

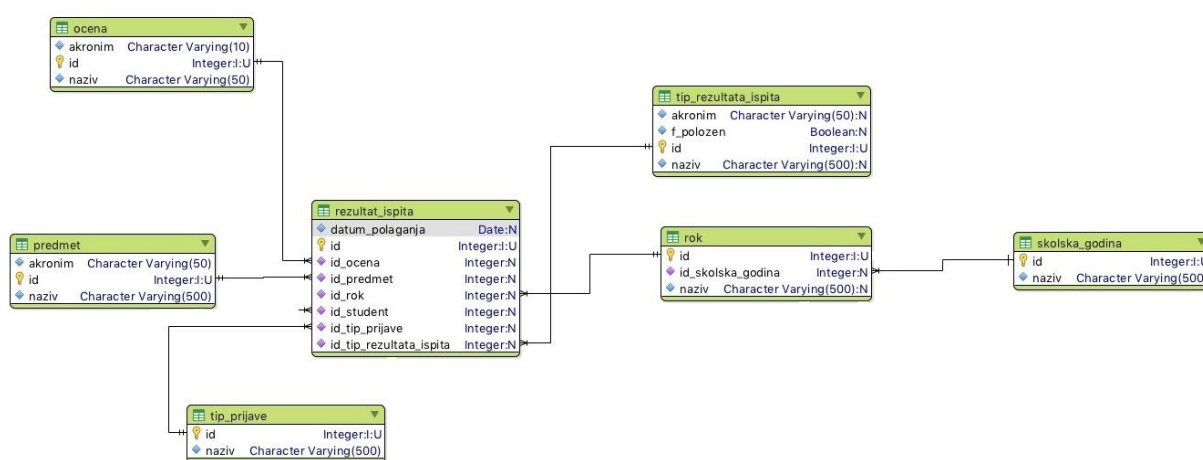
Kako ispravno dizajnirati skladište podataka?

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.

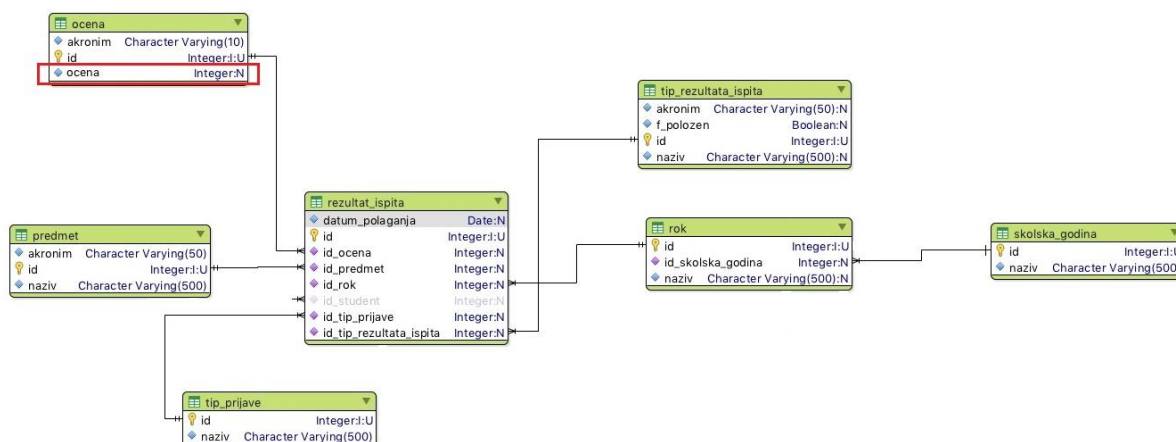
Kroz sledeći primer ćemo prikazati primer dizajniranja skladišta.

Analiziraćemo poslovni proces rezultata ispita, kao deo informacionog sistema fakulteta. Na slici ispod je prikazana šema operativne baze koja se odnosi na proces rezultata ispita, koji nam je od interesa.



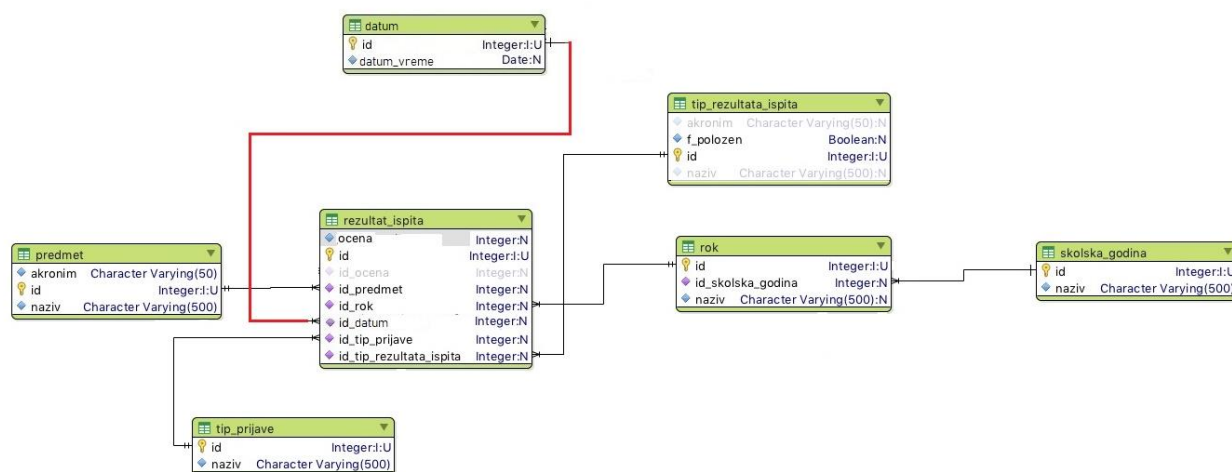
SLIKA 1 - ŠEMA DELA OPERATIVNE BAZE KOJI SE ODNOSI NA REZULTATE ISPITA

Primitićemo da neke kolone (odnosno strani ključevi u tabeli) nemaju veze ka drugim tabelama (npr. *id student* u tabeli *rezultat_ispita*). Ove kolone ukazuju na tabele koje nisu od interesa za našu analizu, pa ih odbacujemo u daljem dizajniranju skladišta podataka. Takođe, pošto želimo da nam ocena bude mera u rezultatu ispita, modifikovaćemo operativnu bazu, koloni *naziv* u tabeli *ocena* menjamo ime i tip, tako da sad imamo brojnu vrednost ocene u koloni *ocena*. (slika 2)



SLIKA 2 – IZMENA U OPERATIVNOJ BAZI I MODELOVANJE SKLADIŠTA PODATAKA

S' obzirom da želimo da vršimo analizu rezultata ispita po datumu polaganja, kreiramo novu tabelu u skladištu podataka u koju mapiramo sve različite datume polaganja ispita. Takođe, s' obzirom da ne želimo da vršimo analizu prema nazivu i akronimu tipa rezultata ispita, već samo prema tome da li je položen ili nije, kolone *naziv* i *akronim* ne dodajemo u našu tabelu *tip_rezultata ispita* u skladištu podataka.

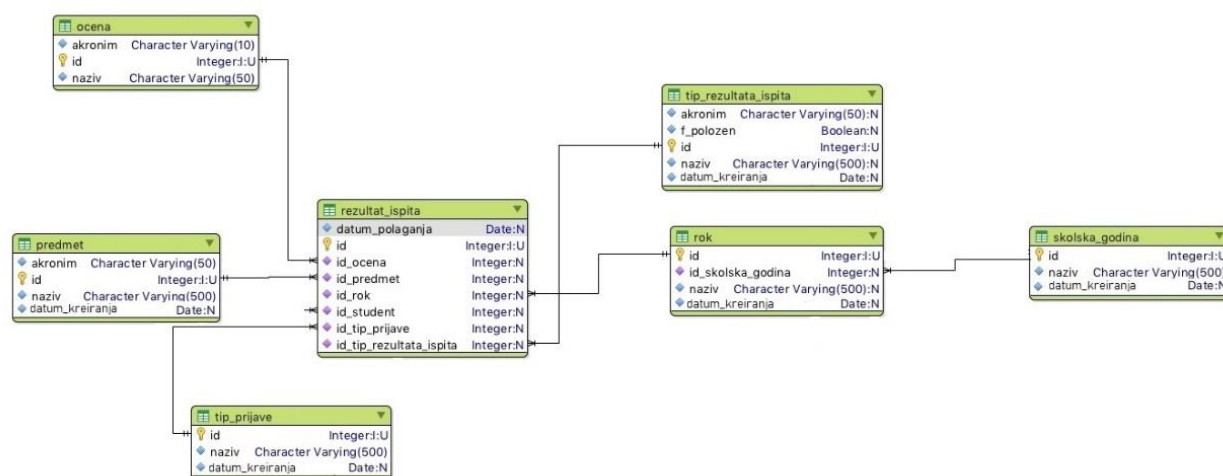


SLIKA 3 - MODELOVANJE SKLADIŠTA PODATAKA

Na slici 3 je prikazana šema skladišta podataka pomoću koje lako možemo vršiti analizu rezultata ispita po sledećim dimenzijama:

- datum polaganja ispita
- predmet koji je polagan
- rok u kome je polagan ispit
- tip prijave ispita (obavezan, izborni...)
- tip rezultata ispita (u ovom primeru razmatramo samo da li je položen ispit ili ne)

NAPOMENA: U skladištu podataka postoji još jedna tabela, *poslednja_izmena*, u kojoj čuvamo datum poslednjeg ažuriranja skladišta podataka. U skladu sa tim, promenice 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 4 - 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 *rezultat_ispita*, odnosno prvo moramo popuniti tabelu *skolska_godina* pa tek onda popunjavamo tabelu *rok* - tabela *rok* se puni podacima u transformaciji *Import dimensions 2*), najpre se izvršavaju transformacije koje nemaju ulaznu zavisnost. Pre svih transformacija „punjenja“, želimo da izbrisemo podatke iz skladišta kako ne bismo imali fatalnu grešku prilikom punjenja – duplikat istog primarnog ključa. Red izvršavanja transformacija je prikazan na sledećoj slici:

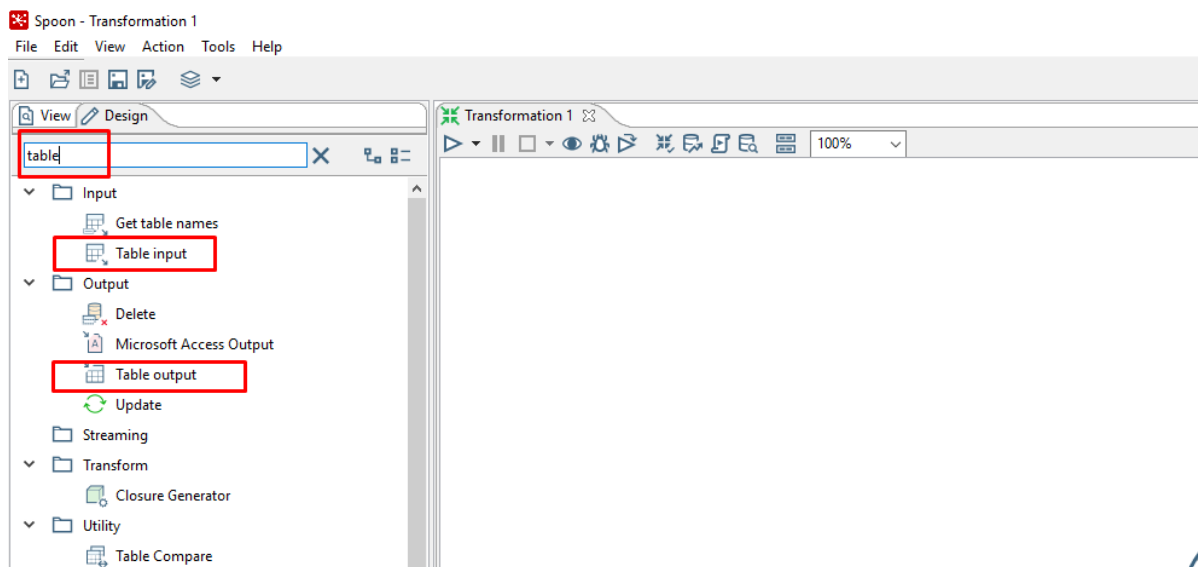


SLIKA 5 - REDOSLED IZVRŠAVANJA TRANSFORMACIJA TOKOM TOTALNOG PUNJENJA

Kako kreirati transformaciju?

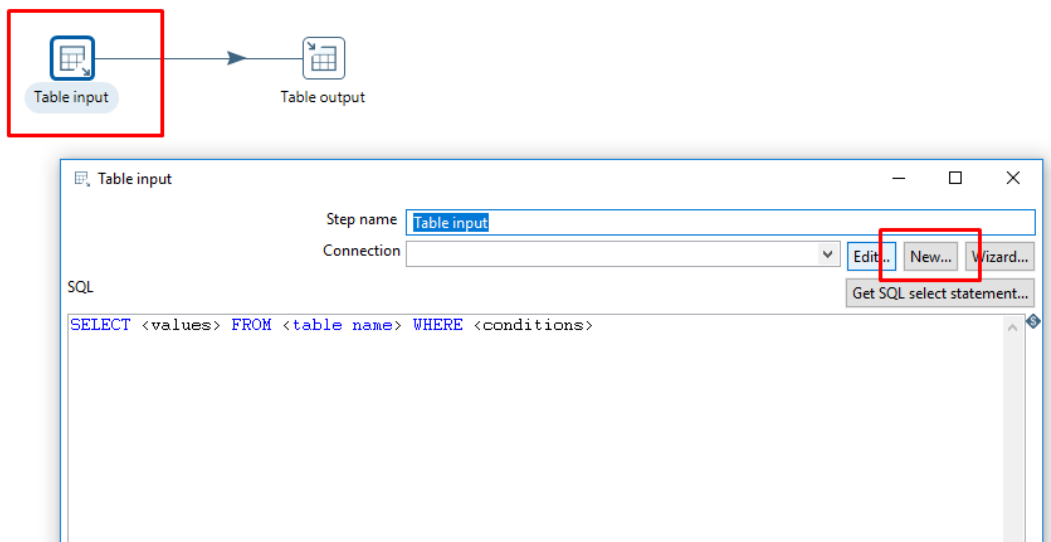
Za kreiranje transformacija koristimo alat *Spoon* koji možete preuzeti na [linku](#).

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).



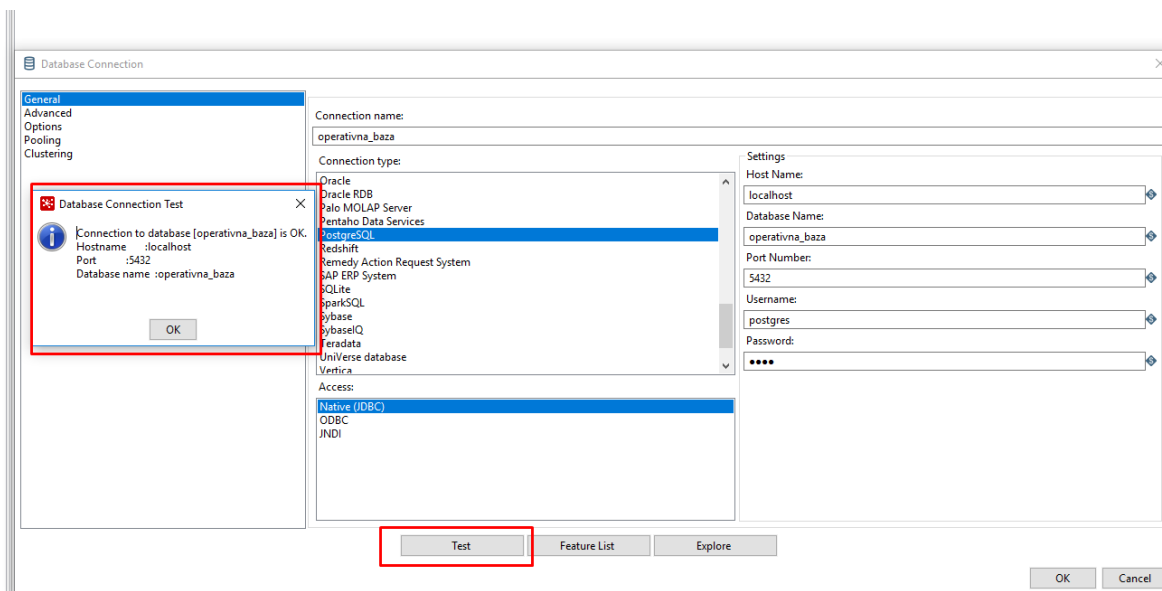
SLIKA 6 - KREIRANJE TRANSFORMACIJE - TABLE INPUT & TABLE OUTPUT

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)



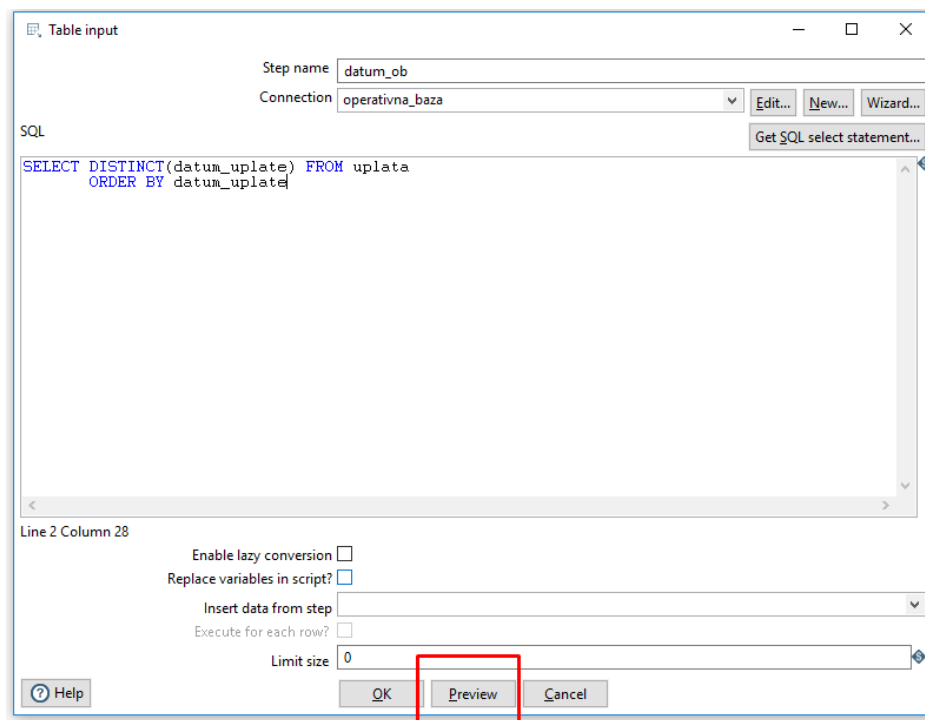
SLIKA 7 - KREIRANJE TRANSFORMACIJE - KREIRANJE KONEKCIJE 1

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



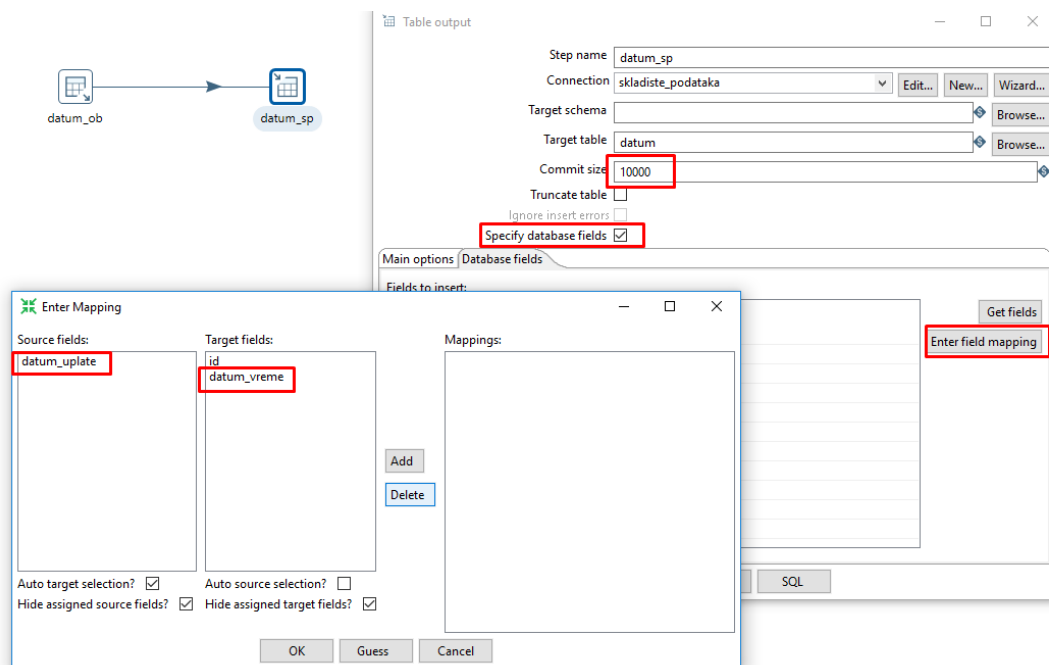
SLIKA 8 - KREIRANJE TRANSFORMACIJE - KREIRANJE TRANSFORMACIJE 2

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 u toku kojih su polagani ispiti. Kako bismo odmah proverili validnost upita, klikom na *Preview* izvršavamo upit.



SLIKA 9 - UPIT U TABLE_INPUT ELEMENTU

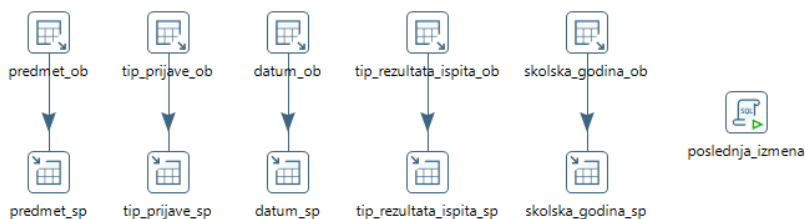
Ukoliko smo upit ispravno napisali, prelazimo na podešavanje *Table output* stavke. Kao i kod prethodne stavke, mora se izabrati (odnosno napraviti) konekcija prema bazi/skladištu. Nakon toga biramo koju tabelu u skladištu punimo podacima, kao i maksimalni broj redova koje prenosimo u skladište. Najvažniji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



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

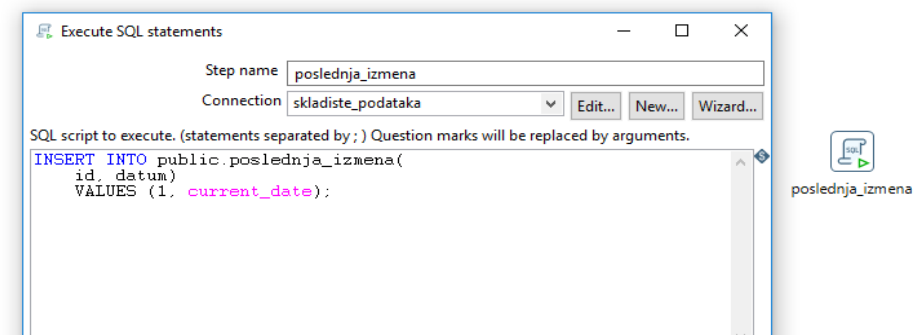
Konkretno, u gore prikazanom primeru, različite datume koje smo selektovali iz tabele *rezultat_ispita* mapiramo u kolonu *datum_vreme*, dok kolona *id* inkrementalno dobija vrednost, što smo podesili u konfiguraciji baze.

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



SLIKA 11 - TRANSFORMACIJA - IMPORT DIMENSIONS 1

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



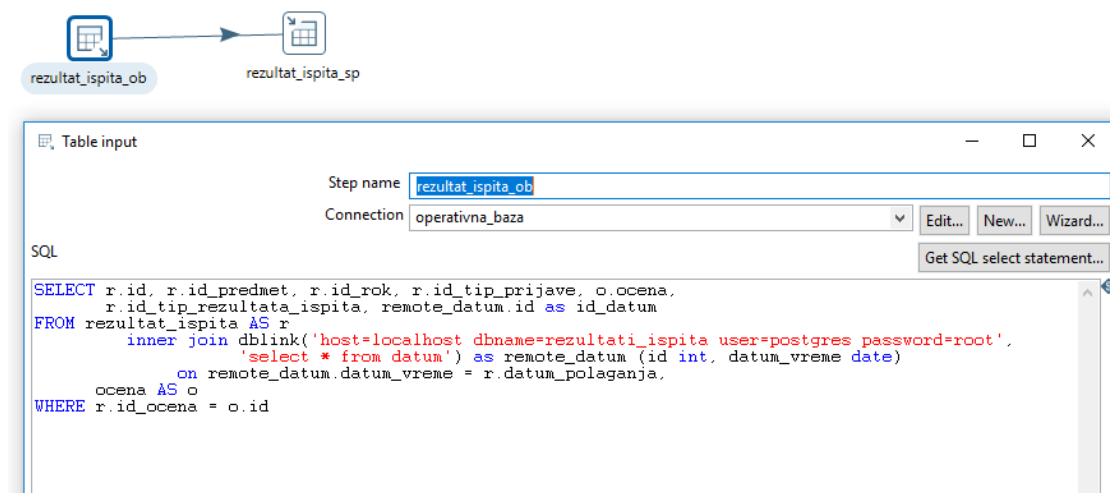
SLIKA 12 - AZURIRANJE DATUMA POSLEDNJE IZMENE

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



SLIKA 13 – TRANSFORMACIJA IMPORT DIMENSIONS 2

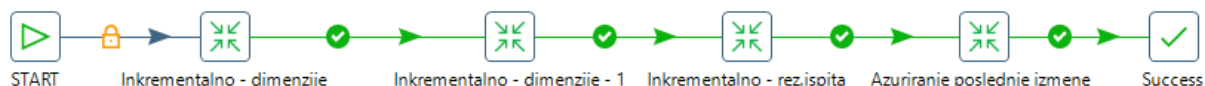
Nakon što smo popunili tabele dimenzija, možemo popuniti i tabele koje sadrže strane ključeve ka ovim tabelama, odnosno tabelu podataka, *rezultat_ispita*. Tabelu *rezultat_ispita* punimo na sličan način kao i tabele dimenzija – složenijim SQL upitom kojim filtriramo podatke od interesa u skladištu podataka,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



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

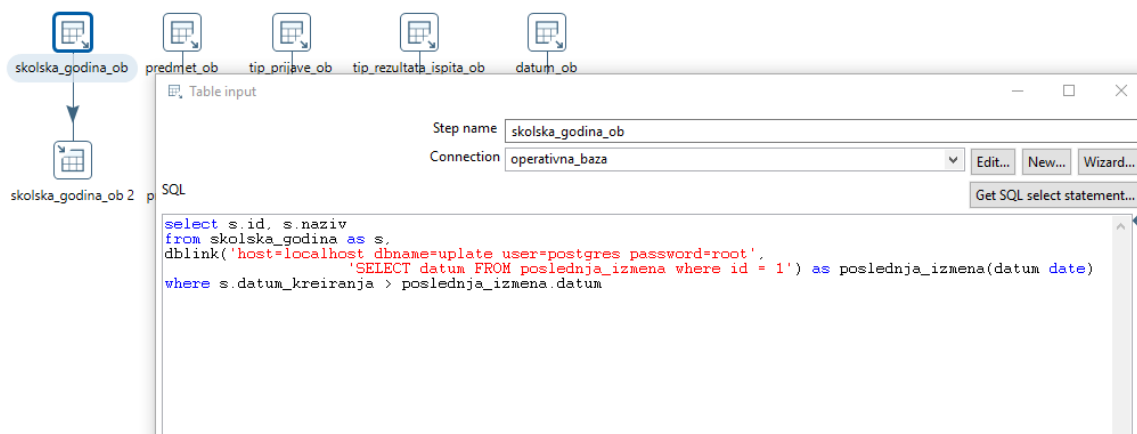
Kako popuniti skladište podacima koji se unose nakon totalnog punjenja?

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 *rezultat_ispita*).



SLIKA 15 -REDOSED IZVRŠAVANJA TRANSFORMACIJA TOKOM INKREMENTALNOG PUNJENJA

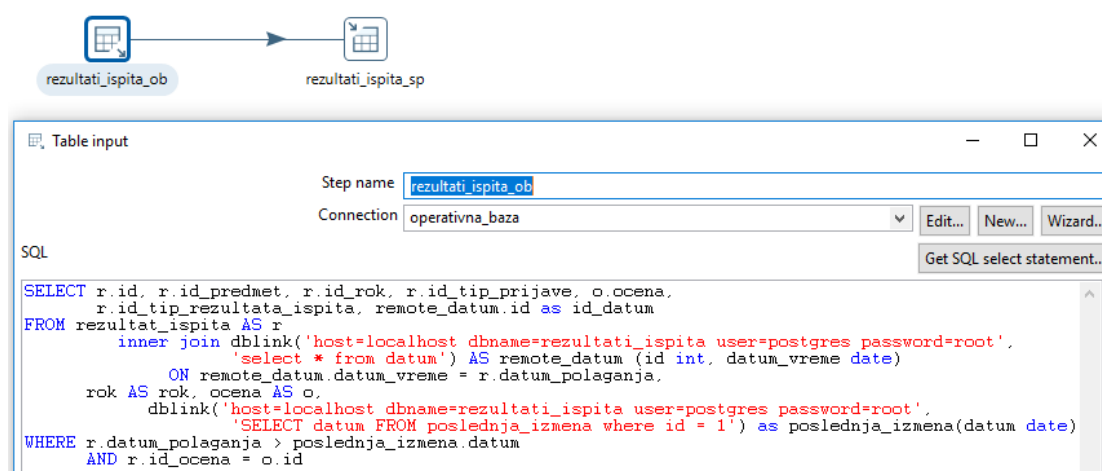
Inkrementalno punjenje dimenzija je implementirano skoro identično kao totalno punjenje (element *Table input* filtrira podatke koji su nam potrebni, dok u elementu *Table output* mapiramo dobijene podatke iz tabela operativne baze u podatke tabela skladišta podataka). Primer inkrementalnog punjenja tabele *skolska_godina* 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 *rezultat_ispita*, vršimo na sličan način kao drugi primer totalnog punjenja (složeniji SQL upit):



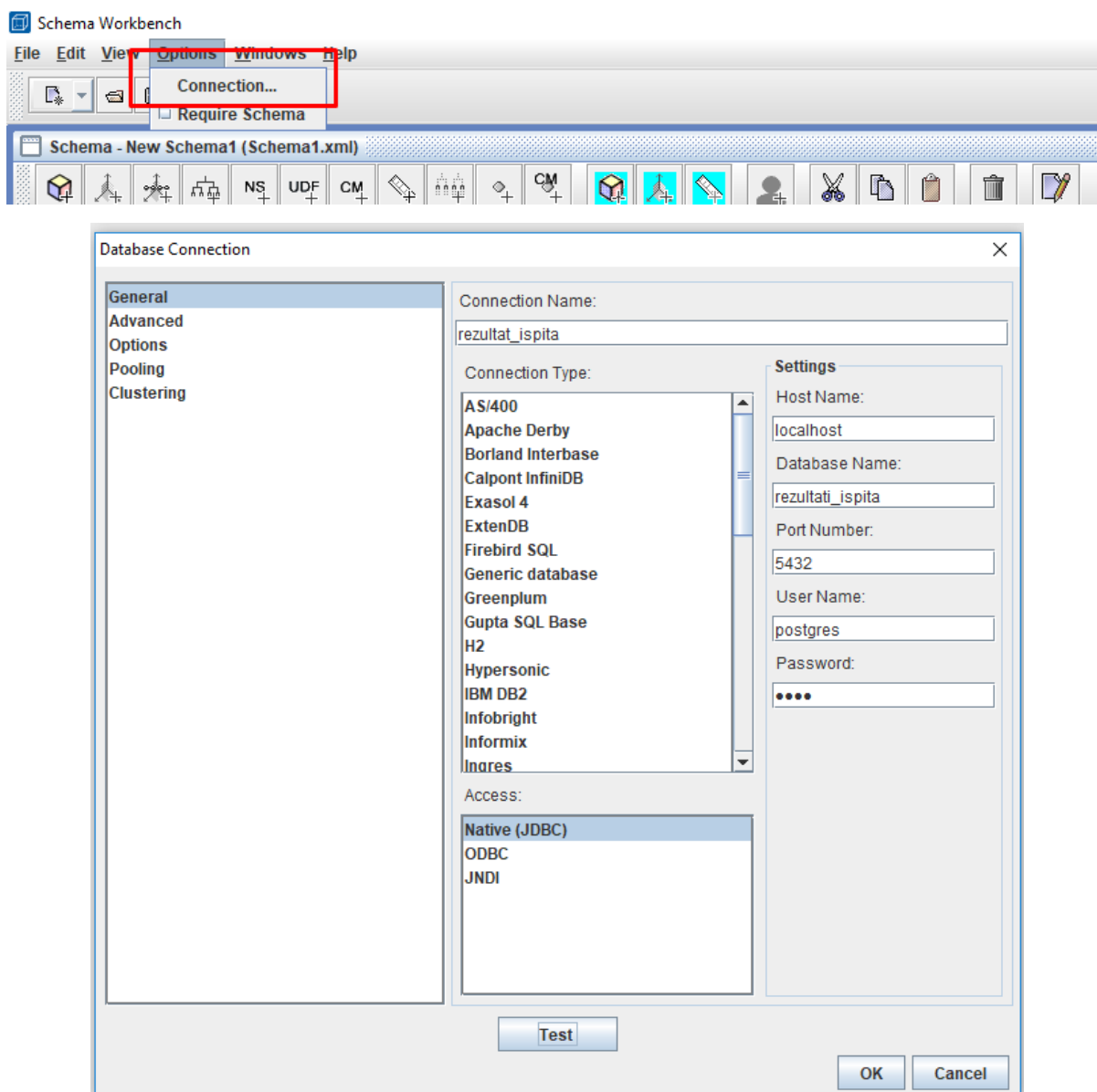
SLIKA 17 - UPIT ZA DOBIJANJE PODATAKA KOJI SU UBAČENI U OPERATIVNU BAZU NAKON POSLEDNJEG AŽURIRANJA SKLADIŠTA - 2

Najpre smo tabelu *rezultat_ispita* spojili sa tabelom *datum* (iz skladišta podataka) po koloni *datum_polaganja*, zatim filtrirali one rezultate ispita koje su izvršene nakon poslednjeg ažuriranja skladišta podataka, i na kraju proširili filtrirane redove sa potrebnim podacima iz tabele *ocena*.

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.

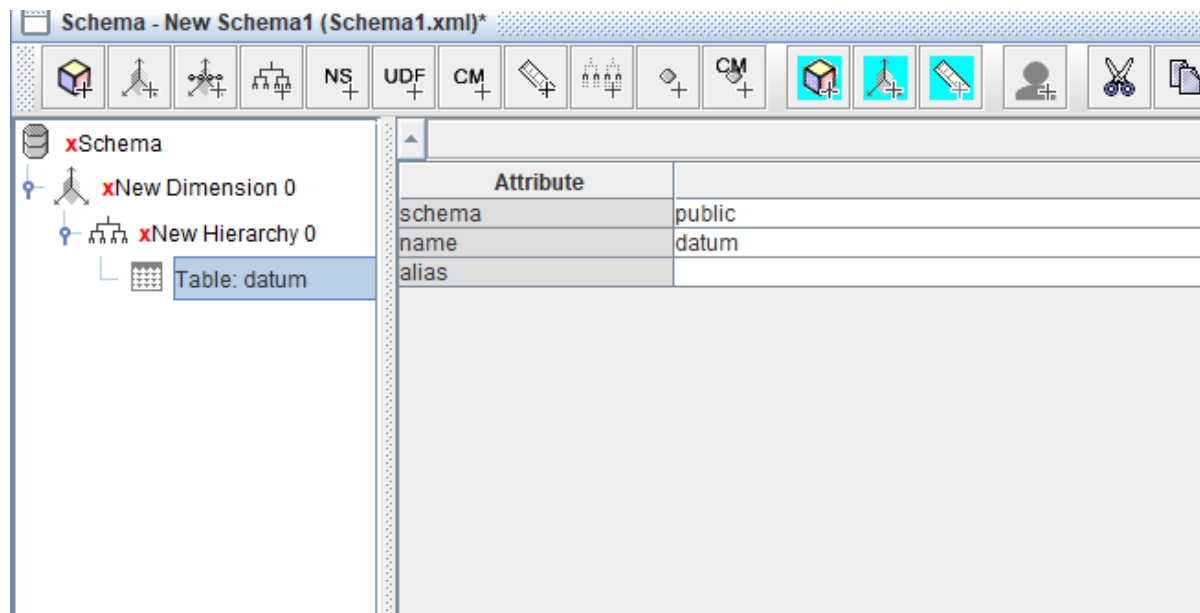


SLIKA 18 - KREIRANJE KONEKCIJE U SCHEMA WORKBENCH ALATU

Postupak kreiranja šeme (logičkog modela) se sastoji od:

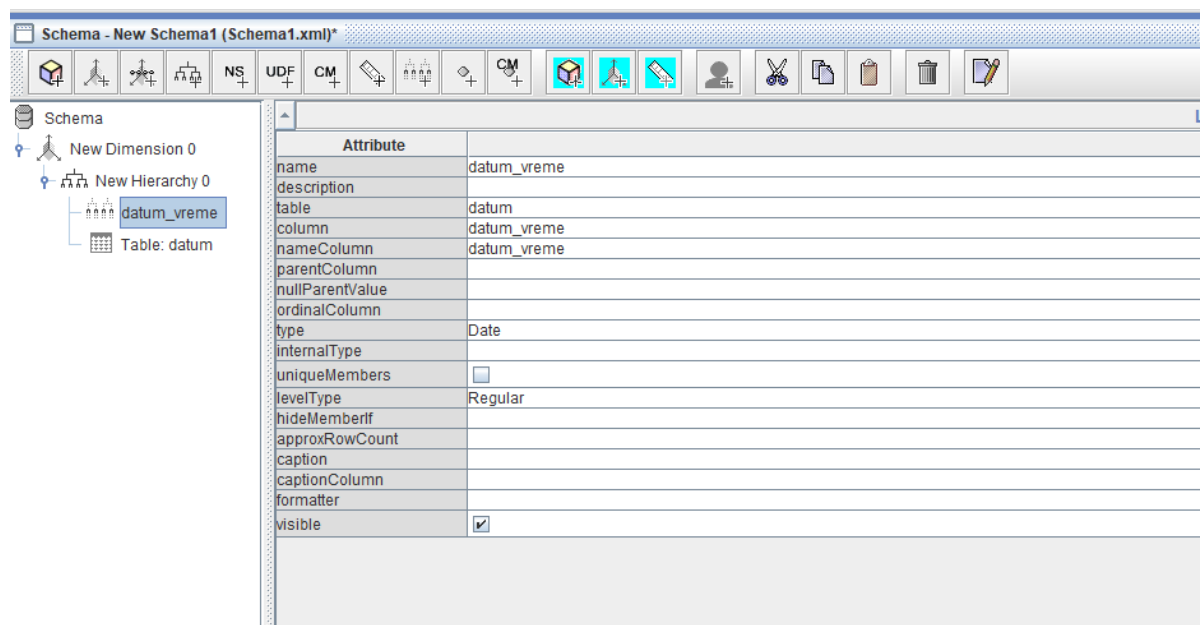
- kreiranja *dimenzija* (logičko predstavljanje tabela dimenzija)
- kreiranja *kocki* (logičko predstavljanje tabela podataka)

Svaka dimenzija mora imati referencu na tabelu skladišta podataka:



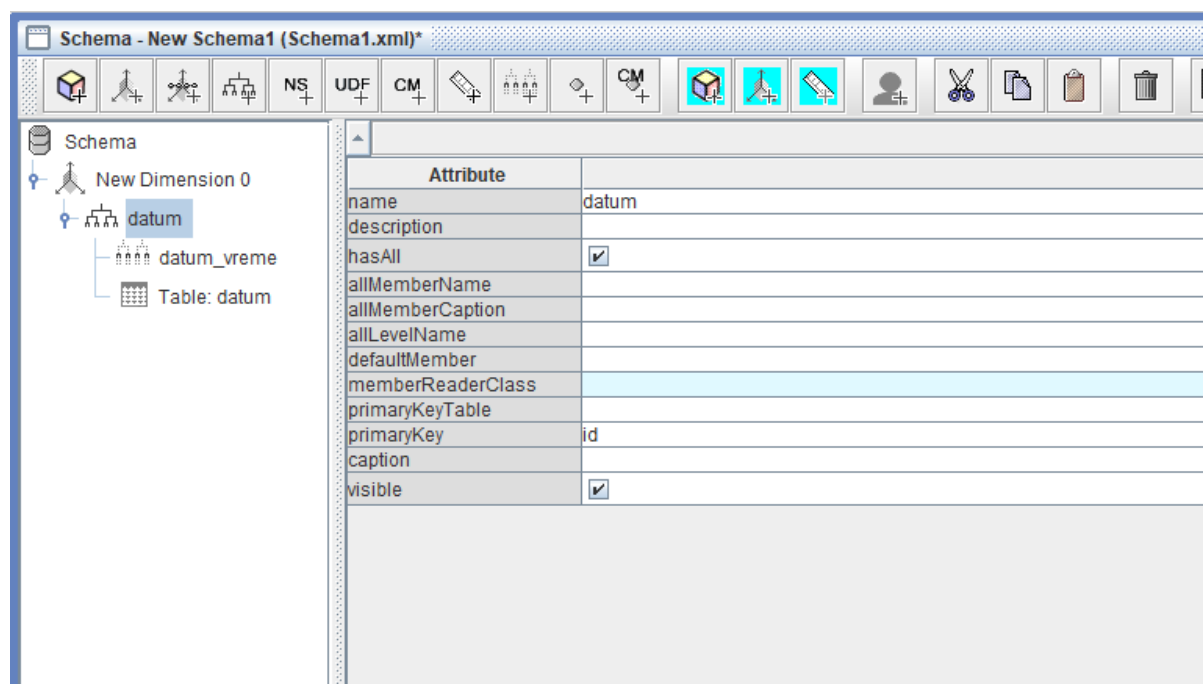
SLIKA 19 - KREIRANJE DIMENZIJE - TABELA

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



SLIKA 20 - KREIRANJE DIMENZIJE - NIVO

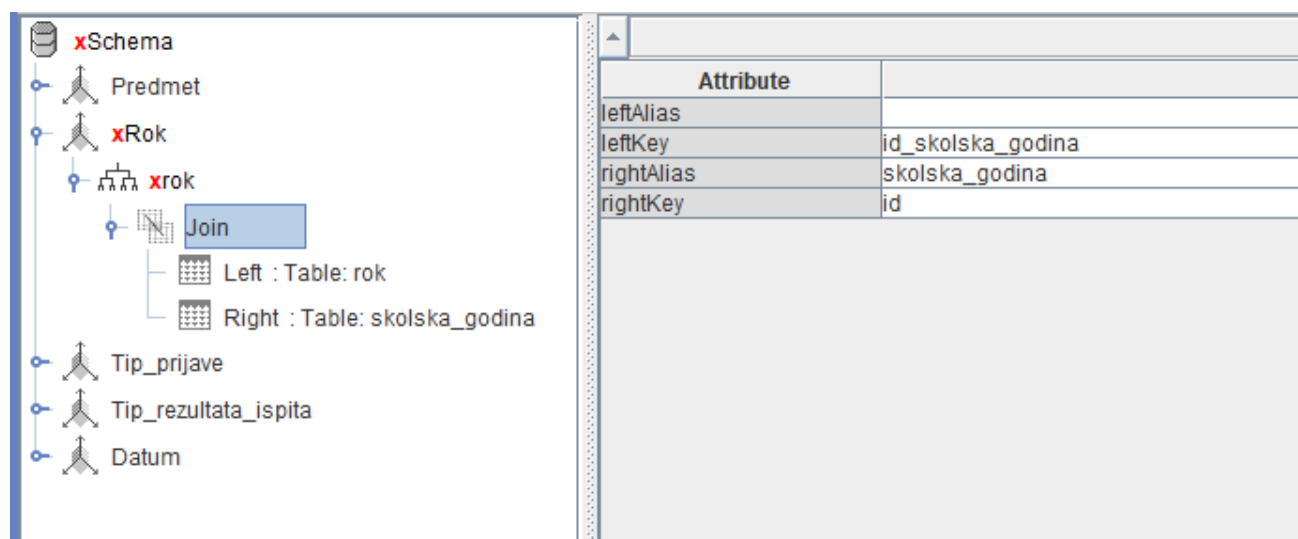
Nakon kreiranja reference na tabelu i nivoa za svaku kolonu tabele dimenzija, vraćamo se na hijerarhiju, dajemo joj ime i postavljamo primarni ključ.



SLIKA 21 - KREIRANJE TABELE - HIJERARHIJA

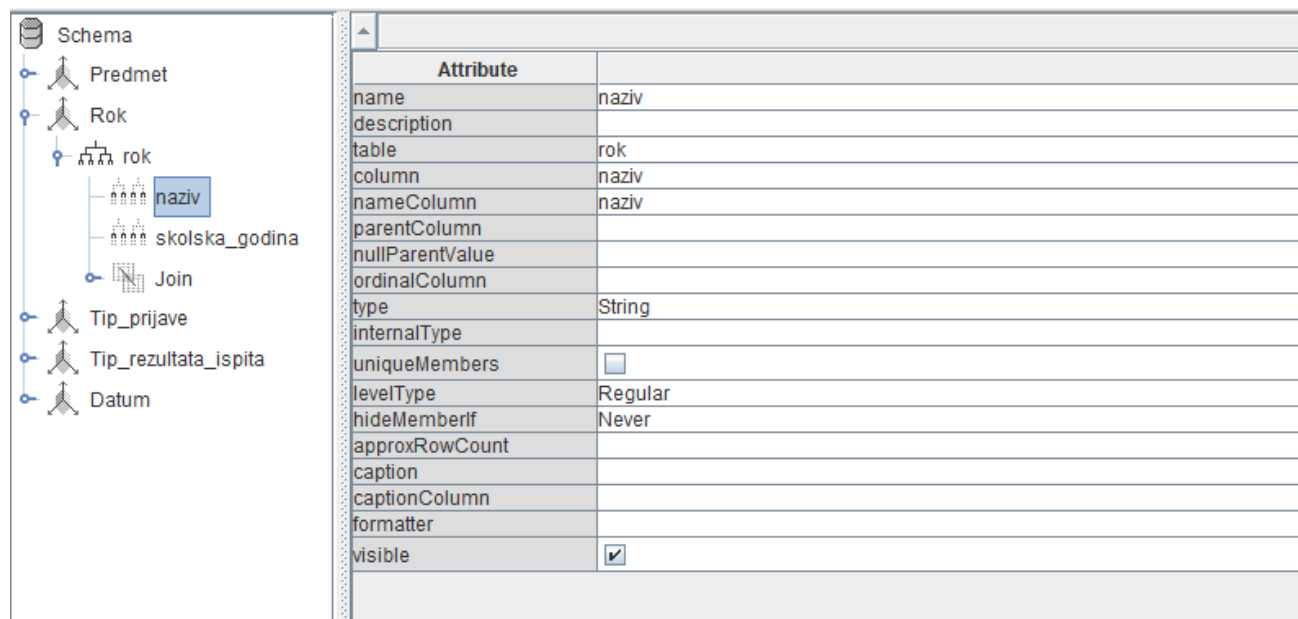
Prethodni postupak treba ponoviti za sve tabele dimenzija iz skladišta podataka : *predmet*, *tip_prijave* i *tip_rezultata_ispita*.

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



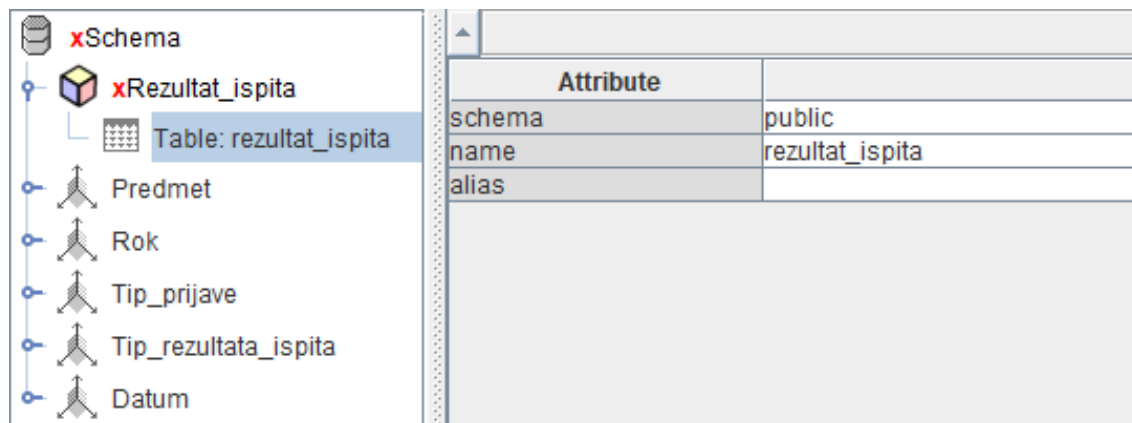
SLIKA 22 - DODAVANJE SPAJANJA TABELA ROK I SKOLSKA_GODINA

Kao i kod ostalih dimenzija, i kod dimenzije *rok*, dodajemo nivoe i to za kolonu *naziv* iz tabele *rok* i kolonu *naziv* iz tabele *rok*:



SLIKA 23 - DODAVANJE NIVOVA U DIMENZIJU ROK

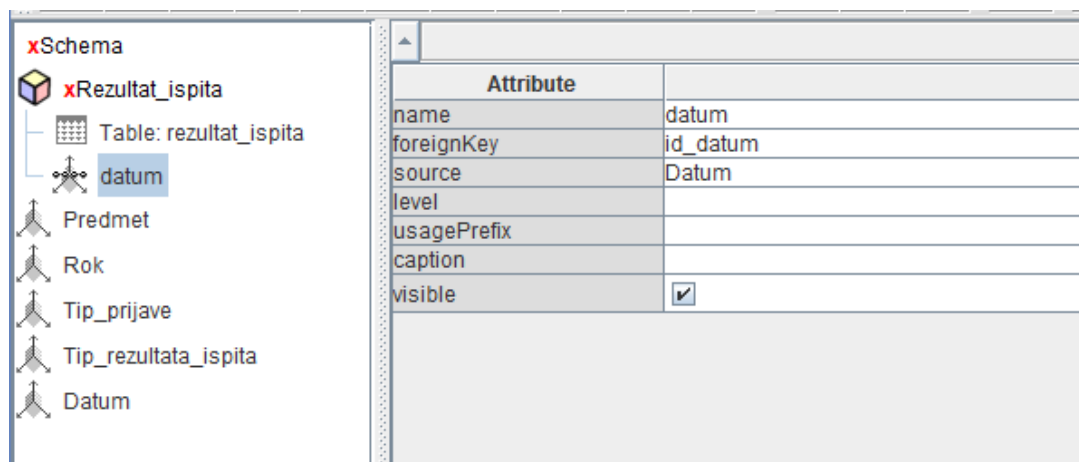
Nakon kreiranja dimenzija, prelazimo na kreiranje *kocki*, odnosno logičku predstavu tabela podataka. Kao i *dimenzije*, i *kocka* mora imati referencu na fizičku tabelu iz skladišta podataka.



SLIKA 24 - KREIRANJE KOCKE

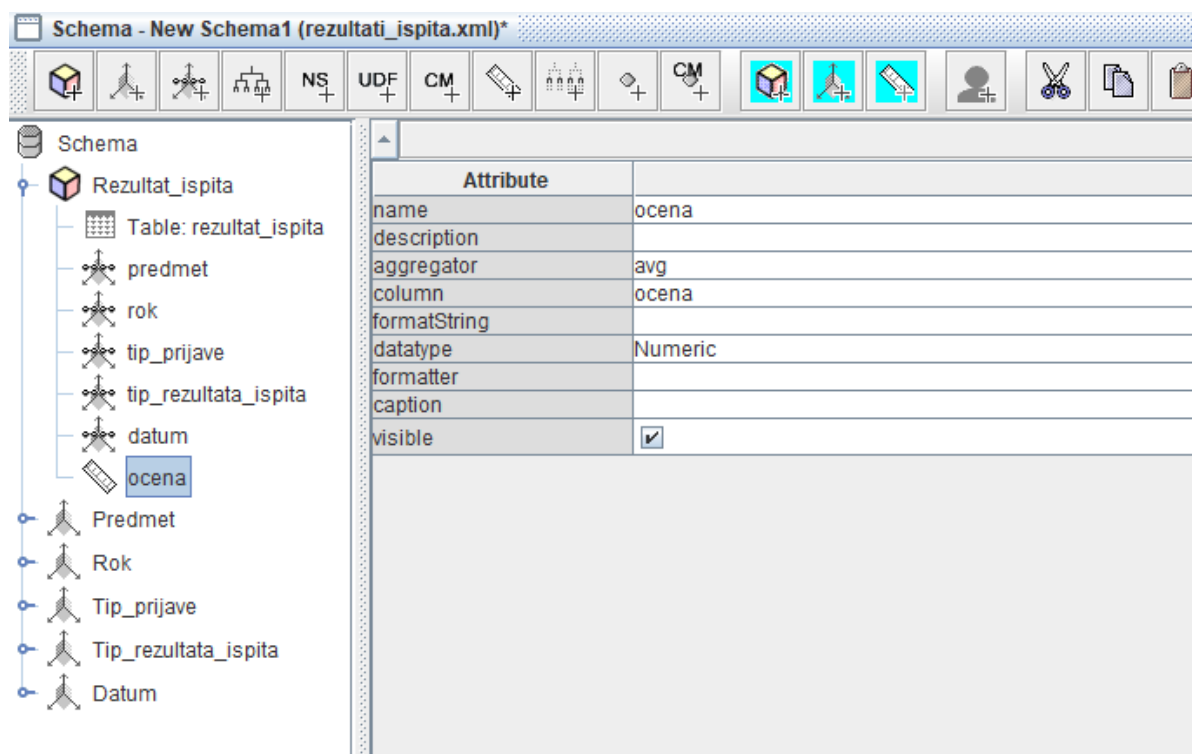
Pored referencu na fizičku tabelu, *kocka* mora sadržati još dva elementa:

- *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

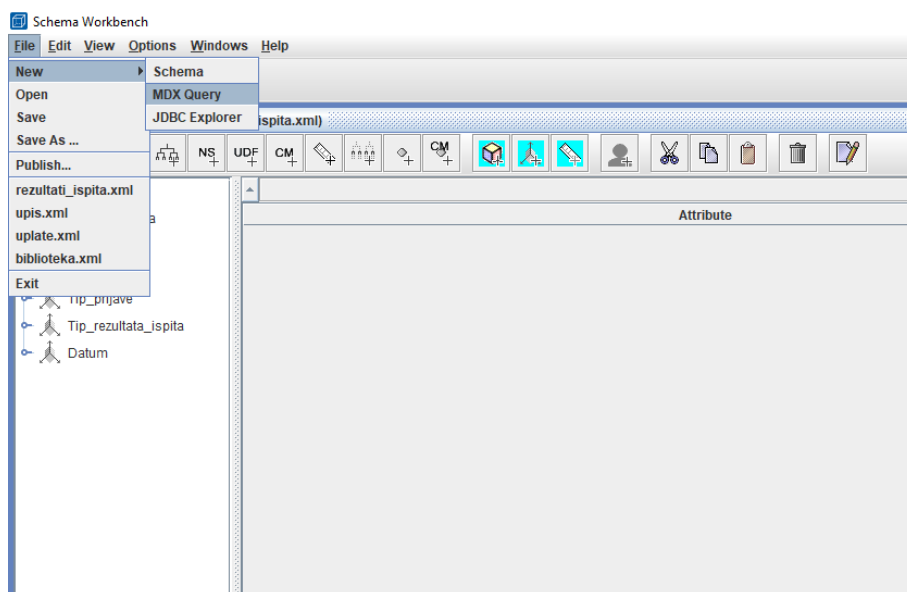
- *Measure*, odnosno mera – podatak koji dobijamo ka rezultat analize



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.

- Broj položenih odnosno nepoloženih ispita po roku
 - SQL

```
SELECT rok.naziv, SUM(CASE WHEN tip.f_polozen = true THEN 1 ELSE 0 END) AS položio,  
SUM(CASE WHEN tip.f_polozen = false THEN 1 ELSE 0 END) AS nije_položio  
FROM rezultat_ispita AS rez, rok AS rok, tip_rezultata_ispita AS tip  
WHERE rez.id_rok = rok.id AND rez.id_tip_rezultata_ispita = tip.id  
GROUP BY rok.naziv
```



SLIKA 28 - BROJ POLOŽENIH ODNOSNO NEPOLOŽENIH ISPITA PO ROKU SQL

- MDX

```
SELECT {[Measures].[count]} ON COLUMNS,  
CROSSJOIN ({[Tip_rezultata_ispita].Members}, [Rok].Members) ON ROWS  
FROM Rezultat_ispita
```

```

-12-11 03:42:41,525 DEBUG [mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[rok].[#null]
-12-11 03:42:41,526 DEBUG [mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[tip_prijave].[#null]
-12-11 03:42:41,526 DEBUG [mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[tip_rezultata_ispita].[#null]
-12-11 03:42:41,526 DEBUG [mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[datum].[#null]
-12-11 03:42:41,527 DEBUG [mondrian.olap.ResultBase] RolapResult<init>: FREE_MEMORY: 974232kb 70.57%
-12-11 03:42:41,527 DEBUG [mondrian.rolap.RolapStar] RolapStar.clearCachedAggregations: schema=New Schema1, star=rezultat_ispita
-12-11 03:42:41,527 DEBUG [mondrian.mdx] 1291: exec: 34 ms
-12-11 03:42:41,527 DEBUG [mondrian.server.monitor] ExecutionInfo{executionId=1291, phaseCount=1, cellCacheRequestCount=0, cellCacheHitCount=0, cellCachePendingCount=0, sqlStatementStartCount=0, sqlStatementExecuteCount=0, sqlStatementEndCount=0, sqlStatementRowFetchCount=0, sqlStatementExecuteNanos=0, cellRequestCount=0}
-12-11 03:42:41,527 DEBUG [mondrian.server.monitor] ExecutionEndEvent(1291)

```

SLIKA 29 - BROJ POLOŽENIH ODNOSNO NEPOLOŽENIH ISPITA PO ROKU MDX

- Broj položenih odnosno nepoloženih ispita u jednom roku

- SQL

```

SELECT SUM(CASE WHEN tip.f_polozen = true THEN 1 ELSE 0 END) AS polozio,
       SUM(CASE WHEN tip.f_polozen = false THEN 1 ELSE 0 END) AS nije_polozio
FROM rezultat_ispita AS rez, rok AS rok, tip_rezultata_ispita AS tip
WHERE rez.id_rok = rok.id AND rez.id_tip_rezultata_ispita = tip.id
      AND rok.naziv = 'Апрелску 2009/10'

```



SLIKA 30 - BROJ POLOŽENIH ODNOSNO NEPOLOŽENIH ISPITA U JEDNOM ROKU SQL

- MDX

```

SELECT {[Measures].[count]} ON COLUMNS,
       CROSSJOIN ({[Tip_rezultata_ispita].Children}, [Rok].[Апрелску 2009/10]) ON ROWS
FROM Rezultat_ispita

```

```

946 DEBUG [mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[tip_prijave].[#null]
946 DEBUG [mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[tip_rezultata_ispita].[#null]
946 DEBUG [mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[datum].[#null]
946 DEBUG [mondrian.olap.ResultBase] RolapResult<init>: FREE_MEMORY: 961600kb 69.66%
946 DEBUG [mondrian.rolap.RolapStar] RolapStar.clearCachedAggregations: schema=New Schema1, star=rezultat_ispita
946 DEBUG [mondrian.mdx] 1300: exec: 8 ms
946 DEBUG [mondrian.server.monitor] ExecutionInfo{executionId=1300, phaseCount=1, cellCacheRequestCount=0, cellCacheHitCount=0, cellCachePendingCount=0, sqlStatementStartCount=0, sqlStatementExecuteCount=0, sqlStatementEndCount=0, sqlStatementRowFetchCount=0, sqlStatementExecuteNanos=0, cellRequestCount=0}

```

SLIKA 31 - BROJ POLOŽENIH ODNOSNO NEPOLOŽENIH ISPITA U JEDNOM ROKU MDX

- Prosečna ocena na predmetu "Bankarstvo"

- SQL

```

SELECT AVG(o.ocena)
FROM rezultat_ispita AS r, ocena AS o, predmet AS p
WHERE r.id_ocena = o.id
      AND r.id_predmet = p.id
      AND p.naziv = 'Банкарство'

```

Data Output Explain Messages Notifications Query History

Successfully run. Total query runtime: 87 msec.
1 rows affected.

SLIKA 32 - PROSEČNA OCENA NA PREDMETU "BANKARSTVO" SQL

- MDX

```
SELECT {[Measures].[ocena]} ON COLUMNS,  
filter ({[Predmet].Members}, [Predmet].naziv.CurrentMember in {[Банкарство]}) ON ROWS  
FROM Rezultat_ispita
```

```
018-12-11 03:48:29,087 DEBUG [mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[datum].[#null]  
018-12-11 03:48:29,094 DEBUG [mondrian.rolap.RolapCube] RolapCube.getUsages: name=Measures  
018-12-11 03:48:29,094 DEBUG [mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[predmet].[#null]  
018-12-11 03:48:29,094 DEBUG [mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[rok].[#null]  
018-12-11 03:48:29,094 DEBUG [mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[tip_prijave].[#null]  
018-12-11 03:48:29,094 DEBUG [mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[tip_rezultata_ispita].[#null]  
018-12-11 03:48:29,094 DEBUG [mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[datum].[#null]  
018-12-11 03:48:29,095 DEBUG [mondrian.olap.ResultBase] RolapResult<init>: FREE_MEMORY: 948695kb 68.72%  
018-12-11 03:48:29,095 DEBUG [mondrian.rolap.RolapStar] RolapStar.clearCachedAggregations: schema=New Schema1, star=rezultat_ispita  
018-12-11 03:48:29,095 DEBUG [mondrian.mdx] 1336: exec: 16 ms  
018-12-11 03:48:29,095 DEBUG [mondrian.server.monitor] ExecutionInfo{executionId=1336, phaseCount=1, cellCacheRequestCount=0, cellCacheHit  
StatementExecuteCount=0, sqlStatementEndCount=0, sqlStatementRowFetchCount=0, sqlStatementExecuteNanos=0, cellRequestCount=0}  
018-12-11 03:48:29,099 DEBUG [mondrian.server.monitor] ExecutionEndEvent(1336)
```

SLIKA 33 - PROSEČNA OCENA NA PREDMETU "BANKARSTVO" MDX

- Prosečna ocena ispita po tipu prijave (obavezan predmet, izborni predmet,...)
 - SQL

```
SELECT t.naziv, AVG(o.ocena)  
FROM rezultat_ispita AS r, ocena AS o, tip_prijave AS t  
WHERE r.id_ocena = o.id  
AND r.id_tip_prijave = t.id  
GROUP BY t.naziv
```

Data Output Explain Messages Notifications Query History

Successfully run. Total query runtime: 136 msec.
3 rows affected.

SLIKA 34 - PROSEČNA OCENA ISPITA PO TIPU PRIJAVE SQL

- MDX

```
SELECT {[Measures].[ocena]} ON COLUMNS,  
order(filter ({[Predmet].Children}, [Measures].[count] > 500), [Measures].[ocena], DESC) ON ROWS  
FROM Rezultat_ispita
```

```
mondrian.rolap.RolapCube] RolapCube.getUsages: name=Measures  
mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[predmet].[#null]  
mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[rok].[#null]  
mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[tip_prijave].[#null]  
mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[tip_rezultata_ispita].[#null]  
mondrian.rolap.RolapMember] RolapMember.makeUniqueName: uniqueName=[datum].[#null]  
mondrian.olap.ResultBase] RolapResult<init>: FREE_MEMORY: 927427kb 67.18%  
mondrian.rolap.RolapStar] RolapStar.clearCachedAggregations: schema=New Schema1, star=rezultat_ispita  
mondrian.mdx] 1366: exec: 12 ms  
mondrian.server.monitor] ExecutionInfo{executionId=1366, phaseCount=1, cellCacheRequestCount=0, cellCacheHit  
StatementEndCount=0, sqlStatementRowFetchCount=0, sqlStatementExecuteNanos=0, cellRequestCount=0}
```

SLIKA 35 - PROSEČNA OCENA ISPITA PO TIPU PRIJAVE MDX