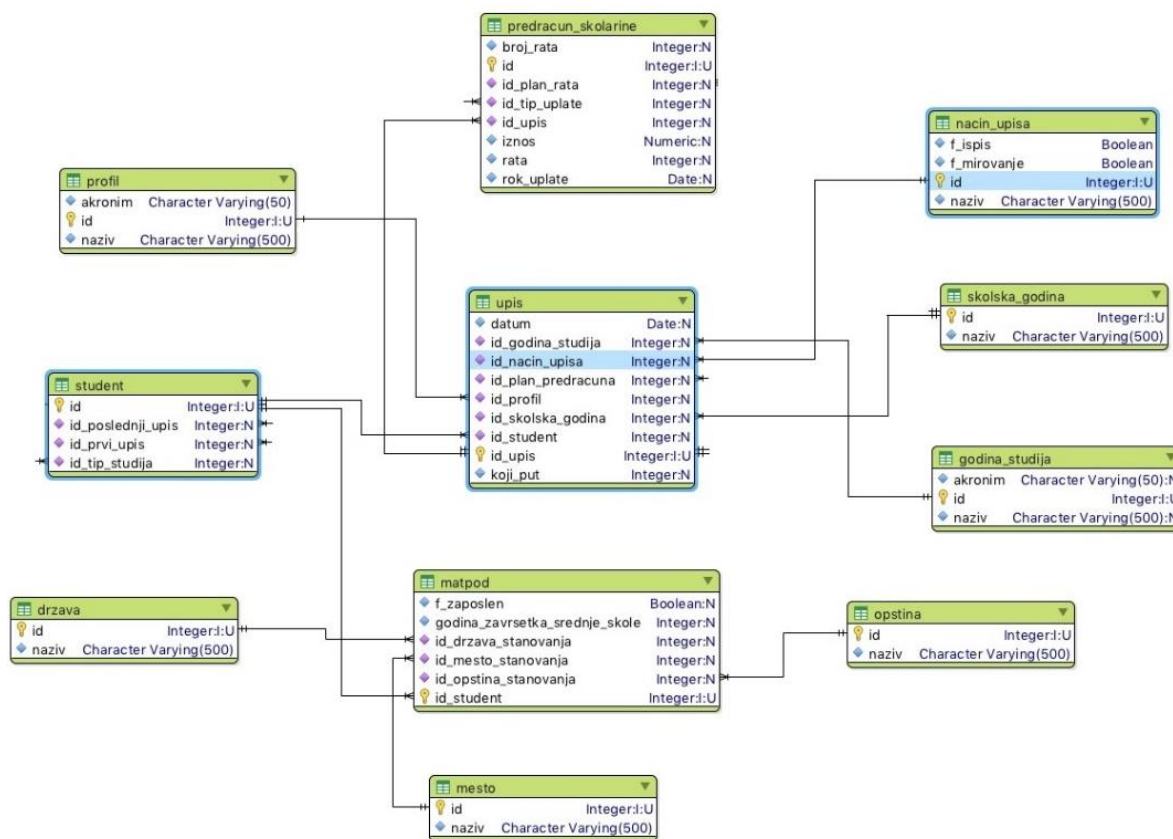


Kako ispravno dizajnirati skladište podataka?

Proces dizajniranja se svodi na sledeća četiri koraka:

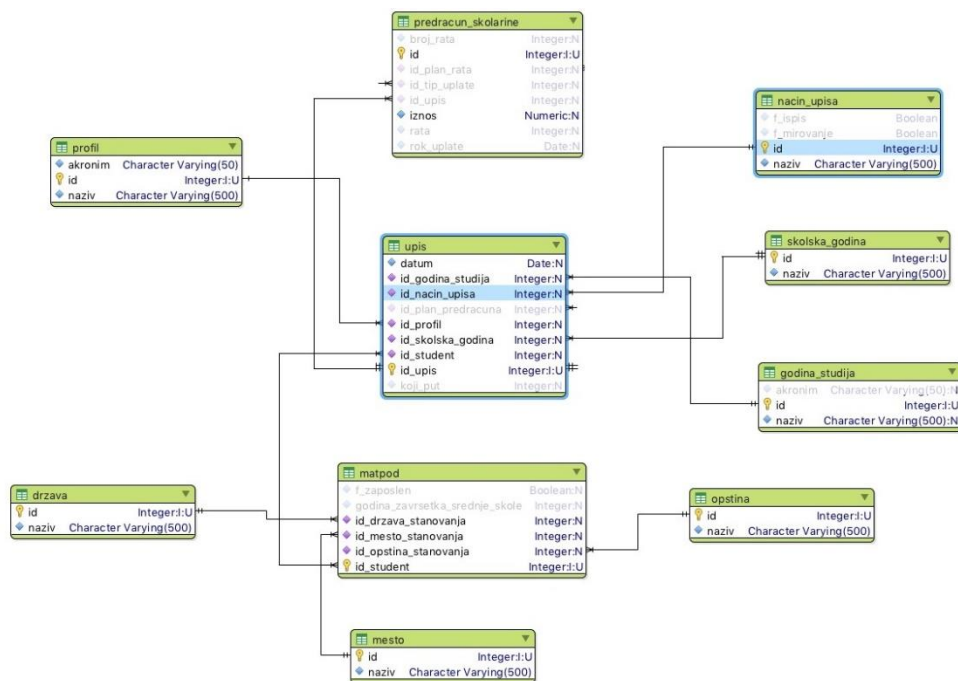
1. Izaberite poslovni proces koji se analizira (npr. prodaja, proizvodnja, utrošak i sl.) kao i glavni cilj tj. šta treba da dobijete kao rezultat.
2. Izaberite srž (centar) procesa koji pratite tj. najveći nivo usitnjavanja.
3. Izaberite dimenzije koje ćete pratiti (vremenski period, lokacija, proizvodi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



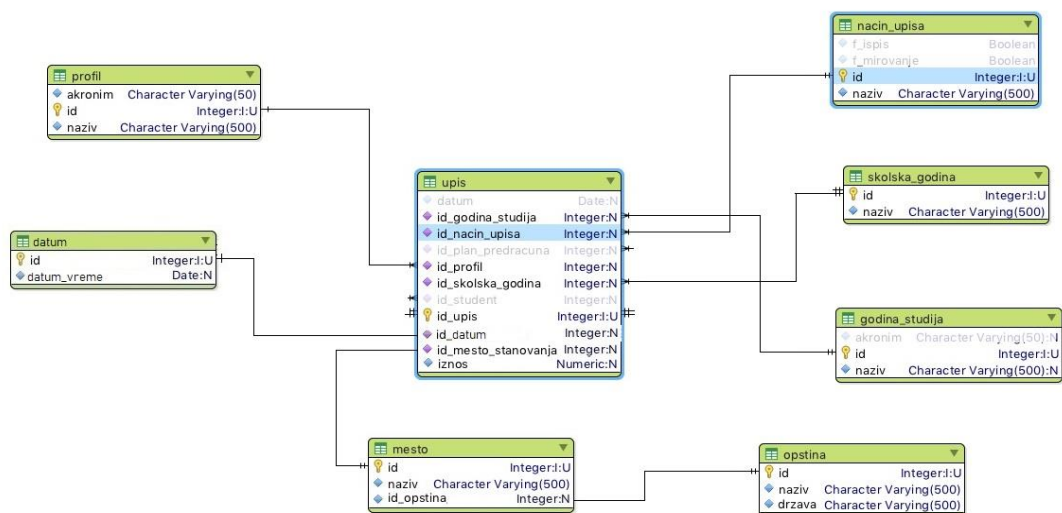
SLIKA 1 -- ŠEMA DELA OPERATIVNE BAZE KOJI SE ODNOSI NA UPIS

Primitićemo 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 upisa. (slika 2)



SLIKA 2 –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 3 - MODELOVANJE SKLADIŠTA PODATAKA 2

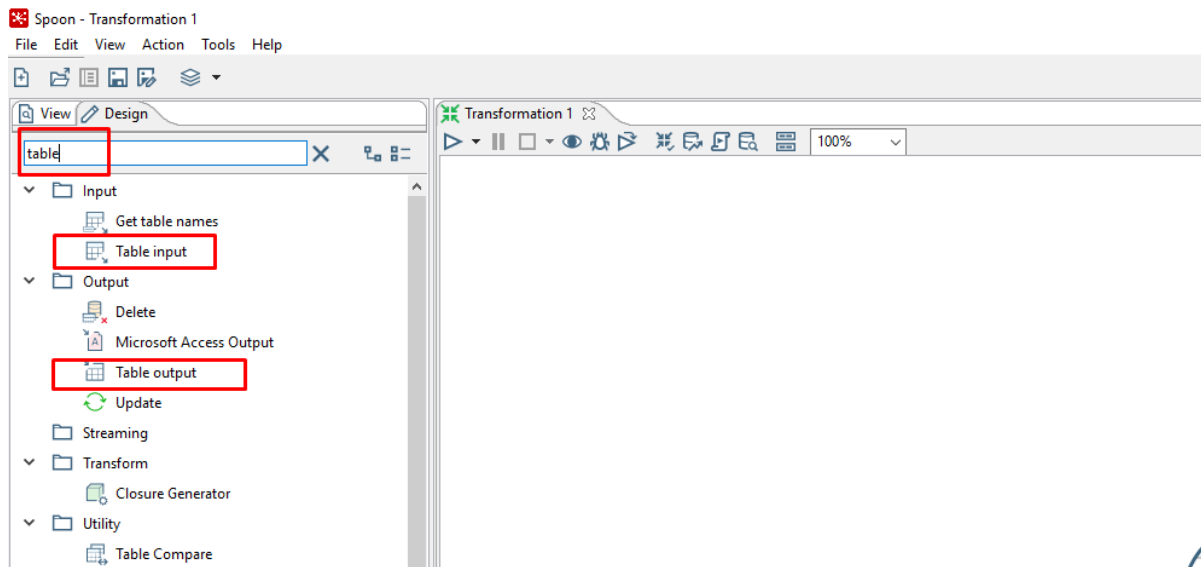
Na slici iznad imamo šemu skladišta podataka pomoću koje lako možemo vršiti analizu upisa po sledećim dimenzijama:

- 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

Kako kreirati transformaciju?

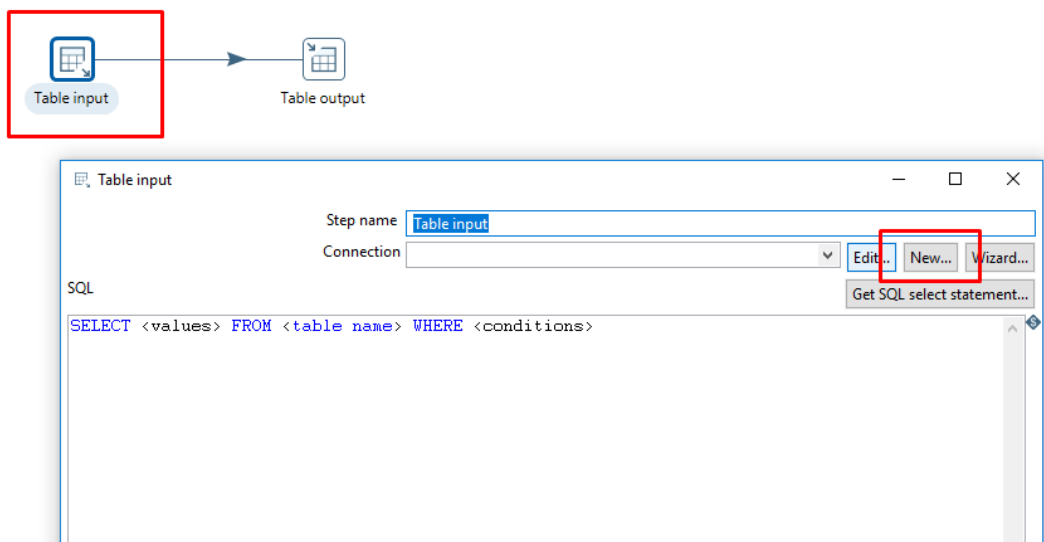
Za kreiranje transformacija koristimo alat *Spoon* koji možete preuzeti na [linku](#).

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).



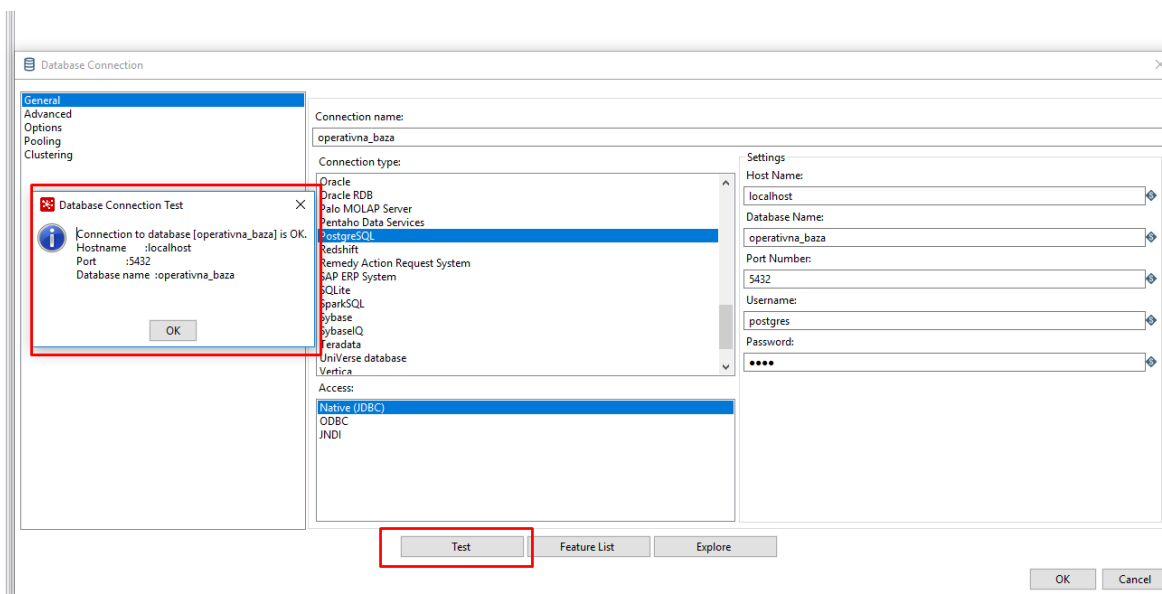
SLIKA 6 - KREIRANJE TRANSFORMACIJE - TABLE INPUT & TABLE OUTPUT

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)



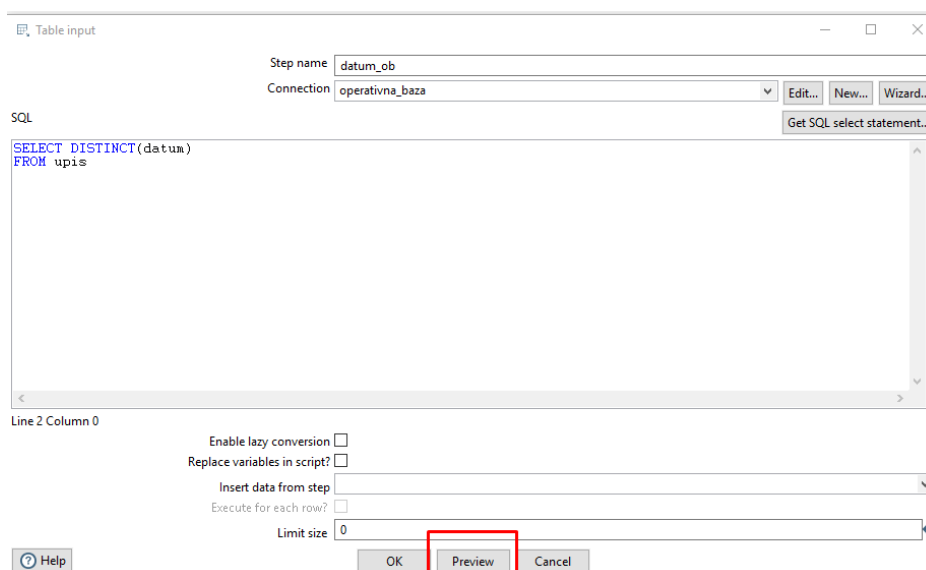
SLIKA 7 - KREIRANJE TRANSFORMACIJE - KREIRANJE KONEKCIJE 1

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



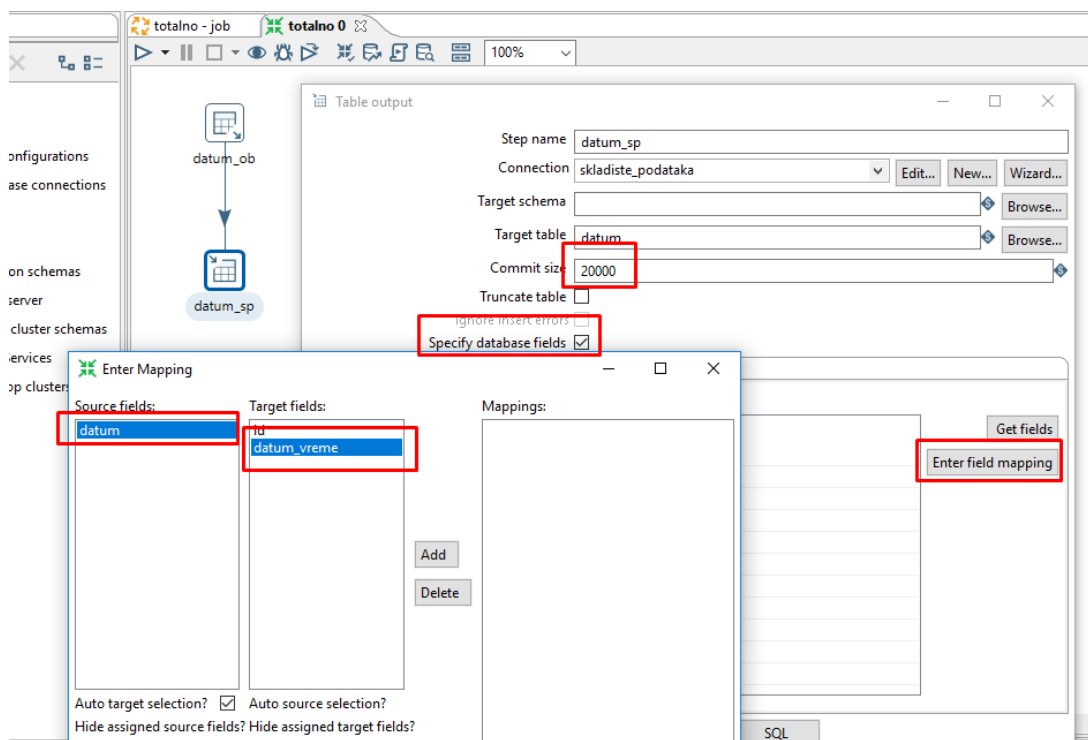
SLIKA 8 - KREIRANJE TRANSFORMACIJE - KREIRANJE TRANSFORMACIJE 2

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 9 - UPIT U TABLE_INPUT ELEMENTU

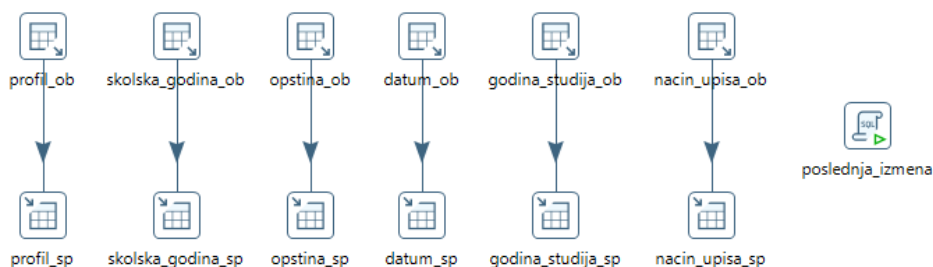
Ukoliko smo upit ispravno napisali, prelazimo na podešavanje *Table output* stavke. Kao i kod prethodne stavke, mora se izabrati (odnosno napraviti) konekcija prema bazi/skladištu. Nakon toga biramo koju tabelu u skladištu punimo podacima, kao i maksimalni broj redova koje prenosimo u skladište. Najvažniji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



SLIKA 10 - MAPIRANJE KOLONA IZ TABLE_INPUT-A U KOLONE TABLE_OUTPUT-A

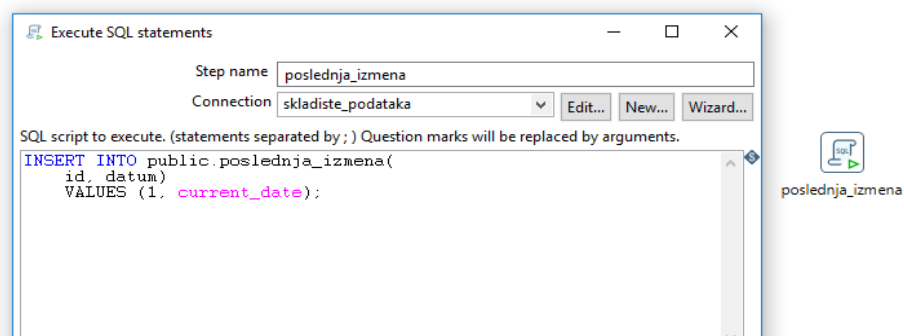
Konkretno, u gore prikazanom primeru, različite datume koje smo selektovali iz tabele *upis* mapiramo u kolonu *datum_vreme*, dok kolona *id* inkrementalno dobija vrednost, što smo podesili u konfiguraciji baze.

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



SLIKA 11 - TRANSFORMACIJA - IMPORT DIMENSIONS 1

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



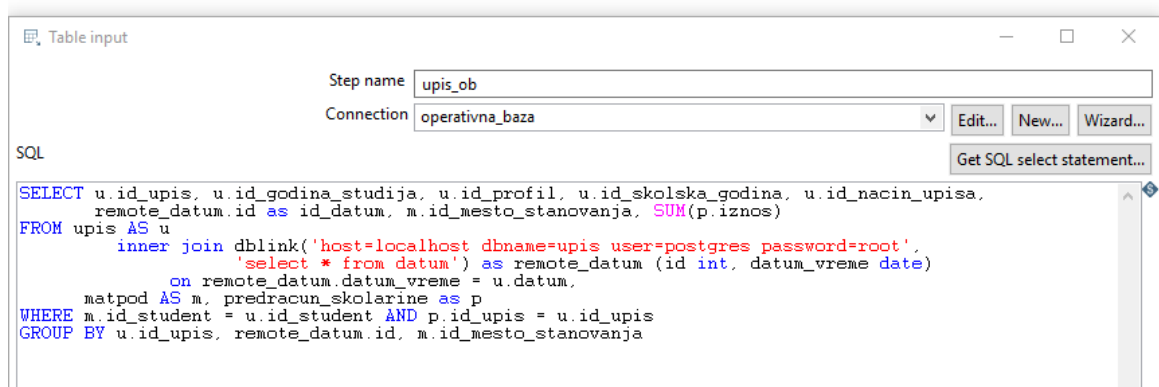
SLIKA 12 - AZURIRANJE DATUMA POSLEDNJE IZMENE

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 13 – TRANSFORMACIJA IMPORT DIMENSIONS 2

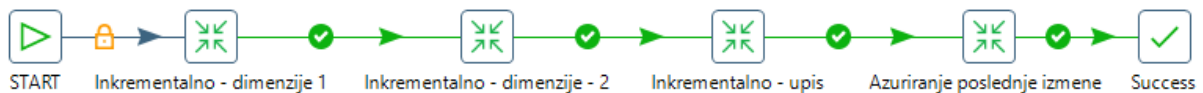
Nakon što smo popunili tabele dimenzija, možemo popuniti i tabele koje sadrže strane ključeve ka ovim tabelama, odnosno tabelu podataka, *upis*. Tabelu *upis* punimo na sličan način kao i tabele dimenzija – složenijim SQL upitom kojim filtriramo podatke od interesa u skladištu podataka, koji je prikazan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



SLIKA 14 – UPIT KOJIM SE DOBIJAJU POTREBNI PODACI ZA TABELU UPIS

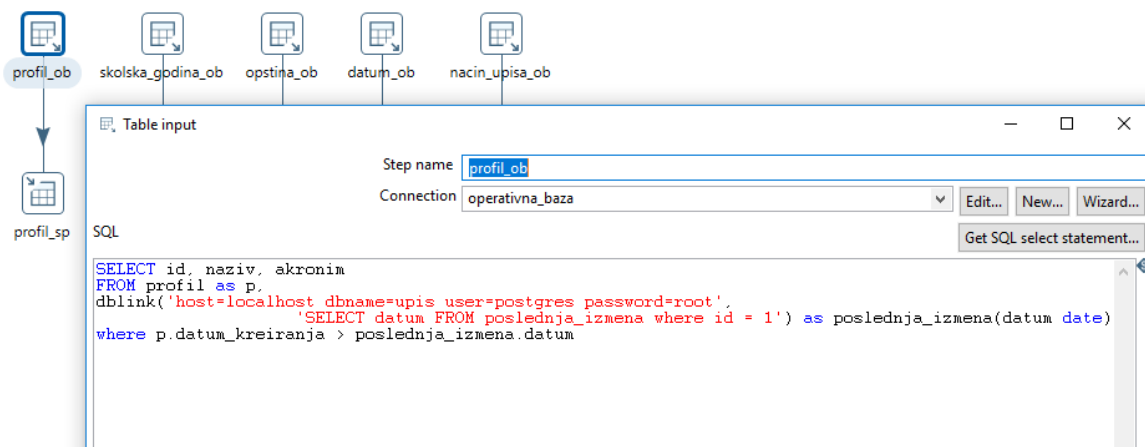
Kako popuniti skladište podacima koji se unose nakon totalnog punjenja?

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 operativnu 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 15 -REDSLED IZVRŠAVANJA TRANSFORMACIJA TOKOM INKREMENTALNOG PUNJENJA

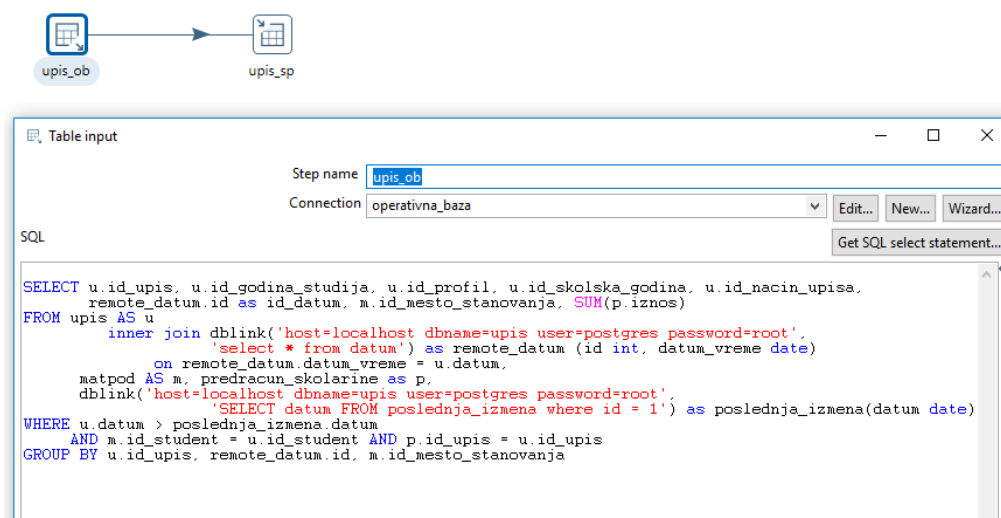
Inkrementalno punjenje dimenzija je implementirano skoro identično kao totalno punjenje (element *Table input* filtrira podatke koji su nam potrebni, dok u elementu *Table output* mapiramo dobijene podatke iz tabela operativne baze u podatke tabela skladišta podataka). Primer inkremenatlnog punjenja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



SLIKA 16 - 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):



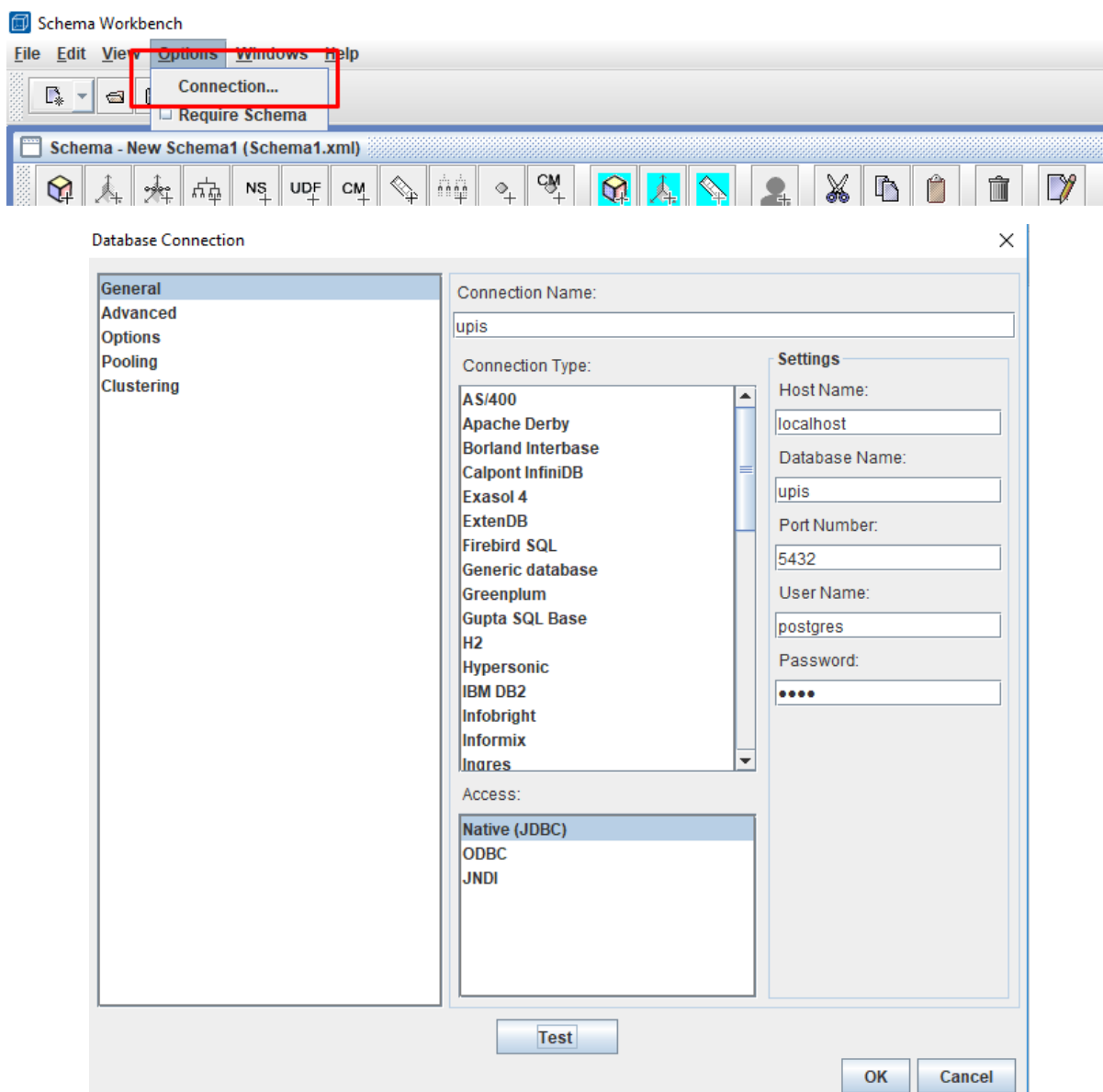
SLIKA 17 - UPIT ZA DOBIJANJE PODATAKA KOJI SU UBAČENI U OPERATIVNU BAZU NAKON POSLEDNJEG AŽURIRANJA SKLADIŠTA - 2

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 moguća, potrebno je ubaciti drajver, koji ćete preuzeti na sledećem [linku](#), u folder *drivers* Schema Workbench alata. Kreiranje logičkog modela je detaljno opisan u [primeru sa sajta](#), 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.

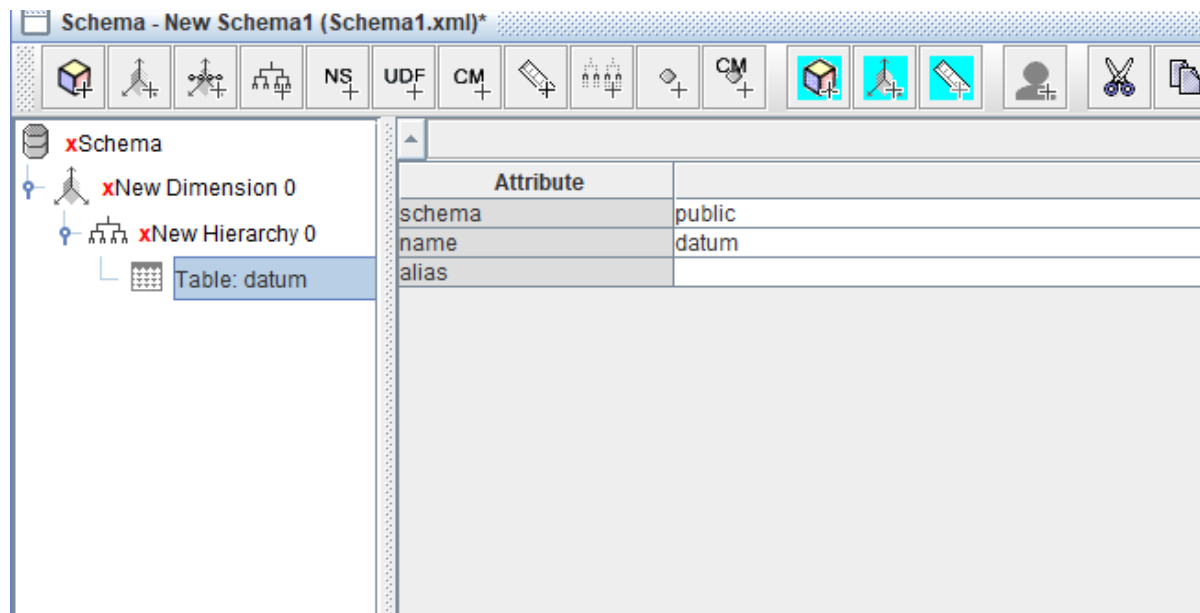


SLIKA 18 - KREIRANJE KONEKCIJE U SCHEMA WORKBENCH ALATU

Postupak kreiranja šeme (logičkog modela) se sastoji od:

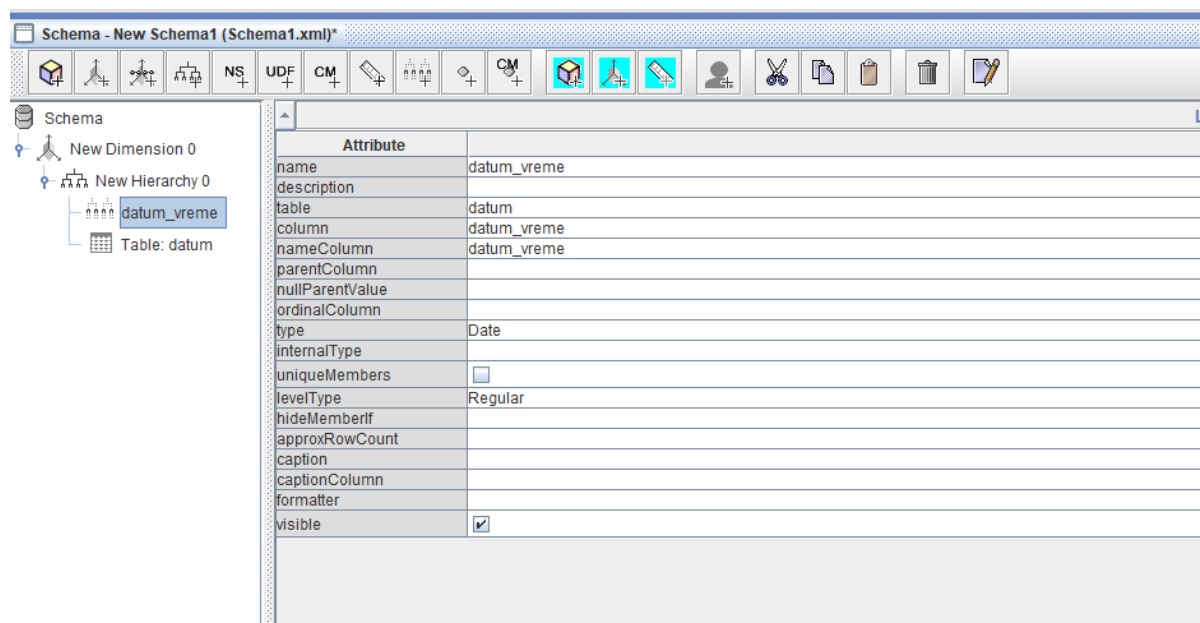
- kreiranja *dimenzija* (logičko predstavljanje tabela dimenzija)
- kreiranja *kocki* (logičko predstavljanje tabela podataka)

Svaka dimenzija mora imati referencu na tabelu skladišta podataka:



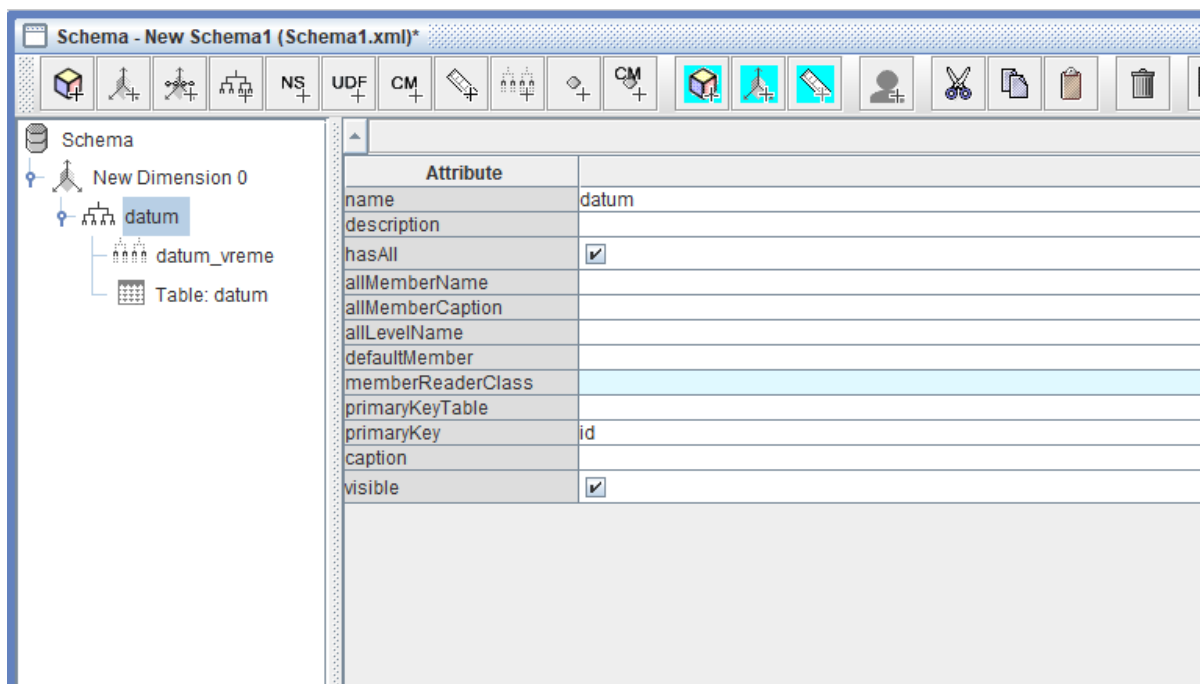
SLIKA 19 - KREIRANJE DIMENZIJE - TABELA

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



SLIKA 20 - KREIRANJE DIMENZIJE - NIVO

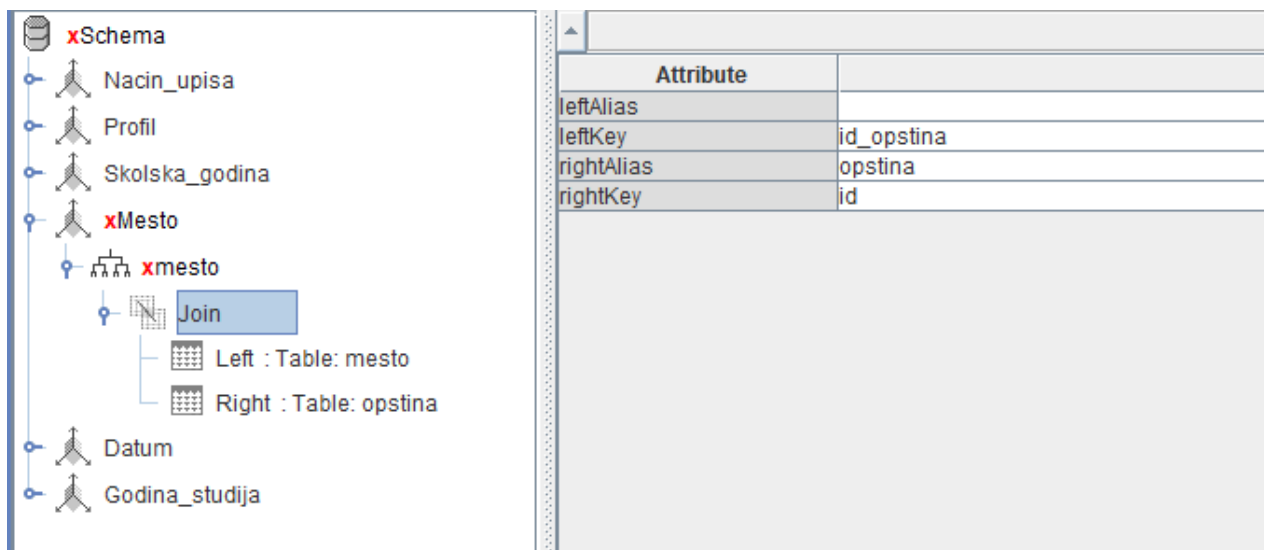
Nakon kreiranja reference na tabelu i nivoa za svaku kolonu tabele dimenzija, vraćamo se na hijerarhiju, dajemo joj ime i postavljamo primarni ključ.



SLIKA 21 - KREIRANJE TABELE - HIJERARHIJA

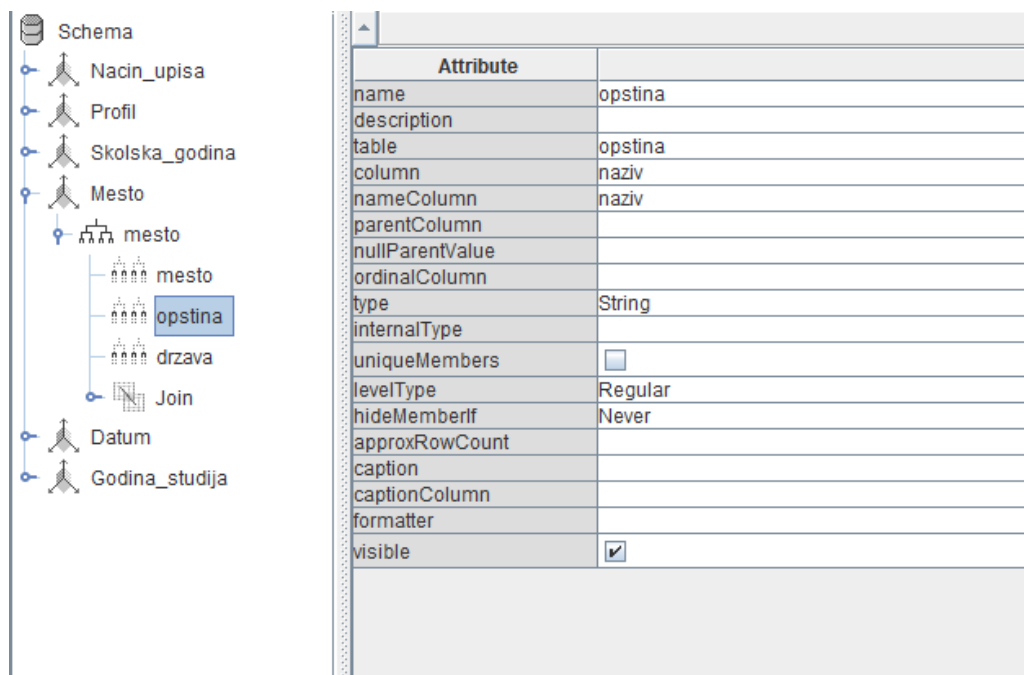
Prethodni postupak treba ponoviti za sve tabele dimenzija iz skladišta podataka : *nacin_upisa*, *profil*, *skolska_godina* i *godina_studija*.

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:



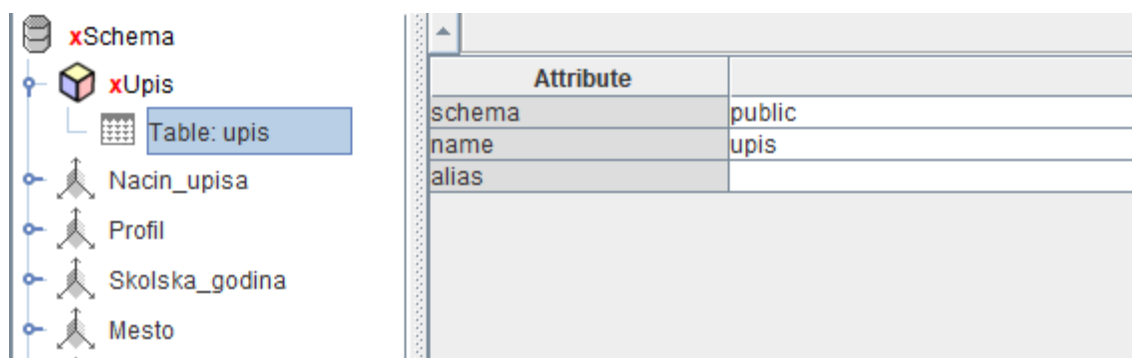
SLIKA 22 - DODAVANJE SPAJANJA TABELA ROK I SKOLSKA_GODINA

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 23 - DODAVANJE NIVOA U DIMENZIJU ROK

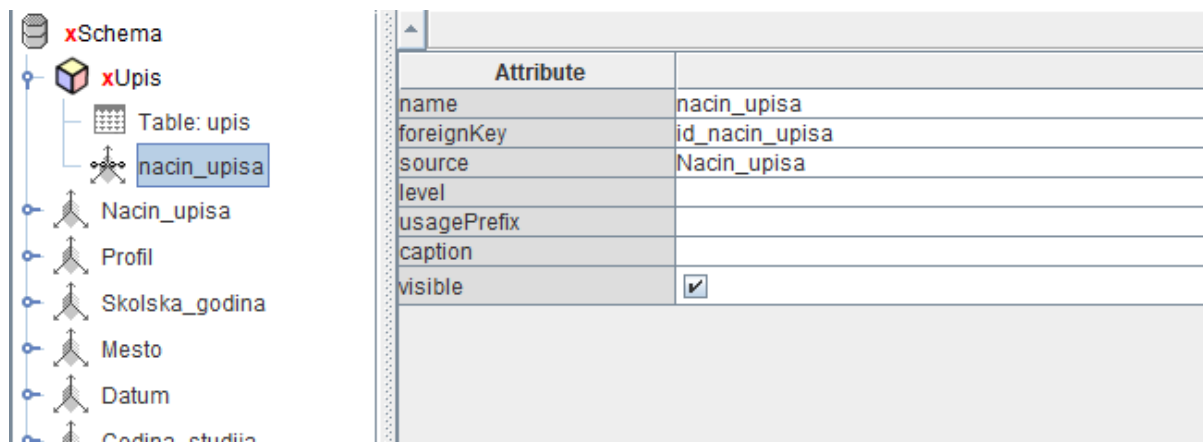
Nakon kreiranja dimenzija, prelazimo na kreiranje *kocki*, odnosno logičku predstavu tabele podataka. Kao i *dimenzije*, i *kocka* mora imati referencu na fizičku tabelu iz skladišta podataka.



SLIKA 24 - KREIRANJE KOCKE

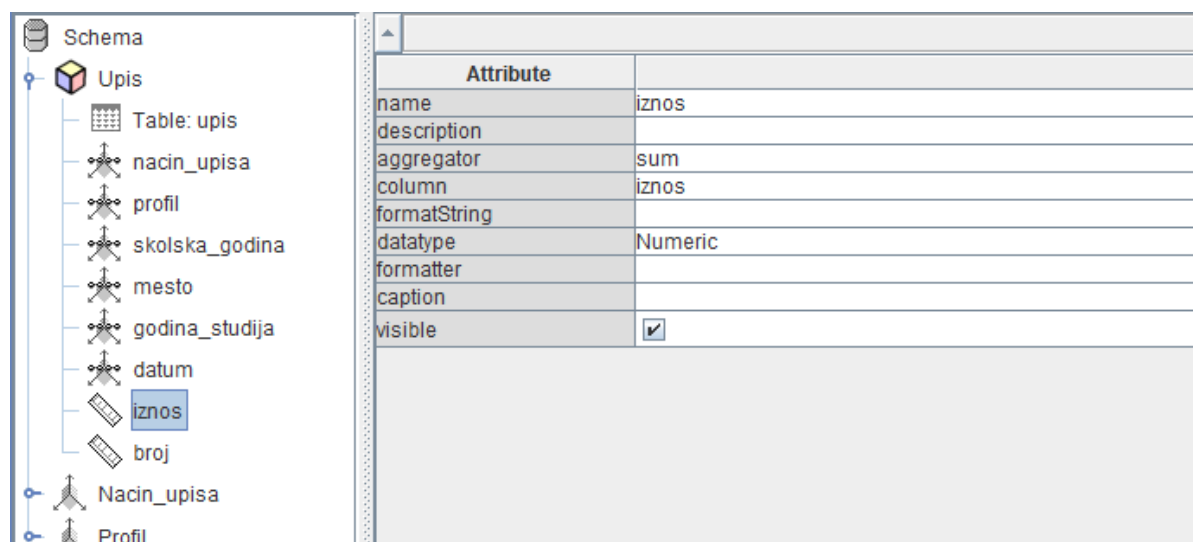
Pored reference na fizičku tabelu, *kocka* mora sadržati još dva elementa:

- *dimension_usage*, odnosno reference na već kreirane dimenzije – pored dimenzije potrebno je izabrati koji strani ključ ukazuje na selektovanu dimenziju



SLIKA 25 - KREIRANJE KOCKE - UPOTREBA DIMENZIJE

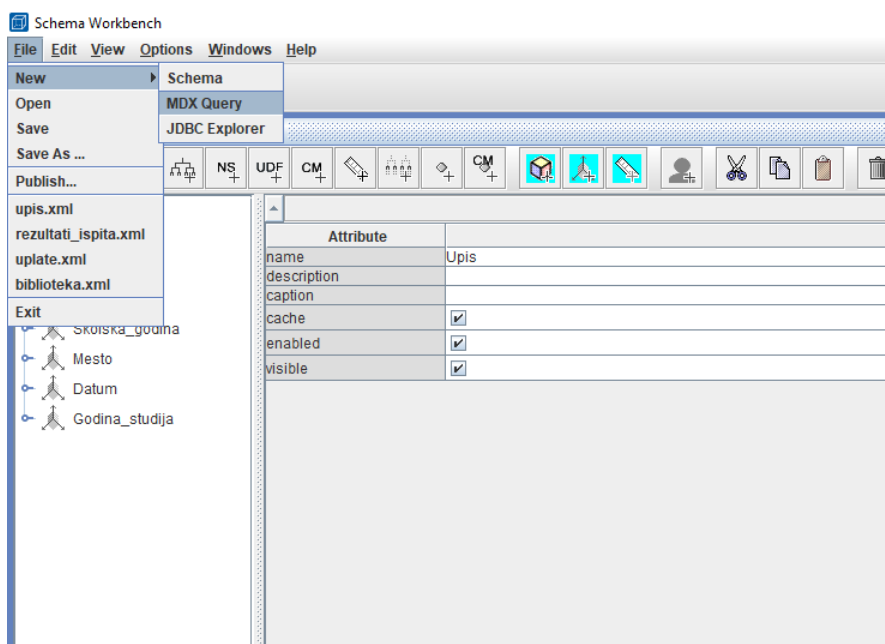
- *Measures*, odnosno mere – podatke koje želimo da dobijemo kao rezultat analize



SLIKA 26 - 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 27 - OTVARANJE PROZORA ZA PISANJE MDX UPITA

MDX je detaljno opisan u [primeru sa sajta](#), tako da ćemo kroz narednih par primera (MDX i SQL) upita uporediti performanse 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 predracun 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 28 - 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
```

```
2018-12-11 02:23:00,749 DEBUG [mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[mesto].[#null]
2018-12-11 02:23:00,749 DEBUG [mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[godina_studija].[#null]
2018-12-11 02:23:00,749 DEBUG [mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[datum].[#null]
2018-12-11 02:23:00,750 DEBUG [mondrian.olap.ResultBase] RolapResult<init>: FREE_MEMORY: 1187475kb 86.02%
2018-12-11 02:23:00,750 DEBUG [mondrian.rolap.RolapStar] RolapStar.clearCachedAggregations: schema=New Schema1, star=upis
2018-12-11 02:23:00,750 DEBUG [mondrian.mdx] 944: exec: 18 ms
2018-12-11 02:23:00,750 DEBUG [mondrian.server.monitor] ExecutionInfo{executionId=944, phaseCount=1, cellCacheRequestCount=0, cellCacheHitCount=0, sqlStatementExecuteCount=0, sqlStatementEndCount=0, sqlStatementRowFetchCount=0, sqlStatementExecuteNanos=0, cellRequestCount=0}
2018-12-11 02:23:00,750 DEBUG [mondrian.server.monitor] ExecutionEndEvent(944)
```

SLIKA 29 - BROJ UPISANIH STUDENATA SVAKE SKOLSKJE GODINE PO GODINAMA MDX

- Broj ispisanih 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 = 'ucnuc'
GROUP BY u.id_upis, sk.naziv
```

Data Output Explain Messages Notifications Query History

Successfully run. Total query runtime: 90 msec.
1 rows affected.

SLIKA 30 - BROJ ISPISANIH STUDENATA PO SKOLSKOJ GODINI SQL

- MDX

```
SELECT {[Measures].[count]} ON COLUMNS,
NON EMPTY(CROSSJOIN ({[Skolska_godina].Children}, [Nacin_upisa].[ucnuc])) ON ROWS
FROM Upis
```

```
lap.RolapMember] RolapMember.makeUniqueName: uniqueName=[skolska_godina].[#null]
lap.RolapMember] RolapMember.makeUniqueName: uniqueName=[mesto].[#null]
lap.RolapMember] RolapMember.makeUniqueName: uniqueName=[godina_studija].[#null]
lap.RolapMember] RolapMember.makeUniqueName: uniqueName=[datum].[#null]
lap.ResultBase] RolapResult<init>: FREE_MEMORY: 1094703kb 79.30%
lap.RolapStar] RolapStar.clearCachedAggregations: schema=New Schema1, star=upis
dx] 1167: exec: 15 ms
erver.monitor] ExecutionInfo{executionId=1167, phaseCount=1, cellCacheRequestCount=0, cellCacheHitCount=0, cellCacheMissCou
ndCount=0, sqlStatementRowFetchCount=0, sqlStatementExecuteNanos=0, cellRequestCount=0}
erver.monitor] ExecutionEndEvent(1167)
```

SLIKA 31 - BROJ ISPISANIH STUDENATA PO SKOLSKOJ GODINI MDX

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

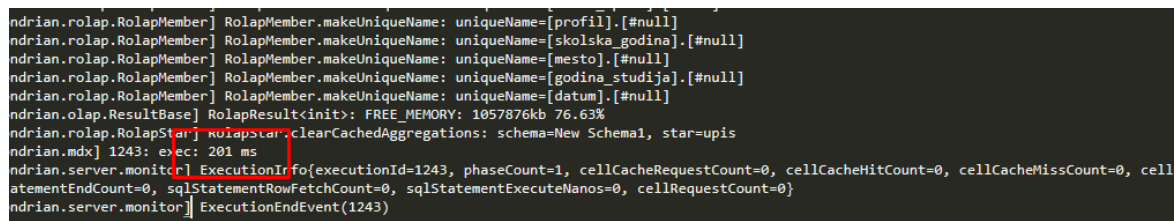
```
SELECT op.naziv, count(*) AS broj
FROM upis AS u, opstina AS op, matpod AS mat
where u.id_student = mat.id_student
      AND mat.id_opstina_stanovanja = op.id
      AND EXISTS (SELECT FROM predracun_skolarine WHERE u.id_upis = id_upis)
GROUP BY op.naziv
ORDER BY broj DESC
```



SLIKA 32 - 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 33 - 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 34 - BROJ UPISANIH STUDENATA PO OPŠTINAMA - NESORTIRANO SQL

- MDX

```
SELECT {[Measures].[count]} ON COLUMNS,
      {[Opstina].Members} ON ROWS
FROM Upis
```

```
[mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[profil].[#null]
[mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[skolska_godina].[#null]
[mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[mesto].[#null]
[mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[godina_studija].[#null]
[mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[datum].[#null]
[mondrian.olap.ResultBase] RolapResult<init>: FREE_MEMORY: 1045236kb 75.72%
[mondrian.rolap.RolapStar] RolapStar.clearCachedAggregations: schema=New Schema1, star=upis
[mondrian.mdx] 1252: exec: 12 ms
[mondrian.server.monitor] ExecutionInfo{executionId=1252, phaseCount=1, cellCacheRequestCount=0, cellCa
sqlStatementEndCount=0, sqlStatementRowFetchCount=0, sqlStatementExecuteNanos=0, cellRequestCount=0}
```

SLIKA 35 - 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.