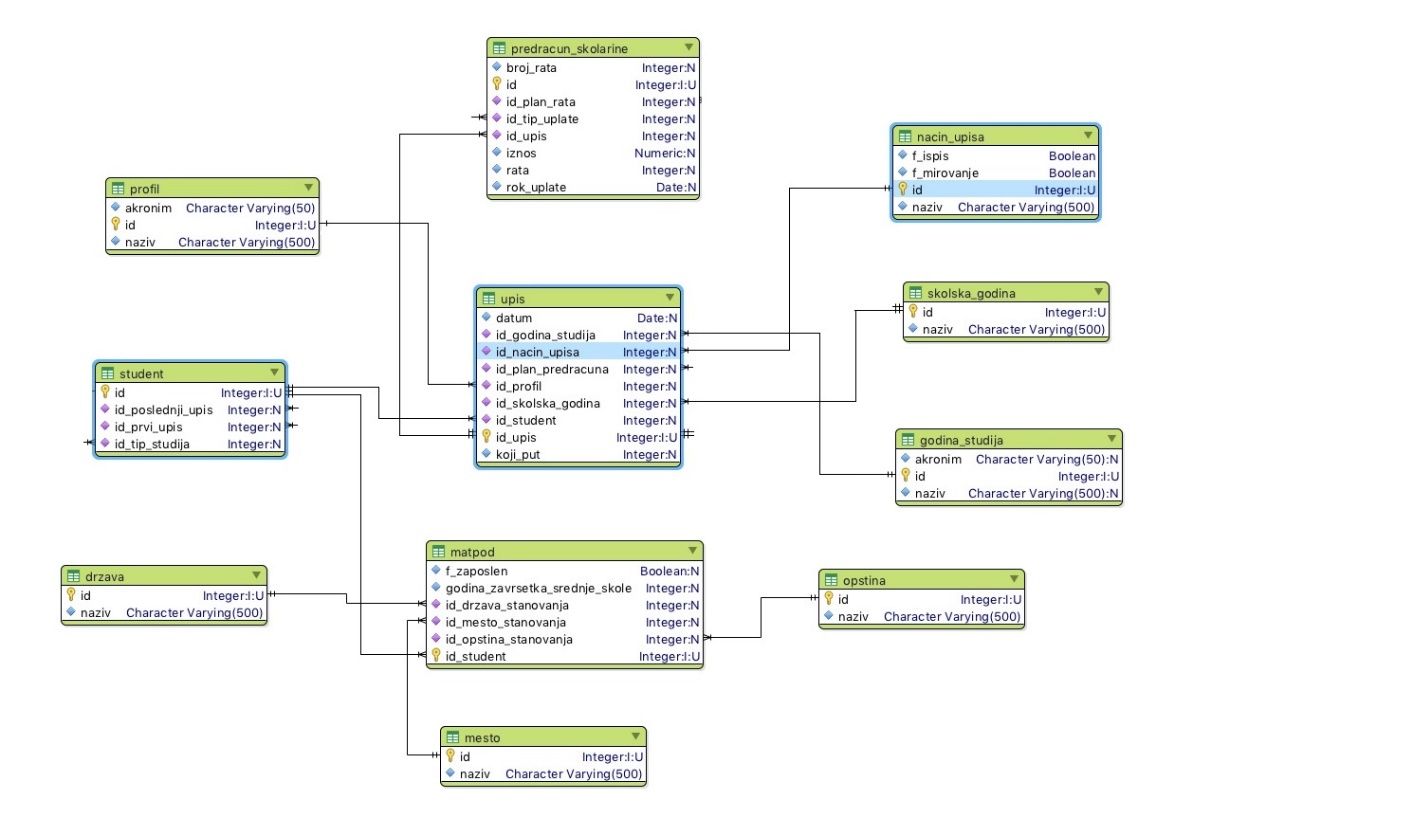
Kako ispravno dizajnirati skladište podataka?

Proces dizajniranja se svodi na sledeća četiri koraka:

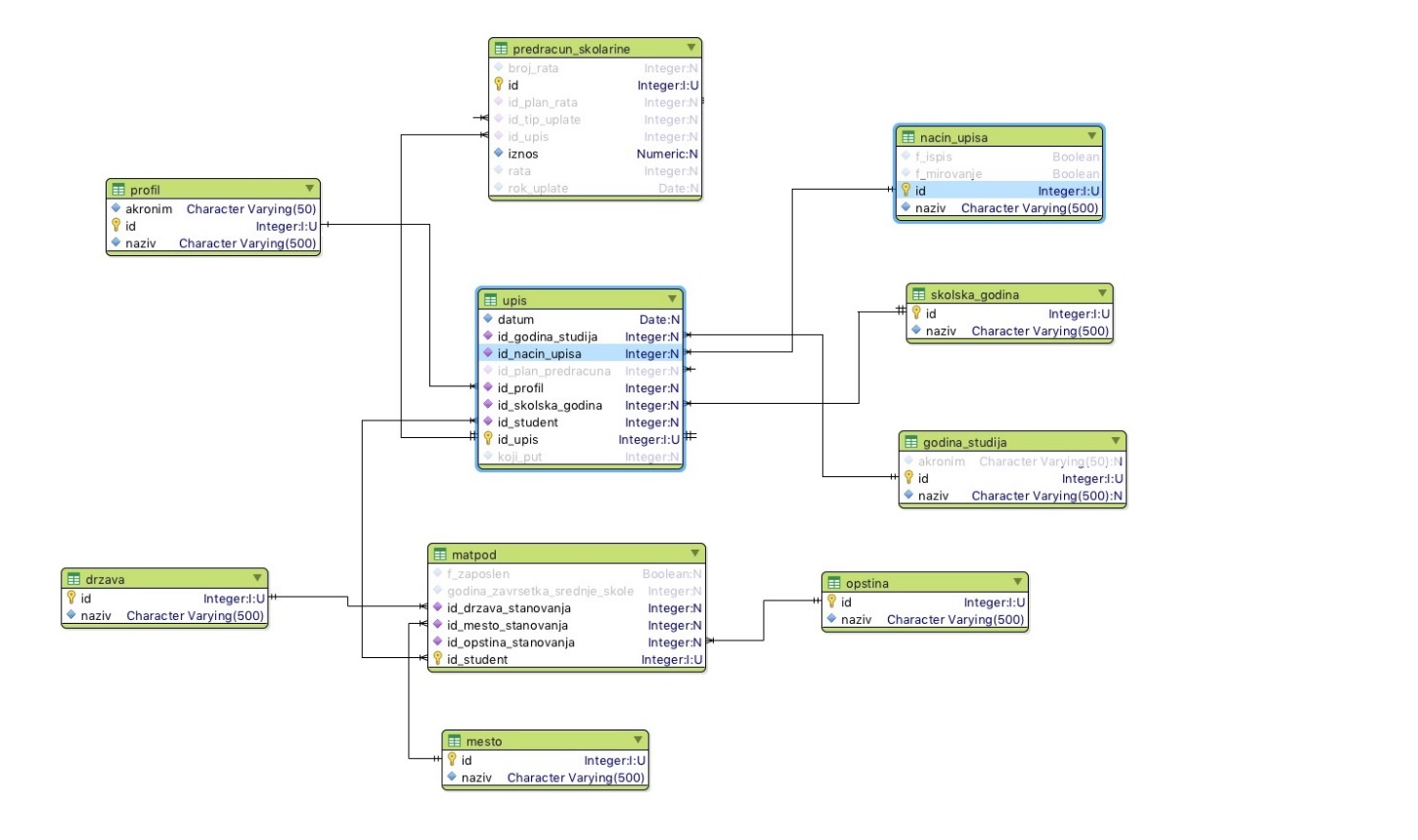
1. Izaberite poslovni proces koji se analizira (npr.prodaja, proizvodnja, utrošak i sl.) kao i glavni cilj tj. šta treba da dobijete kao rezultat.
2. Izaberite srž (centar) procesa koji pratite tj. najveći nivo usitnjavanja.
3. Izaberite dimenzije koje ćete pratiti (vremenski period, lokacija, proizvodi i sl.)
4. Izaberite tabele podataka tj. ono što merite npr. prodati komadi, ostvarena dobit i ukupna prodata vrednost robe.

Kroz sledeći primer ćemo prikazati primer dizajniranja skladišta. Analiziraćemo poslovni proces upisa, kao deo informacionog sistema fakulteta. Na slici ispod je prikazana šema operativne baze koja se odnosi na proces upisa, koji nam je od interesa. (slika 1)



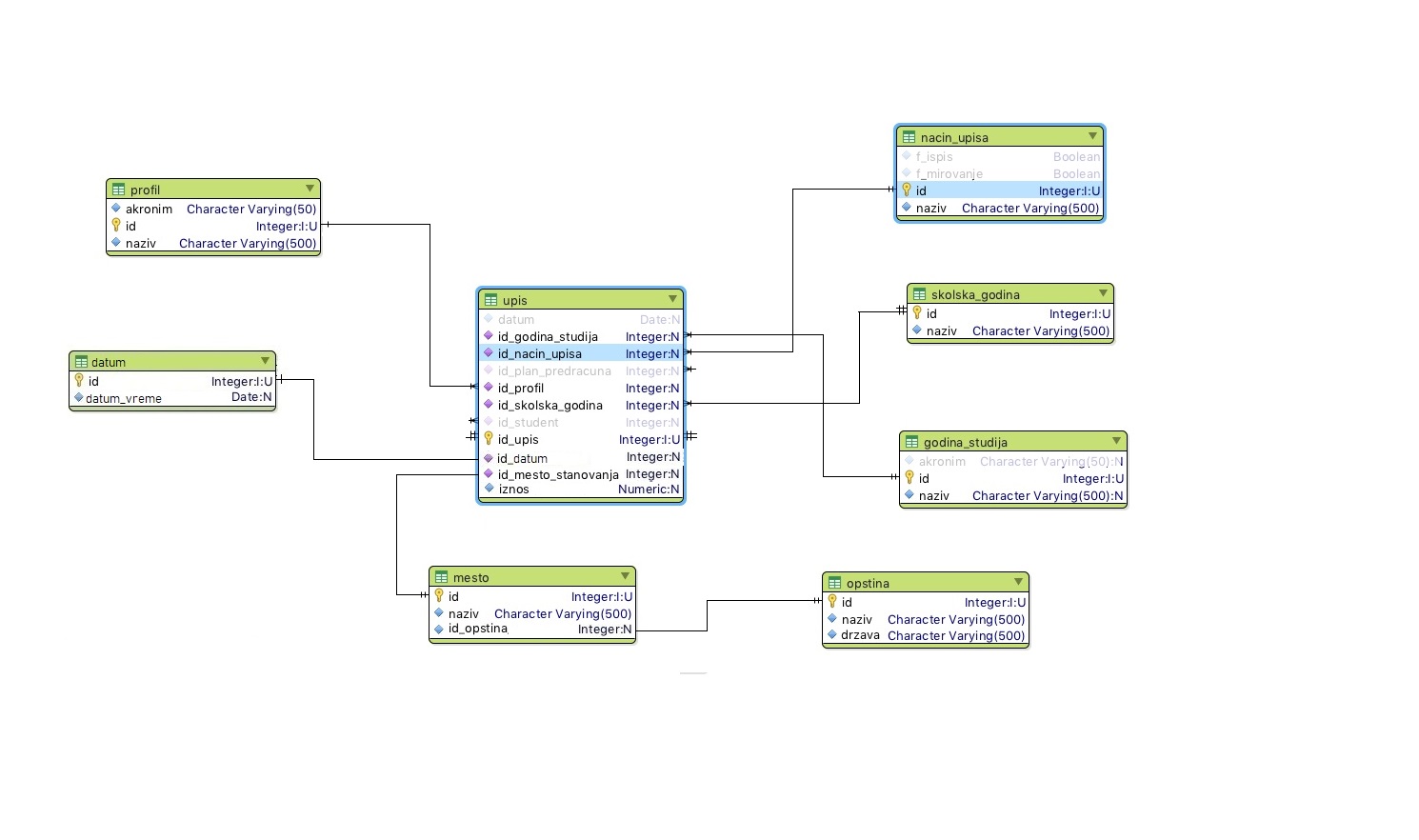
Slika -- ŠEMA DELA OPERATIVNE BAZE KOJI SE ODNOSI NA UPIS

Primetićemo da neke kolone (odnosno strani ključevi u tabeli) nemaju veze ka drugim tabelama (npr. *id\_plan\_predracuna* u tabeli *upis*). Ove kolone ukazuju na tabele koje nisu od interesa za našu analizu, pa ih odbacujemo u daljem dizajniranju skladišta podataka. Takođe, odbacujemo sve one kolone koje nam nisu u interesa u analizi procesa upis. (slika 2)



Slika –Modelovanje skladišta podataka 1

S’ obzirom da želimo da vršimo analizu upisa po datumu, kreiramo novu tabelu u skladištu podataka u koju mapiramo sve različite datume izvršenja uplate. Takođe, s’ obzirom da želimo da vršimo analizu prema mestu stanovanja studenta koji se upisao (hijerarhijski mesto, opština, država stanovanja), vršimo određene promene u relaciji tablela u skladištu podataka:

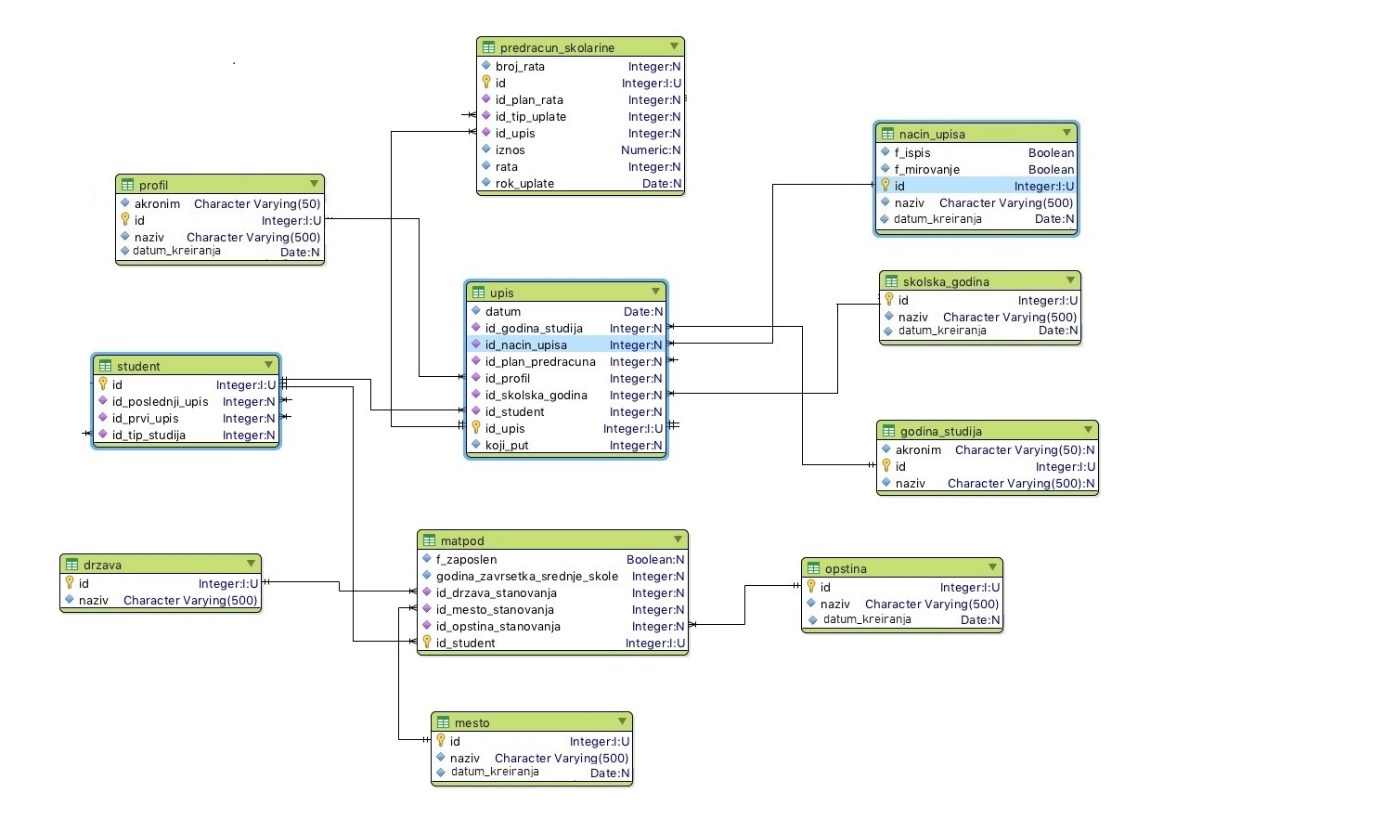


Slika - Modelovanje skladišta podataka 2

Na slici iznad imamo šemu skladišta podataka pomoću koje lako možemo vršiti analizu upisa po sledećim dimenzijama:

* + datum upisa
  + profil koji je upisan
  + mesto-opstina-država stanovanja studenta koji se upisao
  + godina studija u koju se student upisao
  + školska godina u koju se student upisao
  + nacin upisa studenta

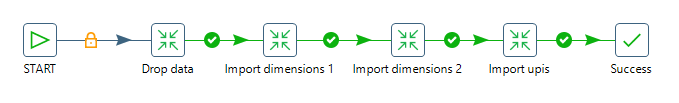
NAPOMENA: U skladištu podataka postoji još jedna tabela, poslednja\_izmena, u kojoj čuvamo datum poslednjeg ažuriranja skladišta podataka. U skladu sa tim, promenicemo strukturu operativne baze, odnosno dodaćemo kolonu datum\_kreiranja u one tabele koje je potrebno ažurirati u skladištu podataka, te struktura dela operativne baze koji posmatramo izgleda:



Slika - Dodavanje kolone datum\_kreiranja u tabelama dimenzije (u operativnoj bazi)

Kako popuniti skladište podataka podacima iz operativne baze?

Skladište podataka punimo izvršavanjem „posla“ koji se sastoji iz niza transformacija kojim podatke iz operativne baze filtriramo, vršimo upite nad njima, spajamo i prosljeđujemo ih u skladište. Kako su podaci iz nekih tabela zavisni od podataka iz drugih tabela (npr. prvo moramo imati popunjene sve tabele dimenzija da bismo popunjavali tabelu upis, odnosno prvo moramo popuniti tabelu opstina pa tek onda popunjavamo tabelu mesto- tabela mesto se puni podacima u transformaciji Import dimensions 2), najpre se izvršavaju transformacije koje nemaju ulaznu zavisnost. Pre svih transformacija „punjenja“, želimo da izbrišemo podatke iz skladišta kako ne bismo imali fatalnu grešku prilikom punjenja – duplikat istog primarnog ključa. Red izvršavanja tranfsormacija je prikazan na sledećoj slici:

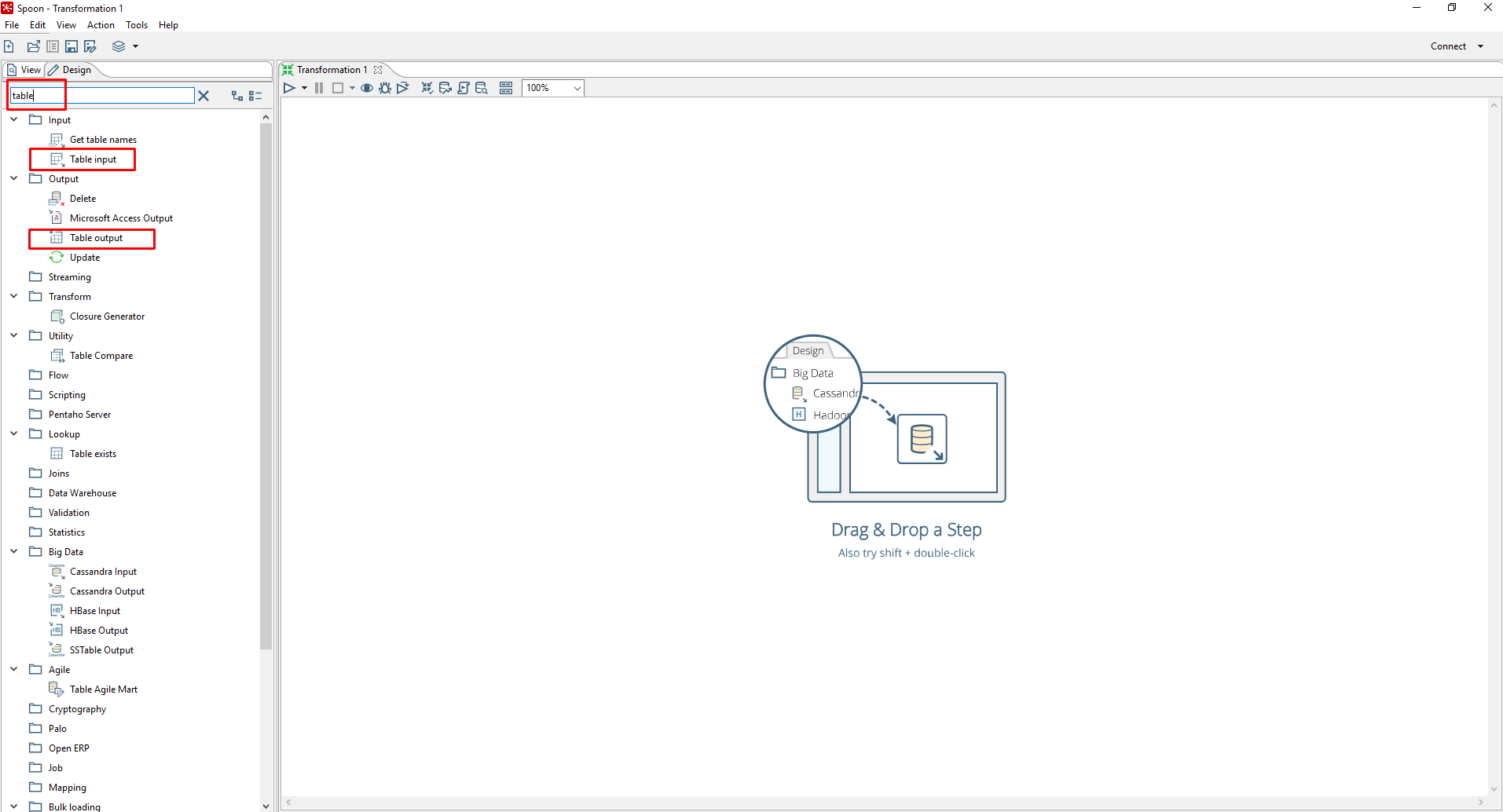


Slika - Redosled izvršavanja transformacija tokom totalnog punjenja

Kako kreirati transformaciju?

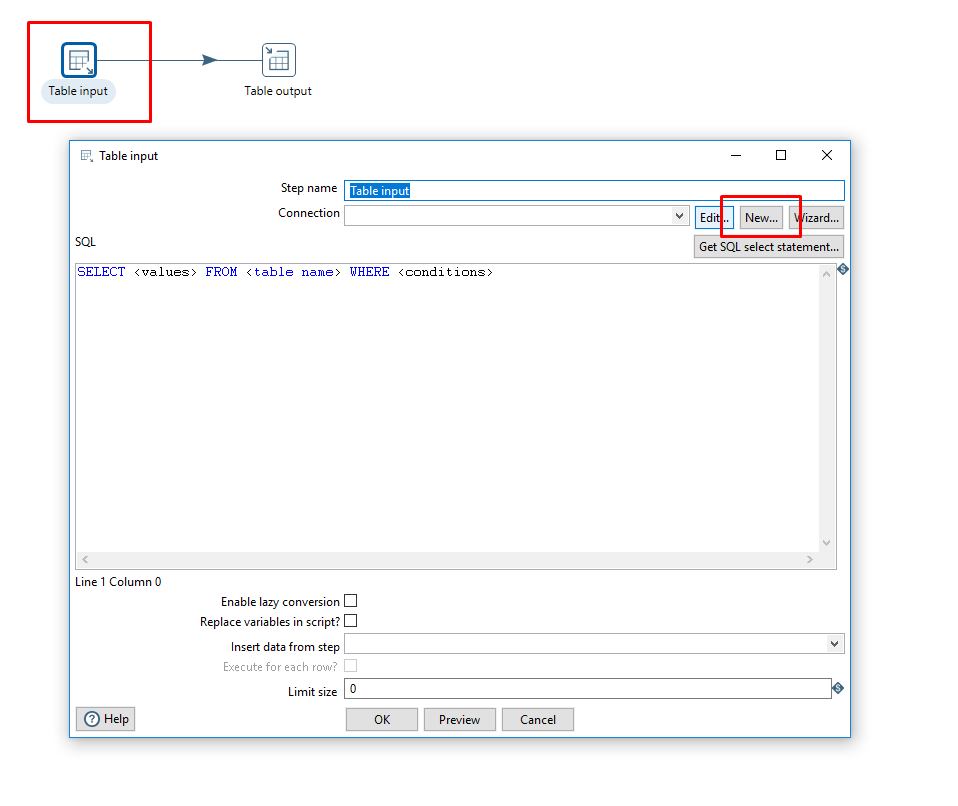
Za kreiranje transformacija koristimo alat *Spoon* koji možete preuzeti na [linku](http://si4is2.etf.rs/Nastava/Vezbe/3.%20trecina/linkovi%20do%20alata).

Kako bismo izvršili prenos podataka iz tabele iz operacione baze u tabelu skladišta podataka, moramo napraviti transformaciju koja se sastoji stavki od *Table input* (izvor podataka) I *Table output* (tabela gde se smeštaju podaci).



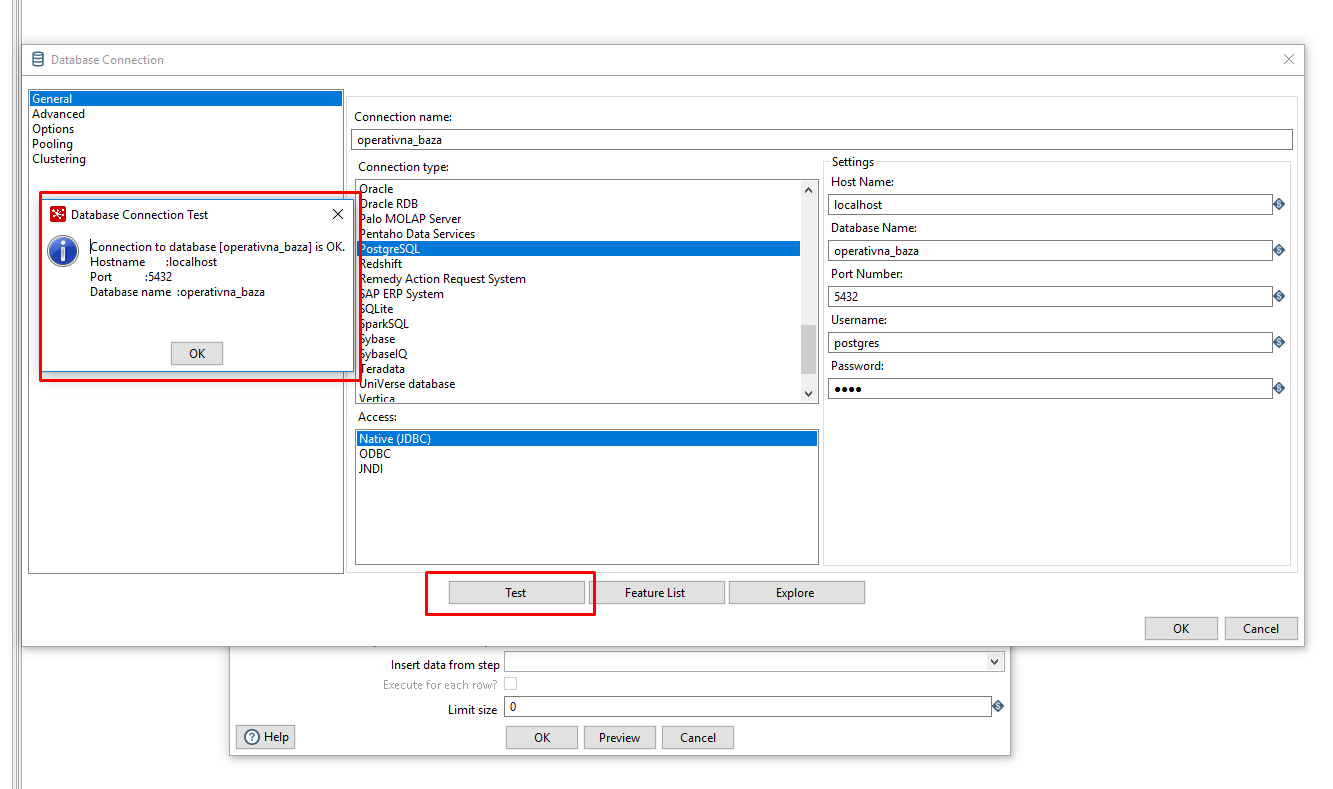
Slika - Kreiranje transformacije - table input & table output

Klikom na stavku Table input se otvara prozor prikazan na sledećoj slici. Prilikom prve transformacije u okviru jedne Transformation šeme, moramo napraviti dve nove konekcije (po jedna za operativnu bazu I skladište podataka)



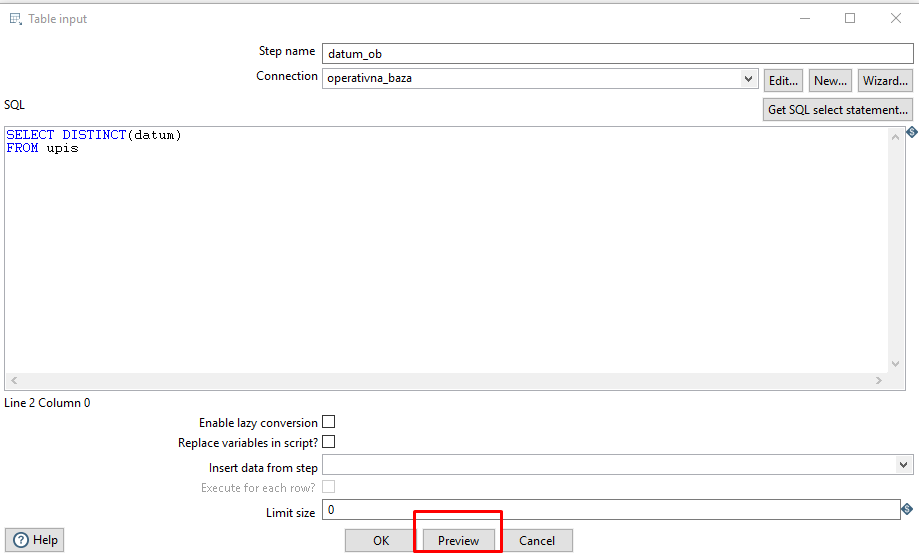
Slika - Kreiranje transformacije - kreiranje konekcije 1

Nakon unetih valdnih podataka, testiramo da li je konekcija ispravna.



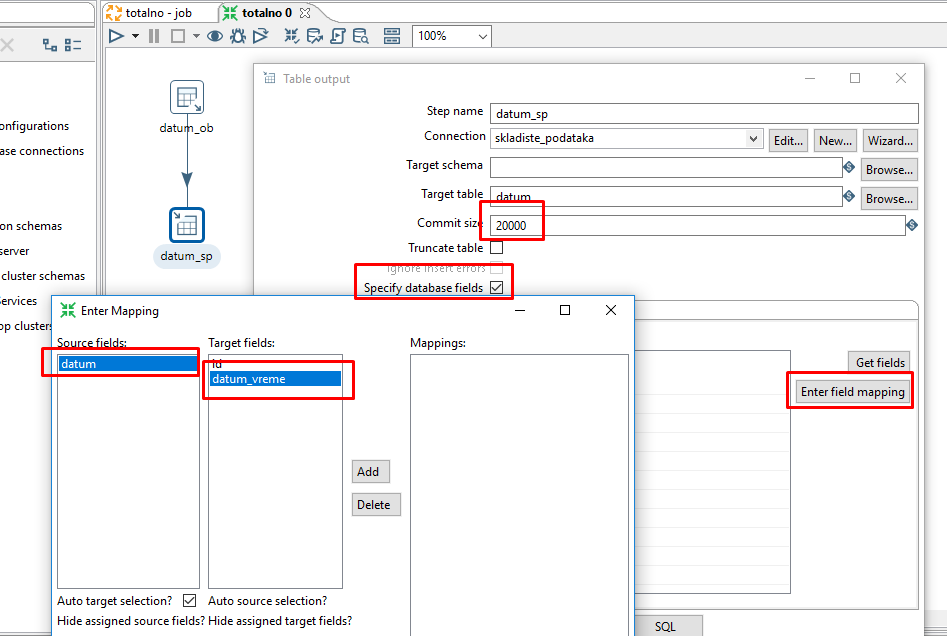
Slika - Kreiranje transformacije - kreiranje transformacije 2

Nakon kreiranja koncekcije sa operativnom bazom, pišemo SELECT upit kojim filtriramo podatke koje želimo da prosledimo u tabelu skladišta podataka. U sledećem primeru, želimo sve različite datume tokom kojih je izvršen upis. Kako bismo odmah proverili validnost upita, klikom na Preview izvršavamo upit.



Slika - Upit u table\_input elementu

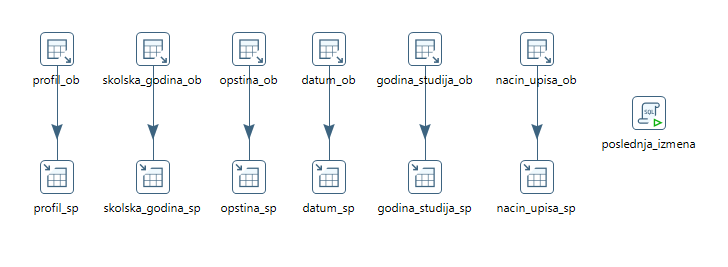
Ukoliko smo upit ispravno napisali, prelazimo na podešavanje Table output stavke. Kao i kod prethodne stavke, mora se izabrati (odnosno napraviti) konekcija prema bazi/skladištu. Nakon toga biramo koju tabelu u skladištu punimo podacima, kao i maksimalni broj redova koje prenosimo u skladište. Najvažniji deo ovog koraka jeste mapiranje kolona koje dobijamo iz Table input-a u kolone tabele skladišta podataka.



Slika - Mapiranje kolona iz table\_input-a u kolone table\_output-a

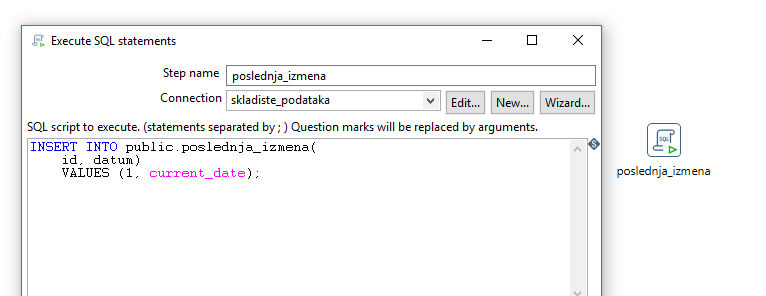
Konkretno, u gore prikazanom primeru, različite datume koje smo selektovali iz tabele upis mapiramo u kolonu datum\_vreme, dok kolona id inkrementalno dobija vrednost, što smo podesili u konfiguraciji baze.

Transformacija Import\_dimensions 1 je prikazana na sledećoj slici (mapiranje tabela iz operativne baze u tabele dimenzija skladišta podataka):



Slika - Transformacija - import dimensions 1

Primetićemo element koji se nalazi skroz desno na slici (Execute SQL statement), koji inicijalizuje datum poslednjeg ažuriranja skladišta podataka:



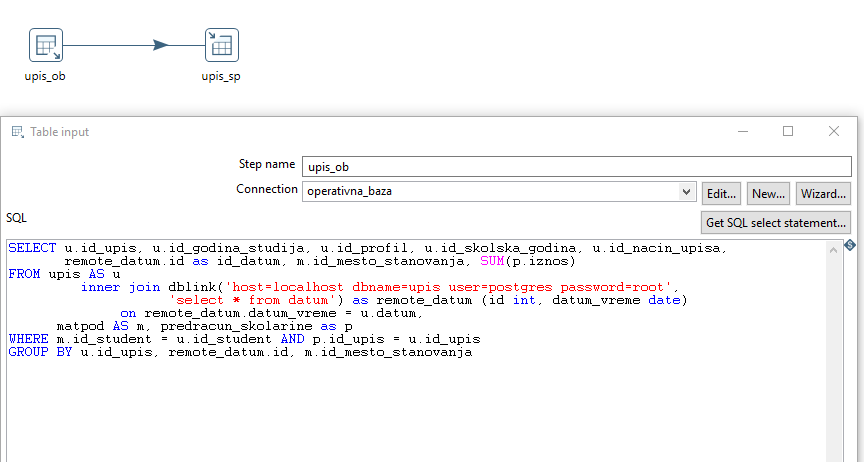
Slika - Azuriranje datuma poslednje izmene

Nakon što smo napunili podacima tabelu opstina, možemo popuniti i tabelu u kojoj postoji strani ključ koji refencira na ovu tabelu, odnosno tabelu mesto.



Slika – Transformacija import dimensions 2

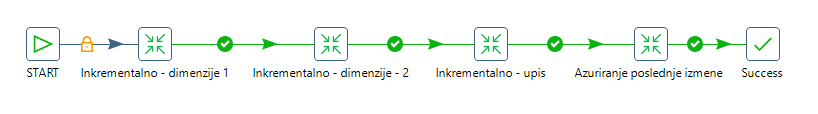
Nakon što smo popunili tabele dimenzija, možemo popuniti i tabele koje sadrže strane ključeve ka ovim tabelama, odnosno tabelu podataka, upis. Tabelu upis punimo na sličan način kao i tabele dimenzija – složenijim SQL upitom kojim filtriramo podatke od interesa u skladištu podataka, koji je prikazan na slici ispod:



Slika – Upit kojim se dobijaju potrebni podaci za tabelu upis

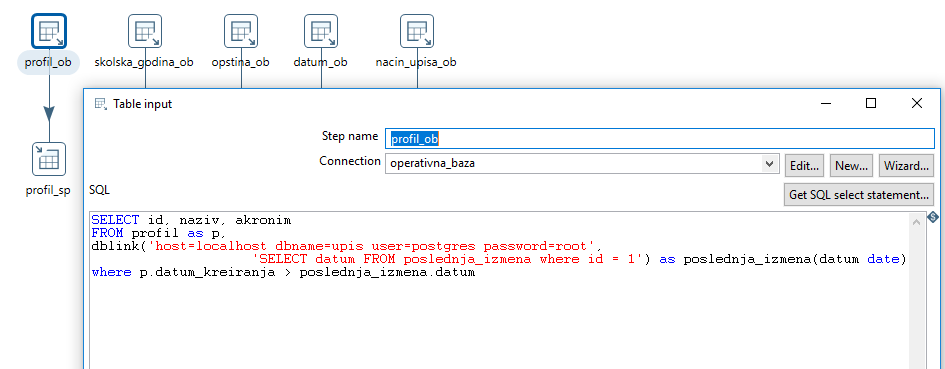
Kako popuniti skladište podacima koji se unose nakon totalnog punjenja?

Naravno, operativna baza podataka se stalno puni novim podacima (brisanje podataka ćemo zanemariti u ovom primeru). Samim tim, moramo imati način da podatke, koji su dodati u operativinu bazu nakon totalnog punjenja, dodamo i u skladište podataka. Taj proces nazivamo inkrementalnim punjenjem. Kao i kod totalnog punjenja, proces se sastoji iz inkrementalnog punjenja tabele dimenzija, a zatim inkrementalnog punjenja tabele podataka (tabele *upisa*).



Slika -Redosled izvršavanja transformacija tokom inkrementalnog punjenja

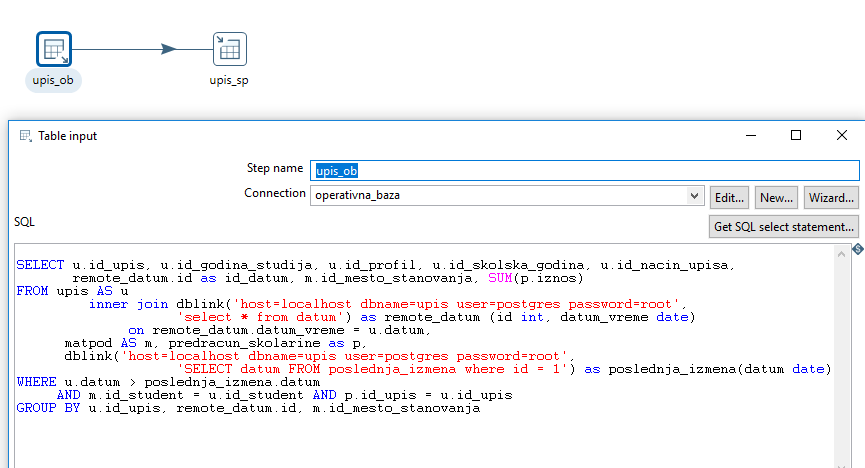
Inkremetalno punjenje dimenzija je implementirano skoro identično kao totalno punjenje (element Table input filtrira podatke koji su nam potrebni, dok u elementu Table output mapiramo dobijene podatke iz tabela operativne baze u podatke tabela skladišta podataka). Primer inkremenatlnog punjenja tabele profil vidimo na sledećoj slici:



Slika - Upit za dobijanje podataka koji su ubačeni u operativnu bazu nakon poslednjeg ažuriranja skladišta 1

SQL upitom vršimo selekciju onih redova koji su kreirani nakon poslednjeg ažuriranja skladišta podataka. (vrednost kolone datum\_kreiranja je noviji datum od datuma poslednje izmene)

Inkrementalno punjenje tabele podataka, odnosno tabele *upis*, vršimo na sličan način kao drugi primer totalnog punjenja (složeniji SQL upit):



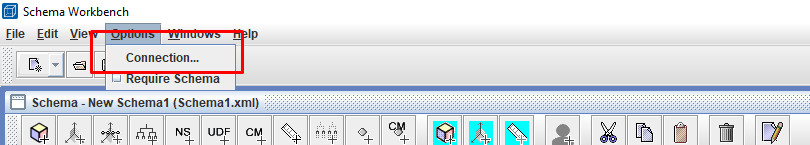
Slika - Upit za dobijanje podataka koji su ubačeni u operativnu bazu nakon poslednjeg ažuriranja skladišta - 2

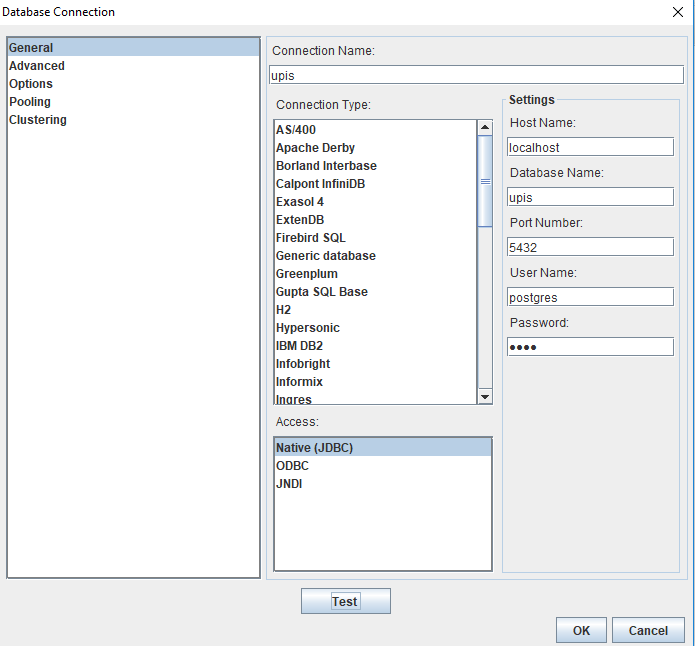
Najpre smo tabelu upis spojili sa tabelom datum (iz skladišta podataka) po koloni datum, zatim filtrirali one uplate koje su izvršene nakon poslednjeg ažuriranja skladišta podataka, i na kraju proširili filtrirane redove sa potrebnim podacima iz tabela matopod, odnosno tabele predracun\_skolarine.

Kako efikasno vršiti upite nad podacima se nalaze u skladištu podataka?

Najpre, moramo kreirati logički model pomoću alata Mondrian Schema Workbench-a i PostgreSQL servera. Da bi ta konekcija bila moguca, potrebno je ubaciti drajver, koji ćete preuzeti na sledećem [linku](https://jdbc.postgresql.org/download.html), u folder drivers Schema Workbench alata. Kreiranje logičkog modela je detaljno opisan u [primeru sa sajta](http://si4is2.etf.rs/Nastava/Vezbe/3.%20trecina/1,%202/Lekarska%20ordinacija.pdf), te kroz ovaj primer nećemo ulaziti u najsitnije detalje kreiranja logičkog modela.

Kao i kod kreiranja transformacija za punjenje, prvo moramo uspostaviti konekciju sa bazom na PostgreSQL serveru, što je prikazano na slikama ispod.



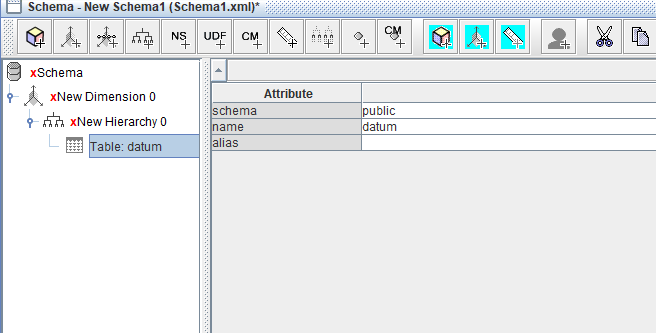


Slika - Kreiranje konekcije u Schema Workbench alatu

Postupak kreiranja šeme (logičkog modela) se sastoji od:

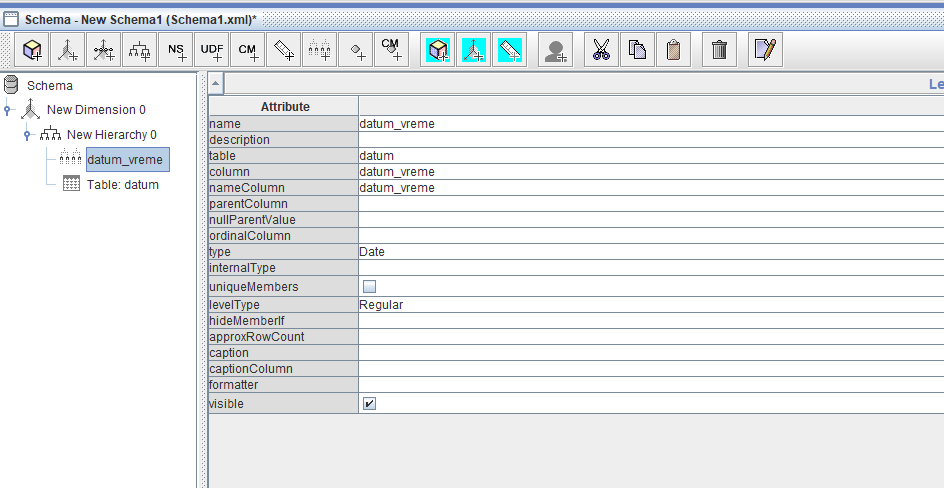
* kreiranja dimenzija (logičko predstavljanje tabela dimenzija)
* kreiranja kocki (logičko predstavljanje tabela podataka)

Svaka dimenzija mora imati referencu na tabelu skladišta podataka:



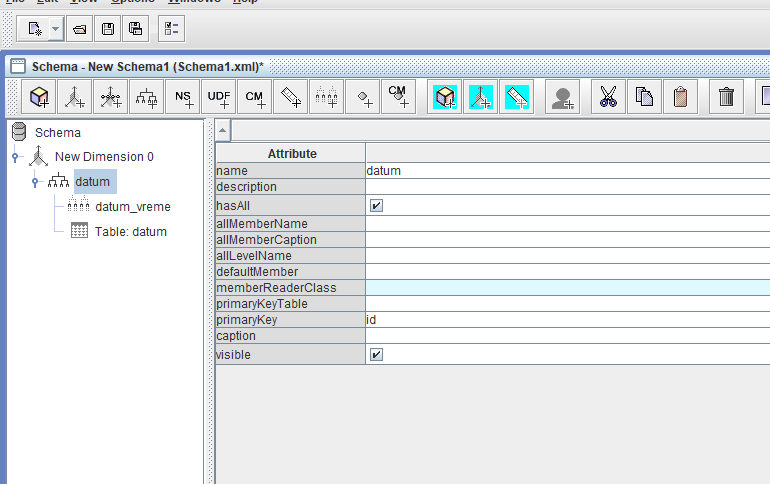
Slika - kreiranje dimenzije - tabela

Zatim, za svaku kolonu tabele dimenzije je potrebno napraviti po jedan nivo na sledeći način:



Slika - kreiranje dimenzije - nivo

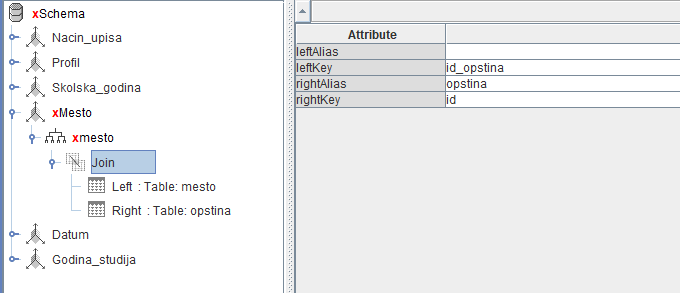
Nakon kreiranja reference na tabelu i nivoa za svaku kolonu tabele dimenzija, vraćamo se na hijerarhiju, dajemo joj ime i postavljamo primarni ključ.



Slika - kreiranje tabele - hijerarhija

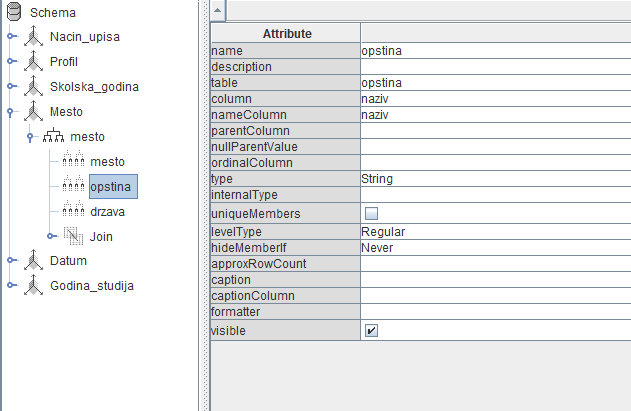
Prethodni postupak treba ponoviti za sve tabele dimenzija iz skladišta podataka : nacin\_upisa, profil, skolska\_godina i godina\_studija.

Tabela dimenzije čije se logičko predstavljanje razlikuje od prethodno opisanog jeste tabela mesto, kod koje imamo spajanje tabela mesto i opstina. Umesto dodavanja tabele u okviru hijerarhije, dodajemo spajanje (join) čija svojstva postavimo kao što je prikazano na slici ispod:



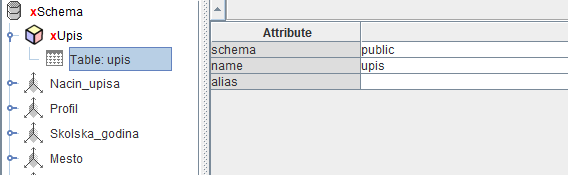
Slika - Dodavanje spajanja tabela rok i skolska\_godina

Kao i kod ostalih dimenzija, i kod dimenzije mesto, dodajemo nivoe i to za kolonu naziv (nivou smo dali ime mesto) iz tabele mesto , kolonu naziv iz tabele opstina (nivou smo dali ime opstina) i kolonu drzava iz tabele opstina:



Slika - Dodavanje nivoa u dimenziju rok

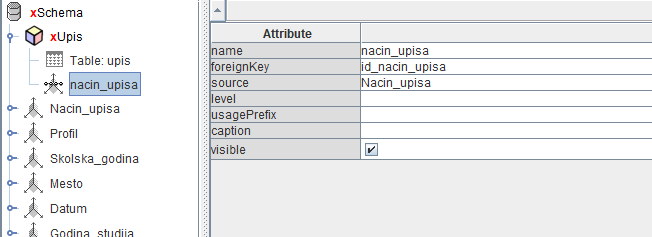
Nakon kreiranja dimenzija, prelazimo na kreiranje kocki, odnosno logičku predstavu tabele podataka. Kao i dimenzije, i kocka mora imati referencu na fizičku tabelu iz skladišta podataka.



Slika - kreiranje kocke

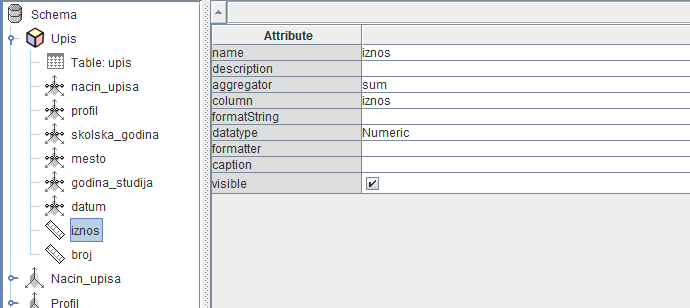
Pored reference na fizičku tabelu, kocka mora sadržati još dva elementa:

* dimension\_usage, odnsono reference na već kreirane dimenzije – pored dimenzije potrebno je izabrati koji strani ključ ukazuje na selektovanu dimenziju



Slika - kreiranje kocke - upotreba dimenzije

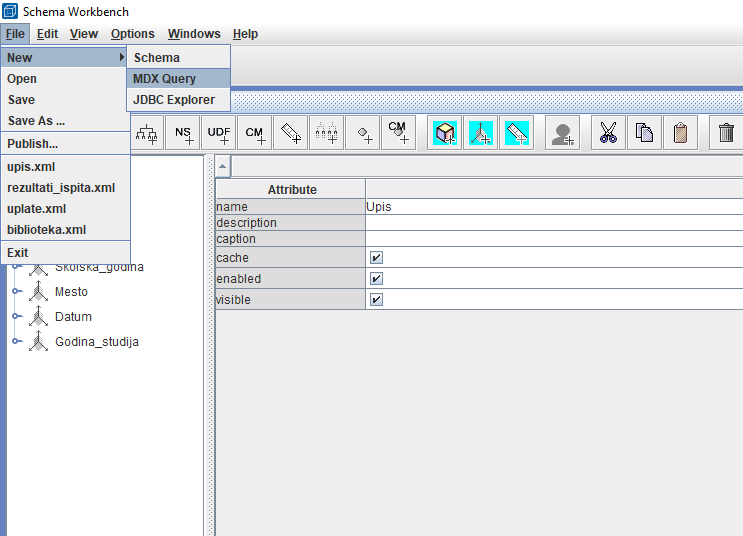
* Measures, odnosno mere – podatke koje želimo da dobijemo kao rezultat analize



Slika - kreiranje kocke - kreiranje mere

MDX upiti, primer i performanse

Nakon kreiranog logičkog modela, možemo vršiti višedimenzionalne upite nad skladištem podataka. Prozor za pisanje I izvršavanje upita otvaramo preko File > New > MDX Query.



Slika - Otvaranje prozora za pisanje MDX upita

MDX je detaljno opisan u [primeru sa sajta](http://si4is2.etf.rs/Nastava/Vezbe/3.%20trecina/1,%202/Lekarska%20ordinacija.pdf), tako da ćemo kroz narednih par primera (MDX i SQL) upita uporediti peformanse izvršavanja ovih upita.

NAPOMENA: U svakom od sledećih SQL upitima ćemo u WHERE uslovu imati *EXISTS (select from predracun\_skolarine where u.id\_upis = id\_upis)*, gde filtriramo samo one upise koji imaju predračun školirane u operativnoj bazi, kako bismo dobijali identične rezultate ovih upita.

* Broj upisanih studenata svake skolske godine po godinama
  + SQL

*SELECT sk.naziv, gs.naziv, count(\*)*

*FROM upis AS u, skolska\_godina AS sk, godina\_studija AS gs*

*WHERE u.id\_godina\_studija = gs.id*

*AND u.id\_skolska\_godina = sk.id*

*AND EXISTS (SELECT FROM predracun\_skolarine WHERE u.id\_upis = id\_upis)*

*GROUP BY sk.id, gs.id*

**

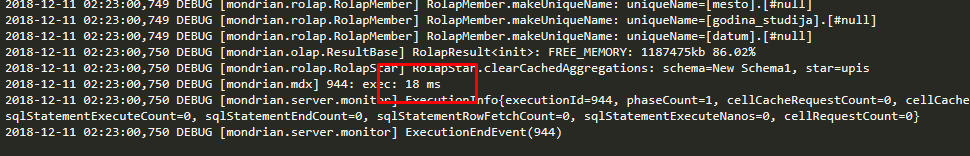
Slika - Broj upisanih studenata svake skolske godine po godinama SQL

* + *MDX*

*SELECT {[Measures].[count]} ON COLUMNS,*

*NON EMPTY(CROSSJOIN ({[Skolska\_godina].Children}, [Godina\_studija].Children)) ON ROWS*

*FROM Upis*

**

Slika - Broj upisanih studenata svake skolske godine po godinama MDX

* Broj ispisnanih studenata po skolskoj godini
  + SQL

*SELECT sk.naziv, count(\*)*

*FROM upis AS u, skolska\_godina AS sk, nacin\_upisa AS ns*

*WHERE u.id\_nacin\_upisa = ns.id*

*AND u.id\_skolska\_godina = sk.id*

*AND EXISTS (SELECT FROM predracun\_skolarine WHERE u.id\_upis = id\_upis)*

*AND ns.naziv = 'испис'*

*GROUP BY u.id\_upis, sk.naziv*

**

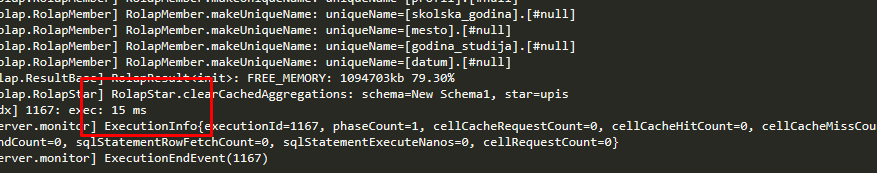
Slika - Broj ispisnanih studenata po skolskoj godini SQL

* + *MDX*

*SELECT {[Measures].[count]} ON COLUMNS,*

*NON EMPTY(CROSSJOIN ({[Skolska\_godina].Children}, [Nacin\_upisa].[испис])) ON ROWS*

*FROM Upis*

**

Slika - Broj ispisnanih studenata po skolskoj godini MDX

* Broj upisanih studenata po opštinama sortirano od najvećeg broja ka najmanjem
  + SQL

*SELECT op.naziv, count(\*) AS broj*

*FROM upis AS u, opstina AS op, matpod AS mat*

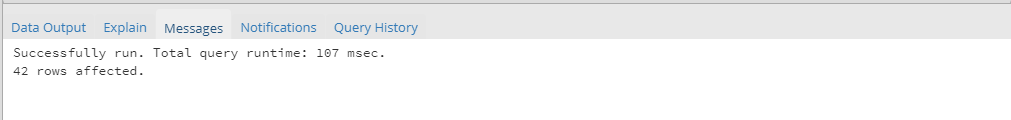
*where u.id\_student = mat.id\_student*

*AND mat.id\_opstina\_stanovanja = op.id*

*AND EXISTS (SELECT FROM predracun\_skolarine WHERE u.id\_upis = id\_upis)*

*GROUP BY op.naziv*

*ORDER BY broj DESC*

**

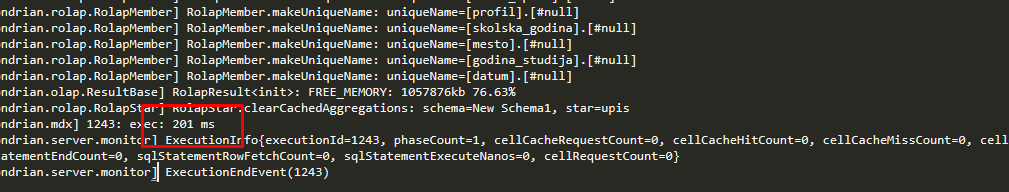
Slika - Broj upisanih studenata po opštinama sortirano od najvećeg broja ka najmanjem SQL

* + *MDX*

*SELECT {[Measures].[count]} ON COLUMNS,*

*ORDER({[Opstina].Members}, [Measures].[count], DESC) ON ROWS*

*FROM Upis*

**

Slika - Broj upisanih studenata po opštinama sortirano od najvećeg broja ka najmanjem MDX

* Broj upisanih studenata po opštinama - nesortirano
  + SQL

*SELECT op.naziv, count(\*) AS broj*

*FROM upis AS u, opstina AS op, matpod AS mat*

*where u.id\_student = mat.id\_student*

*AND mat.id\_opstina\_stanovanja = op.id*

*AND EXISTS (SELECT FROM predracun\_skolarine WHERE u.id\_upis = id\_upis)*

*GROUP BY op.naziv*

**

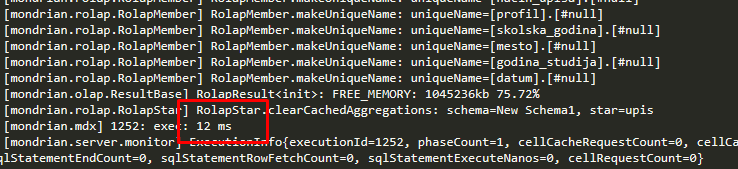
Slika - Broj upisanih studenata po opštinama - nesortirano SQL

* + *MDX*

*SELECT {[Measures].[count]} ON COLUMNS,*

*{[Opstina].Members} ON ROWS*

*FROM Upis*

**

Slika - Broj upisanih studenata po opštinama - nesortirano MDX

Iz poslednja dva primera, možemo zaključiti da je ORDER funkcija značajno skuplja prilikom izvršavanja MDX upita u odnosu na izvršavanje SQL upita.