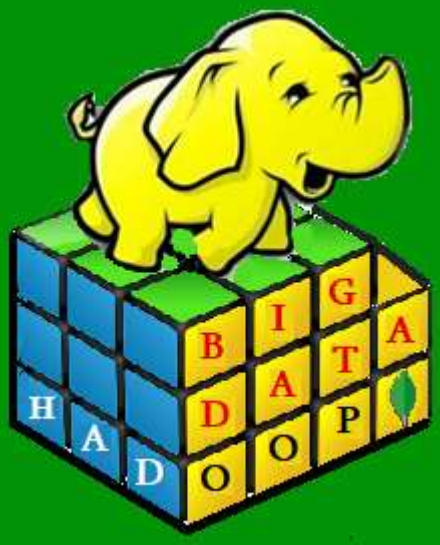
დოქტორანტის ბიბლიოთეკა

ბადრი მეფარიშვილი

დიდ მონაცემთა ტექნოლოგია – ბიზნესანალიტიკა, დეველოპმენტი და იმპლემენტაცია

(პრაქტიკულის მეთოდური მითითებანი)



„სტუ-ს IT კონსალტინგის სამეცნიერო ცენტრი”

საქართველოს ტექნიკური უნივერსიტეტი

ბადრი მეფარიშვილი

სადოქტორო პროგრამა „ინფორმატიკა“

დიდ მონაცემთა ტექნოლოგია – ბიზნესანალიტიკა, დეველოპმენტი და იმპლემენტაცია

(პრაქტიკული სამუშაოს მეთოდური მითითებანი)

დამტკიცებულია:

სტუ-ს „IT კონსალტინგის სამეცნიერო ცენტრის“ სარედაქციო კოლეგიის მიერ
ოქმი N7, 15.10.2020

თბილისი
2020

უაკ 004.5

განხილულია დიდ მონაცემთა ტექნოლოგიების გამოყენების მიზნით ბიზნესანალიტიკის, დეველოპმენტისა და იმპლემენტაციის პროცესების საკითხები, თანამედროვე კორპორატიული საინფორმაციო სისტემების ფართო სპექტრისათვის. წარმოდგენილია ტიპური პრაქტიკული მაგალითები მონაცემთა საცავებში ინფორმაციის ოპერატიული დამუშავების ეფექტური მეთოდების გამოყენების შესახებ, აგრეთვე ელექტრონული დოკუმენტაციის, მომხმარებლის იდენტიფიკაციისა და ინფორმაციული ტექნოლოგიების გამოყენებით, ბიზნესპროექტების იმპლემენტაციის, IT-სერვისის ამოცანების, მონაცემთა შენახვისა და დაცვის კონცეფციებით. შემოთავაზებულია Hadoop-ის და Mapreduce-ს პრაქტიკული გამოყენების საილუსტრაციო მაგალითის გარჩევა რეალური Dataset-ებისათვის, აგრეთვე მონაცემთა საცავისა და OLAP კუბების გამოყენების მაგალითი უმაღლესი განათლების სისტემაში ინფორმაციული ტექნოლოგიების საფუძველზე. მეთოდური მითითებანი რეკომენდებულია ინფორმატიკის სპეციალობის დოქტორანტებისათვის, ინფორმაციული და კომუნიკაციური ტექნოლოგიების სფეროში (ICT 0613).

რეცენზენტები:

ასოც. პროფ. **ი. ქართველიშვილი** (სტუ)
პროფ. **ე. თურქია** (საქ. ეროვნული ბანკი)

ISBN 978-9941-8-2869-0

სასწავლო კურსის მიზანი

თანამედროვე კორპორატიულ საინფორმაციო სისტემების ფართო სპექტრისათვის ბიზნეს ანალიტიკის დეველოპმენტსა და იმპლემენტაციის პროცესში დიდი მონაცემების ტექნოლოგიების გამოყენების შესწავლა. დიდ მონაცემთა პარადიგმასა და პლატფორმების შესახებ ფუნდამენტური ცოდნის შეძენა და მისი შემდგომი გამოყენება მეტად სასარგებლო იქნება როგორც ინფორმატიკის აკადემიური დოქტორის მოსაპოვებლად გამიზნული ნაშრომის შესრულების პროცესში, ასევე პროფესიულ ერუდიციის გაფართოების თვალსაზრისით.

საგნის შესწავლის შედეგად მიღებული ცოდნა და შეძენილი უნარები

1. განსაზღვრავს ორგანიზაციის, კომპანიის ინფრასტრუქტურასა და ბიზნეს არქიტექ-ტურაში, ბიზნეს პროცესებში დიდი მოცულობის მონაცემთა შენახვის, დამუშავების, ანალიზისა და გადაწყვეტილების მიღების ეტაპებზე დიდ მონაცემთა სისტემების დეველოპმენტისა და იმპლემენტაციის პრობლემებს;
2. უსადაგებს კვლევის მეთოდებსა და ალგორითმებს, IT სერვისების პლატფორმებისა და აპლიკაციების გამოყენების გადაწყვეტის გზებს;
3. აანალიზებს დიდ მონაცემთა სისტემების გამოყენების მნიშვნელობას სხვადასხვა სფეროში;
4. აიდენტიფიცირებს დიდ მონაცემთა სისტემების შესაძლებლობებს, კერძოდ დიდ მონაცემთა შეგროვების, შენახვისა და დამუშავების გამოყენება პროგნოზირების, ანალიზისა და მართვის პრობლემის გადაწყვეტაში;
5. განსაზღვრავს დიდ მონაცემთა შეგროვების, შენახვისა და დამუშავების ტექნოლო-გიების იმპლემენტაციის შესახებ მიღებული შედეგების ეფექტურობას;
6. აფორმირებს ლიტერატურის გამოყენებით მიღებული გამოცდილების საფუძველ¬ზე, დიდ მონაცემთა შეგროვების, შენახვისა და დამუშავების, გამოყენება, პროგნოზირების, ანალიზისა და მართვის პროცესებს;
7. აწარმოებს დიდ მონაცემთა ანალიზის სისტემებში მიღებული კვლევის შედეგების დასაბუთებულად და გარკვევით წარმოჩენას, საერთაშორისო სამეცნიერო საზოგადოებასთან დიდ მონაცემთა პარალელური დამუშავების საკითხებთან დაკავშირებულ პოლემიკაში ჩართვას.

საათების განაწილება (სტუდენტის დატვირთვა)

კრედიტების რაოდენობა 8. ლექცია 30 სთ., პრაქტიკული (ჯგუფში მუშაობა) – 30 სთ. დამოუკიდებელი მუშაობა 137 სთ.

პრაქტიკული მეცადინეობების თემების დასახელება და შინაარსი

1. თანამედროვე ტიპიური საწარმოს მაგალითების მიმოხილვა. კორპორატიული სისტემების, მცირე და საშუალო ბიზნესის მართვის თავისებურებების განხილვა;

2. საინფორმაციო სისტემების მაგალითების გარჩევა საწარმოთა ტიპების მიხედვით. ბიზნეს ანალიტიკის ეტაპებისა და ინსტრუმენტარიის დახასიათება. ბიზნეს ანალიტიკის პრაქტიკული გამოყენების სფეროების მიმოხილვა;

3. მონაცემთა ბაზების, საცავებისა და დიდი მონაცემების სისტემების პრაქტიკული გამოყენების ასპექტების გარჩევა. OLTP მონაცემთა ბაზების, მონაცემთა საცავების აგებისა და OLAP კუბების გამოყენების მაგალითების გარჩევა;

4. დიდი მონაცემების შენახვისა და დამუშავების სისტემების გამოყენების სპეციფიკის შესწავლა თანამედროვე საწარმოს მართვის თვალსაზრისით;

5. Hadoop-ისა და Mapreduce-ის პრაქტიკული გამოყენების (კერძოდ, Sales და Loan risk-ების გამოთვლის) მაგალითების გარჩევა რეალური Dataset-ებისათვის;

6. Hadoop-ის ეკოსისტემის კომპონენტების პრაქტიკული გამოყენების შემთხვევების მიმოხილვა;

7. დიდ მონაცემთა ანალიტიკის მეთოდებისა და ინსტრუმენტარიის (ხელოვნური ინტელექტის, ხელოვნური ნეირონული ქსელების, მანქანური სწავლების, ღრმა სწავლებისა და სხვ.) მოკლე მიმოხილვა სხვადასხვა სფეროში ბიზნეს ანალიტიკის დეველოპმენტისა და იმპლემენტაციისათვის გამოყენებაში;

8. მონაცემთა ანალიზის ბაზისური მეთოდებისა და ალგორითმების (კერძოდ, რეგრესული ანალიზის, ბაიესის დასკვნების, ფარული მარკოვის მოდელების, გადაწყვეტილებათა ხეების, K-means -სა და სხვ) გამოყენების მაგალითების (კლასტერიზაციისა და კლასიფიკაციის ამოცანების) განხილვა ბიზნეს ანალიტიკის სხვადასხვა სფეროს შემთხვევაში;

9. ხელოვნური ნეირონული ქსელების აგებისა და გამოყენების მაგალითების გარჩევა ბიზნეს ანალიტიკის ამოცანების მიხედვით;

10. მანქანური სწავლების (კერძოდ, კონვოლუციური, რეკურენტული ქსელების, აგრეთვე კოჰონენის თვითორგანიზებადი ქსელების და სხვ.) პრაქტიკული გამოყენების მიზანშეწონილობის შესწავლა;

11. მანქანური სწავლების ფრეიმვორკების Apache Mahout-ის გამოყენების . Apache Spark-ისა და Mapreduce-ის გამოყენების შედარებითი ასპექტების გარჩევა ბიზნეს ანალიტიკის სხვადასხვა ამოცანის შემთხვევაში;

12. ღრმა სწავლების ბიბლიოთეკებისა და არქიტექტურების მიმოხილვა და შედარებითი ანალიზი პრაქტიკული გამოყენების თვალსაზრისით. ტენზორებზე ოპერაციების შესწავლა. Tensorflow-ს გამოყენების მაგალითების გარჩევა;

13. რეალური სამყაროსათვის ღრმა სწავლების აპლიკაციების (კერძოდ, სახეთა ამოცნობის, აუდიოანალიზის, სამედიცინო დიაგნოსტიკის, ავტომატური მანქანური თარგმნისა და სხვ.) გამოყენების განხილვა;

14. ბიზნეს ანალიტიკის დეველოპმენტისა და იმპლემენტაციის პრაქტიკული რეკომენდაციების განხილვა გამოყენების სფეროების მიხედვით;

15. დიდი მონაცემების ღრუბლოვანი გამოთვლების მაგალითების განხილვა ბიზნეს ანალიტიკის დეველოპმენტის თვალსაზრისით.

პრაქტიკული სამუშაოს შესრულების ნიმუშები

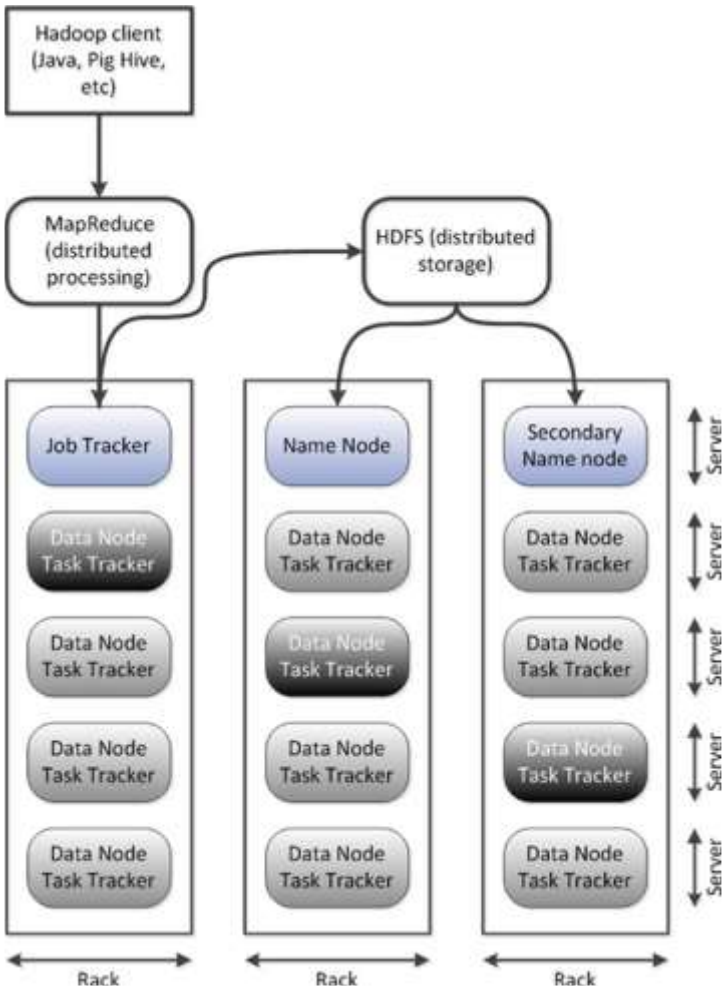
პრაქტიკული N 5

Hadoop-ისა და Mapreduce-ის პრაქტიკული გამოყენების მაგალითის გარჩევა რეალური Dataset-ებისათვის

თემა: გაყიდვების ანალიზი Hadoop & Mapreduce-ის გამოყენებით

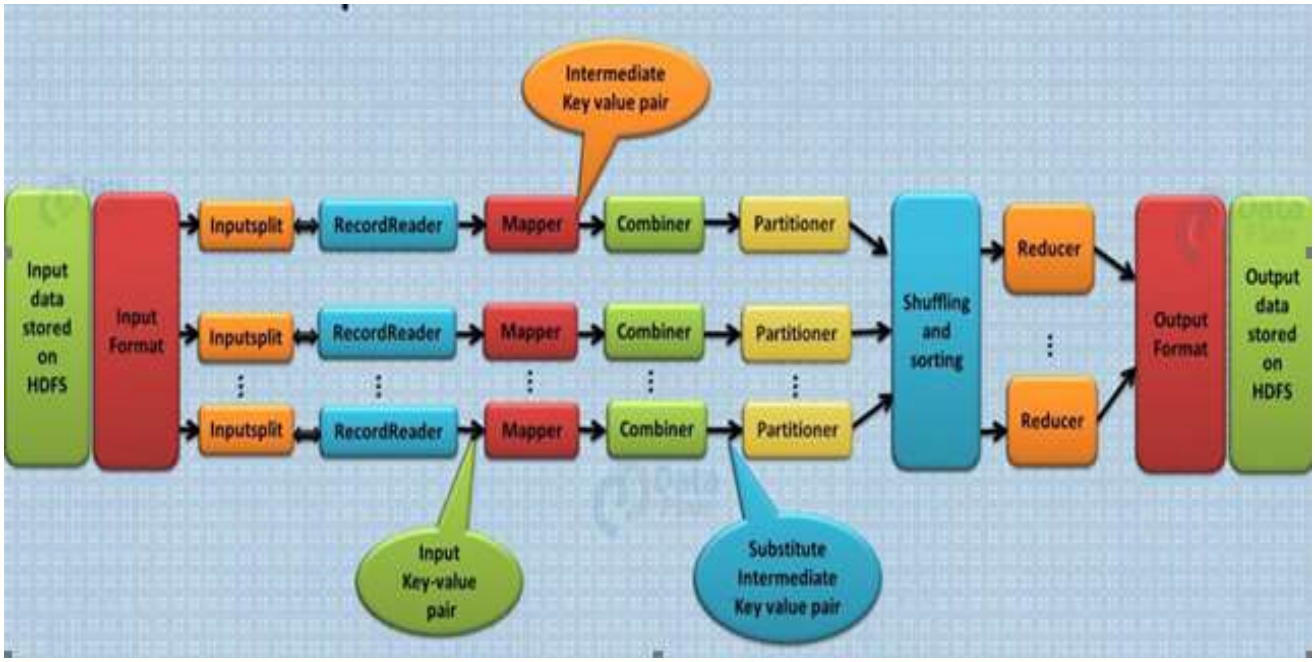
თანამედროვე კომერციულ ბიზნესში დიდი როლი ენიჭება ყიდვა-გაყიდვების მონაცემთა მენეჯმენტს მარკეტინგის დინამიკური ანალიზის, ბაზრის კონიუნქტურის როგორც მოკლევადიანი, ისე გრძელვადიანი ტრენდის პროგნოზირების, ბიზნეს პროცესების ეფექტური რეალიზაციის თვალსაზრისით დიდი მონაცემების ტექნოლოგიების, კერძოდ Hadoop & Mapreduce-ის გამოყენებით.

Hadoop-ის არქიტექტურა ნაჩვენებია 1-ელ ნახაზზე.



ნახ. 1

განვიხილოთ Hadoop MapReduce მუშაობის პროცესში MapReduce Job-ის შესრულების მონაცემთა ნაკადების ზოგადი სქემა (ნახ.2.).



ნახ. 2

ნაკადების ზოგადი სქემა შედგება შემდეგი ძირითადი კომპონენტებისგან:

1) Input Files. MapReduce ამოცანის მონაცემები ინახება შესატან ფაილებში, რომლებიც ჩვეულებრივ ინახება HDFS-ში. ამ ფაილების ფორმატი ნებისმიერია, თუმცა შესაძლებელია ორობითი ფორმატის გამოყენებაც;

2) **InputFormat**. სწორედ, InputFormat განსაზღვრავს, თუ როგორ მოხდება ამ შესატანი ფაილების დაყოფა და წაკითხვა. იგი ირჩევს ფაილებს ან სხვა ობიექტებს, რომლებიც გამოიყენება შეყვანისთვის;

3) InputSplits. ის იქმნება InputFormat-ის მიერ, ლოგიკურად წარმოადგენს იმ მონაცემებს, რომლებიც დამუშავდება ინდივიდუალური Mapper-ის მიერ. თითოეული map task იქმნება სათითაოდ ყოველი split-სათვის. ამრიგად, Map-ების ამოცანების რაოდენობა InputSplits-ის რაოდენობის ტოლი იქნება. ყოველი split იყოფა ჩანაწერებად და თითოეული ჩანაწერი დამუშავდება Map-ზე;

4) RecordReader. Hadoop MapReduce-ში RecordReader უკავშირდება InputSplit-ს და გარდაქმნის მონაცემებს „გასაღები-მნიშვნელობა“ წყვილში, რომელიც შესაფერისი ფორმატისაა mapper-ის მიერ წასაკითხად. RecordReader უსიტყვოდ (By default) იყენებს TextInputFormat-ს. RecordReader იმყოფება InputSplit-თან კავშირში, ვიდრე ფაილის კითხვა არ დასრულდება. იგი ანიჭებს ფაილში თითოეულ სტრიქონს ბაიტის ოფსეტს (უნიკალური ნომერი). გარდა ამისა, „გასაღები-მნიშვნელობა“ წყვილები იგზავნება mapper-ზე შემდგომი დამუშავებისთვის;

5) Mapper. ის ამუშავებს RecordReader-დან თითოეულ შეყვანის ჩანაწერს და წარმოქმნის ახალ „გასაღები-მნიშვნელობა“ წყვილს, ხოლო Mapper-ის მიერ წარმოქმნილი „გასაღები-მნიშვნელობა“ წყვილი სრულიად განსხვავდება შესაყვანი წყვილისგან. Mapper-ის გამომავალი ცნობილია, როგორც შუალედური გამომავალი, რომელიც იწერება ლოკალურ დისკზე. Mapper-ის გამომავალი პროდუქტი არ ინახება HDFS- ზე, რადგან ეს დროებითი მონაცემია. Mapper -ების გამოსავალი გადაეცემა combiner-ზე შემდგომი დამუშავებისათვის.

6) **Combiner.** კომბაინერები ასევე ცნობილია, როგორც მინი-რედუქტორი (Mini-reducer) ანუ „დამადაბლებელი“. Hadoop MapReduce Combiner ასრულებს Mapper-ების გამოსავალზე ლოკალურ აგრეგაციას, რაც უზრუნველყოფს მონაცემთა გადაცემის მინიმიზებას mapper-სა და reducer-ს შორის. კომბაინერის ფუნქციონირების დასრულების შემდეგ, გამოსავალი შემდეგ გაეცემა დამანაწევრებელზე (partitioner) შემდგომი მუშაობისთვის;

7) **Partitioner.** Hadoop MapReduce-ში Partitioner კომბაინერებიდან იღებს გამოსასვლელს და ასრულებს დანაწევრებას გასაღების საფუძველზე და შემდომი სორტირებისათვის. ჰეშ-ფუნქციის საფუძველზე, გასაღები (ან გასაღების ქვესიმრავლე) გამოიყენება დანაყოფის (partition) მისაღებად. MapReduce-ში, გასაღებური მნიშვნელობის მიხედვით, ხდება თითოეული კომბაინერის გამოსავალის დანაწევრება. ხოლო ჩანაწერი, რომელსაც აქვს იგივე გასაღებური მნიშვნელობა, გადადის იმავე დანაყოფში, შემდეგ კი თითოეული დანაყოფი იგზავნება reducer-საკენ. ამგვარად, დანაწევრება reducer-ის მეშვეობით map-ის გამოსასვლელის განაწილების საშუალებას იძლევა;

8) **Shuffling and Sorting.** ამჯერად, კვანძების შემცირების მიზნით, slave კვანძთან ხდება გამოსასვლელის გადალაგება (Shuffling), რაც არის, ფაქტობრივად, მონაცემების ფიზიკური გადაადგილება, რომელიც კეთდება ქსელში. მას შემდეგ, რაც ყველა mapper დაასრულებს თავის სამუშაოს, ხოლო მათი გამოსასვლელების გადალაგება მოხდება reducer კვანძებზე, მაშინ მიღებული შუალედური გამოსასვლელები შერწყმული და სორტირებული იქნება, რაც შემდგომ უკვე განიხილება როგორც reduce ფაზის შესასვლელი;

9) **Reducer**. იგი იღებს mapper-ების მიერ წარმოქმნილ შუალედური „გასაღები-მნიშვნელობა“ წყვილების სიმრავლეს და შემდეგ თითოეულ მათგანზე ასრულებს reducer ფუნქციას, რათა გამოიმუშავოს გამოსასვლელი. Reducer-ის გამოსასვლელი კი იქნება საბოლოო გამოსასვლელი, რომელიც ინახება HDFS-ში;

10) **RecordWriter.** იგი ჩაიწერს ამ გამოსასვლელ „გასაღები-მნიშვნელობის“ წყვილებს Reducer ფაზიდან გამოსასვლელ ფაილებამდე;

11) **OutputFormat.** ეს არის „გასაღები-მნიშვნელობის“ წყვილების გამოსასვლელი ფორმა, რომელიც ჩაიწერება გამოსასვლელ ფაილებში RecordWriter-ის მეშვეობით. იგი განისაზღვრება OutputFormat-ით. ამრიგად, reducer-ის საბოლოო გამოსასვლელი შედეგები HDFS-ზე იწერება OutputFormat ინსტანციების საშუალებით.

განვიხილოთ გაყიდვების ანალიზის ამოცანა Hadoop & Mapreduce-ის გამოყენებით, სადაც გაყიდვები შეიცავს შემდეგ ინფორმაციას: პროდუქციის დასახელება, ფასი, გადახდის სახეობა, ქალაქი, კლიენტის ქვეყანა და ა.შ.

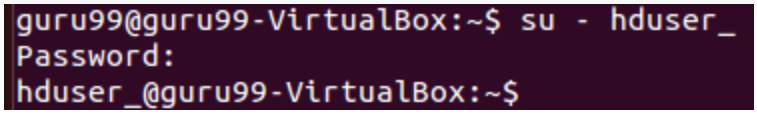
საწყისი მონაცემებად აღებულია SalesJan2009.csv -დან (იხილე დანართში, რომლის ფრაგმენტი მოყვანილია ქვემოთ ცხრილის სახით) [sales-jan-2009.png (942×370) (guru99.com)].

ამოცანის მიზანია: ვიპოვოთ თითოეულ ქვეყანაში გაყიდული პროდუქციის რაოდენობა.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Transaction_date	Product	Price	Payment_	Name	City	State	Country	Account_Created	Last_Login	Latitude	Longitude
2	01-02-2009 06:17	Product1	1200	Mastercar	carolina	Basildon	England	United Kir	01-02-2009 06:00	01-02-2009 06:08	51.5	-1.11667
3	01-02-2009 04:53	Product1	1200	Visa	Betina	Parkville	MO	United Sta	01-02-2009 04:42	01-02-2009 07:49	39.195	-94.6819
4	01-02-2009 13:08	Product1	1200	Mastercar	Federica e	Astoria	OR	United Sta	01-01-2009 16:21	01-03-2009 12:32	46.18806	-123.83
5	01-03-2009 14:44	Product1	1200	Visa	Gouya	Echuca	Victoria	Australia	9/25/05 21:13	01-03-2009 14:22	-36.1333	144.75
6	01-04-2009 12:56	Product2	3600	Visa	Gerd W	Cahaba He	AL	United Sta	11/15/08 15:47	01-04-2009 12:45	33.52056	-86.8025
7	01-04-2009 13:19	Product1	1200	Visa	LAURENCE	Mickleton	NJ	United Sta	9/24/08 15:19	01-04-2009 13:04	39.79	-75.2381
8	01-04-2009 20:11	Product1	1200	Mastercar	Fleur	Peoria	IL	United Sta	01-03-2009 09:38	01-04-2009 19:45	40.69361	-89.5889
9	01-02-2009 20:09	Product1	1200	Mastercar	adam	Martin	TN	United Sta	01-02-2009 17:43	01-04-2009 20:01	36.34333	-88.8503
10	01-04-2009 13:17	Product1	1200	Mastercar	Renee Elis	Tel Aviv	Tel Aviv	Israel	01-04-2009 13:03	01-04-2009 22:10	32.06667	34.76667
11	01-04-2009 14:11	Product1	1200	Visa	Aidan	Chatou	Ile-de-Fra	France	06-03-2008 04:22	01-05-2009 01:17	48.88333	2.15
12	01-05-2009 02:42	Product1	1200	Diners	Stacy	New York	NY	United Sta	01-05-2009 02:23	01-05-2009 04:59	40.71417	-74.0064
13	01-05-2009 05:39	Product1	1200	Amex	Heidi	Eindhover	Noord-Bra	Netherlan	01-05-2009 04:55	01-05-2009 08:15	51.45	5.466667
14	01-02-2009 09:16	Product1	1200	Mastercar	Sean	Shavano P	TX	United Sta	01-02-2009 08:32	01-05-2009 09:05	29.42389	-98.4933
15	01-05-2009 10:08	Product1	1200	Visa	Georgia	Eagle	ID	United Sta	11-11-2008 15:53	01-05-2009 10:05	43.69556	-116.353
16	01-02-2009 14:18	Product1	1200	Visa	Richard	Riverside	NJ	United Sta	12-09-2008 12:07	01-05-2009 11:01	40.03222	-74.9578
17	01-04-2009 01:05	Product1	1200	Diners	Leanne	Julianstov	Meath	Ireland	01-04-2009 00:00	01-05-2009 13:36	53.67722	-6.31917

თავდაპირველად უნდა დავრწმუნდეთ, რომ Hadoop და ინსტალირებულია. დასაწყისში მომხმარებელი შევცვალოთ 'hduser' -ით (userid გამოვიყენოთ Hadoop config განმავლობაში).

```
su - hduser_
```



ბიჯი 1)

შევქმნათ ახალი დირექტორია სახელწოდებით **MapReduceTutorial**

```
sudo mkdir MapReduceTutorial
```

მივანიჭოთ ნებართვები:

```
sudo chmod -R 777 MapReduceTutorial
```

SalesMapper.java

```
package SalesCountry;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class SalesMapper extends MapReduceBase implements Mapper <LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);

    public void map(LongWritable key, Text value, OutputCollector <Text, IntWritable> output, Reporter reporter) throws IOException {

        String valueString = value.toString();
        String[] SingleCountryData = valueString.split(",");
        output.collect(new Text(SingleCountryData[7]), one);
    }
}
```

SalesCountryReducer.java

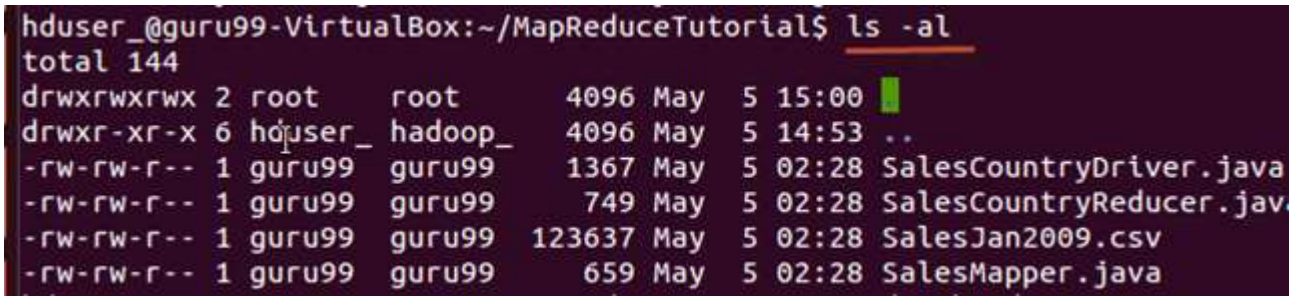
```
package SalesCountry;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class SalesCountryReducer extends MapReduceBase implements Reducer<Text, IntWritable,
Text, IntWritable> {

    public void reduce(Text t_key, Iterator<IntWritable> values, OutputCollector<Text,IntWritable>
output, Reporter reporter) throws IOException {
            Text key = t_key;
            int frequencyForCountry = 0;
            while (values.hasNext()) {
                    // replace type of value with the actual type of our value
                    IntWritable value = (IntWritable) values.next();
                    frequencyForCountry += value.get();

            }
            output.collect(key, new IntWritable(frequencyForCountry));
    }
}
```

SalesCountryDriver.java

```
package SalesCountry;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class SalesCountryDriver {
   public static void main(String[] args) {
      JobClient my_client = new JobClient();
      // Create a configuration object for the job
      JobConf job_conf = new JobConf(SalesCountryDriver.class);

      // Set a name of the Job
      job_conf.setJobName("SalePerCountry");

      // Specify data type of output key and value
      job_conf.setOutputKeyClass(Text.class);
      job_conf.setOutputValueClass(IntWritable.class);
```

```
        // Specify names of Mapper and Reducer Class
        job_conf.setMapperClass(SalesCountry.SalesMapper.class);
        job_conf.setReducerClass(SalesCountry.SalesCountryReducer.class);

        // Specify formats of the data type of Input and output
        job_conf.setInputFormat(TextInputFormat.class);
        job_conf.setOutputFormat(TextOutputFormat.class);

        // Set input and output directories using command line arguments,
        //arg[0] = name of input directory on HDFS, and arg[1] =  name of output directory to be created
        // to store the output file.

        FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));

        my_client.setConf(job_conf);
        try {
           // Run the job
           JobClient.runJob(job_conf);
        } catch (Exception e) {
           e.printStackTrace();
        }
     }
}
```

შევამოწმოთ ნებართვები ყველა ამ ფაილებისათვის



და თუ 'read' ნებართვები გამოტოვებულია, მივანიჭოთ

```
hduser_@guru99-VirtualBox:~/MapReduceTutorial$ sudo chmod +r *.*
```

ბიჯი 2)

Export classpath

export CLASSPATH="$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.2.0.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-common-2.2.0.jar:$HADOOP_HOME/share/hadoop/common/hadoop-common-2.2.0.jar:~/MapReduceTutorial/SalesCountry/*:$HADOOP_HOME/lib/*"

```
hduser_@guru99-VirtualBox:~/MapReduceTutorial$ export CLASSPATH="$HADOOP_HOME/share/hadoop/mapreduce/hadoo
p-mapreduce-client-core-2.2.0.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-common-2.2.0
.jar:$HADOOP_HOME/share/hadoop/common/hadoop-common-2.2.0.jar:~/MapReduceTutorial/SalesCountry/*:$HADOOP_H
OME/lib/*"
hduser_@guru99-VirtualBox:~/MapReduceTutorial$
```

ბიჯი 3)

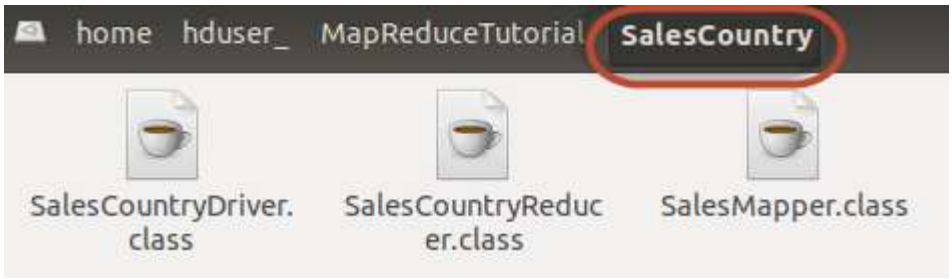
მოვახდინოთ Java ფაილების კომპილაცია (ეს ფაილები წარმოდგენილია **Final-MapReduceHandsOn** დირექტორიაში). მათი კლასის ფაილები შეიტანება package დირექტორიაში

javac -d . SalesMapper.java SalesCountryReducer.java SalesCountryDriver.java

```
hduser_@guru99-VirtualBox:~/MapReduceTutorial$ javac -d . SalesMapper.java SalesCountryReducer.java SalesC
ountryDriver.java
/home/guru99/Downloads/hadoop/share/hadoop/common/hadoop-common-2.2.0.jar(org/apache/hadoop/fs/Path.class)
: warning: Cannot find annotation method 'value()' in type 'LimitedPrivate': class file for org.apache.had
oop.classification.InterfaceAudience not found
1 warning
hduser_@guru99-VirtualBox:~/MapReduceTutorial$
```

ეს გაფრთხილება შეიძლება უსაფრთხოდ იქნას იგნორირებული.

კომპილაცია ქმნის დირექტორიას მიმდინარე დირექტორიაში java საწყისი ფაილის სახელწოდების პაკეტთან (**SalesCountry** ჩვენს შემთხვევაში) და შეაქვს მასში ყველა კომპილირებული ფაილი.



ბიჯი 4)

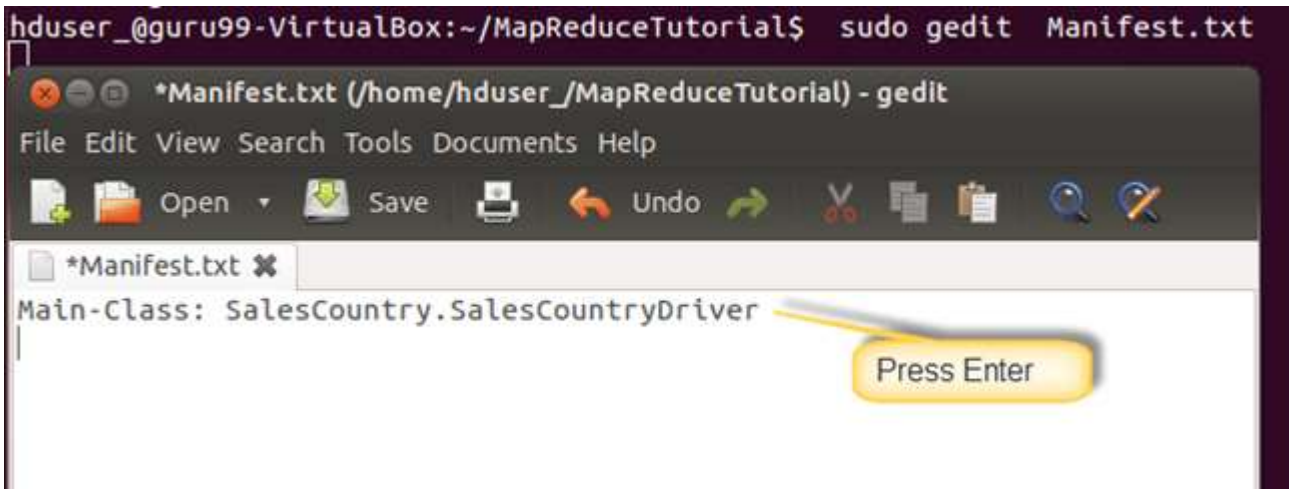
შევქმნათ ახალი ფაილი**Manifest.txt**

sudo gedit Manifest.txt

დავამატოთ მასში შემდეგი სტრიქონები,

Main-Class: SalesCountry.SalesCountryDriver

Click ღილაკზე Enter.



SalesCountry.SalesCountryDriver არის მთავარი კლასის სახელი.

ბიჯი 5)

შევქნათ Jar ფაილი:

```
jar cfm ProductSalePerCountry.jar Manifest.txt SalesCountry/*.class
```

შევამოწმოთ jar ფაილი შექმნილია თუ არა.

ბიჯი 6)

გავუშვათ Hadoop

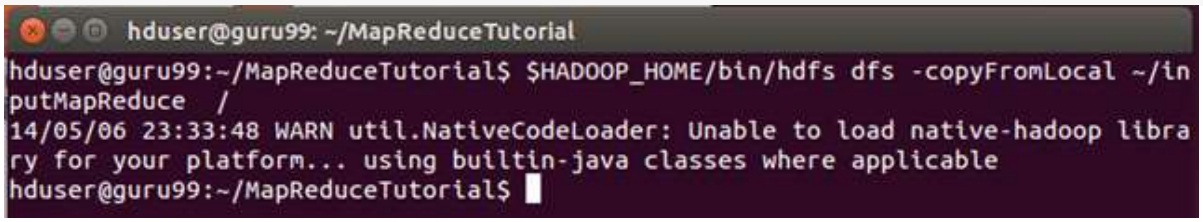
```
$HADOOP_HOME/sbin/start-dfs.sh
$HADOOP_HOME/sbin/start-yarn.sh
```

ბიჯი 7)

ვაკოპირებთ **SalesJan2009.csv** ფაილს **~/inputMapReduce** -ში.

ახლა ვიყენებთ **~/inputMapReduce** ბრძანებას HDFS-ში კოპირებისათვის:

```
$HADOOP_HOME/bin/hdfs dfs -copyFromLocal ~/inputMapReduce /
```



ეს გაფრთხილება შეიძლება უსაფრთხოდ იქნას იგნორირებული.

შევამოწმოთ არის თუ არა ფაილი კოპირებული.

```
$HADOOP_HOME/bin/hdfs dfs -ls /inputMapReduce
```

```
hduser@guru99:~/MapReduceTutorial$ $HADOOP_HOME/bin/hdfs dfs -ls /inputMapReduce
14/05/06 23:35:54 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Found 1 items
-rw-r--r--   1 hduser supergroup     123637 2014-05-06 23:33 /inputMapReduce/Sal
esJan2009.csv
hduser@guru99:~/MapReduceTutorial$
```

ბიჯი 8)

გავუშვათ შესრულებაზე MapReduce job:

```
$HADOOP_HOME/bin/hadoop jar ProductSalePerCountry.jar /inputMapReduce
/mapreduce_output_sales
```

```
hduser@guru99: ~/MapReduceTutorial
hduser@guru99:~/MapReduceTutorial$ $HADOOP_HOME/bin/hadoop jar ProductSalePerCou
ntry.jar /inputMapReduce /mapreduce_output_sales
```

ეს HDFS-ზე შექმნის გამოტანის დირექტორიას mapreduce_output_sales სახელწოდებით, რომელიც შეიცავს გაყიდულპროდუქციას თვითოეული ქვეყნისათვის.

ბიჯი 9)

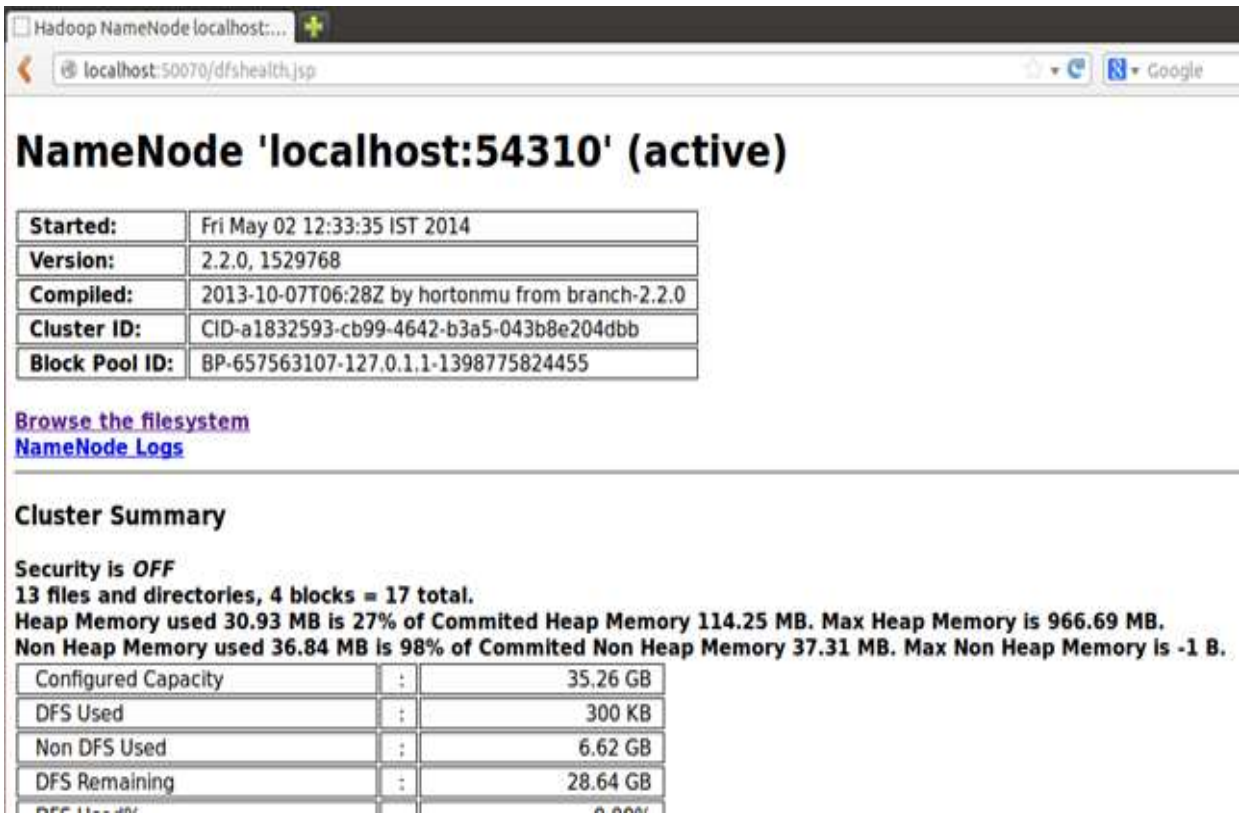
შედეგების ნახვა შესაძლებელია ინტერფეისის ბრძანების მეშვეობით:

```
$HADOOP_HOME/bin/hdfs dfs -cat /mapreduce_output_sales/part-00000
```

```
hduser@guru99: ~/MapReduceTutorial
hduser@guru99:~/MapReduceTutorial$ $HADOOP_HOME/bin/hdfs dfs -cat /mapreduce_out
put_sales/part-00000
14/05/02 13:03:46 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Argentina       1
Australia       38
Austria 7
Bahrain 1
Belgium 8
Bermuda 1
Brazil  5
Bulgaria        1
CO      1
Canada  76
Cayman Isls     1
```

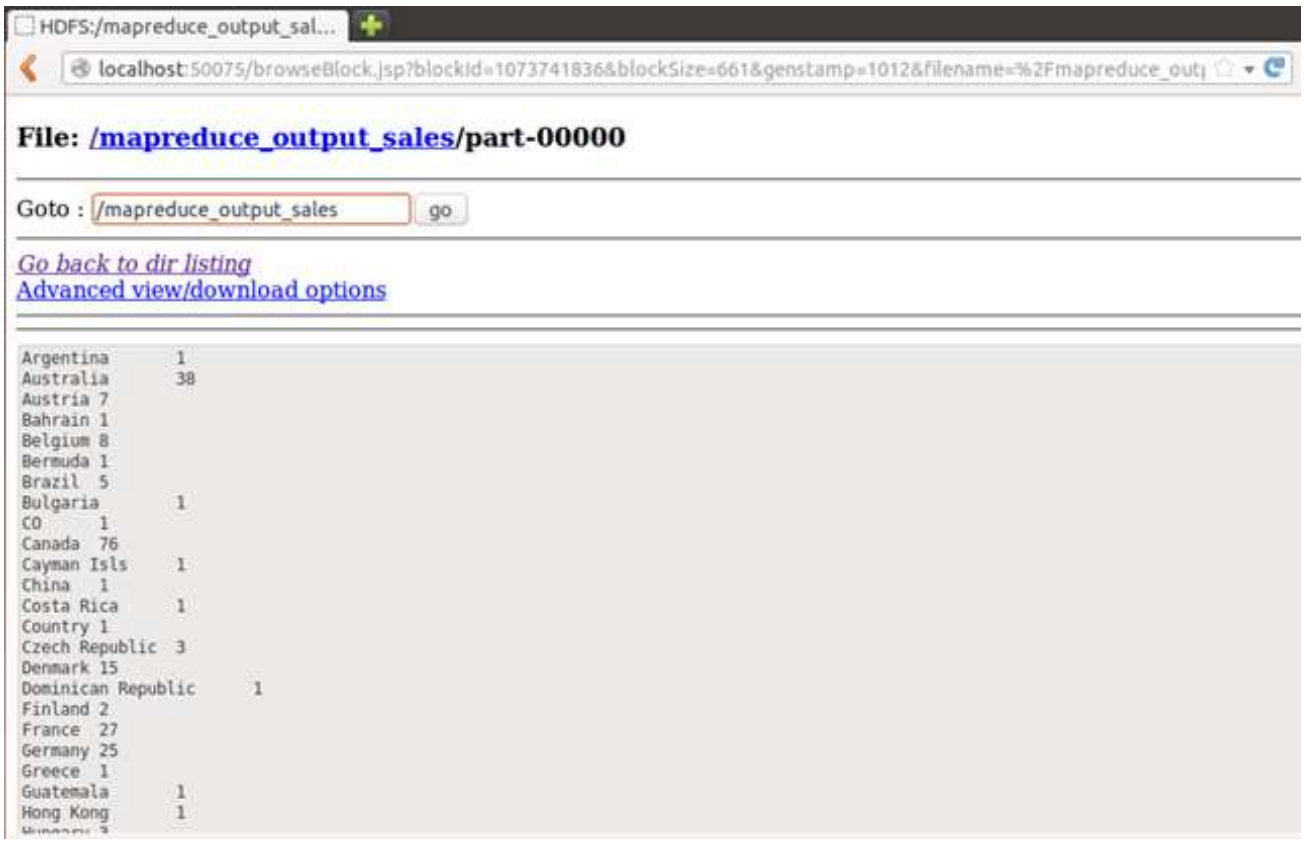
შედეგების ნახვა შესაძლებელია აგრეთვე web ინტერფეისის მეშვეობითაც. ამისათვის,

web browser -ში გავხნათ r.



ვირჩევთ **'Browse the filesystem'** და გადავდივართ **/mapreduce_output_sales** ში:

გავხსნათ **part-r-00000**



Java კოდების გარჩევა

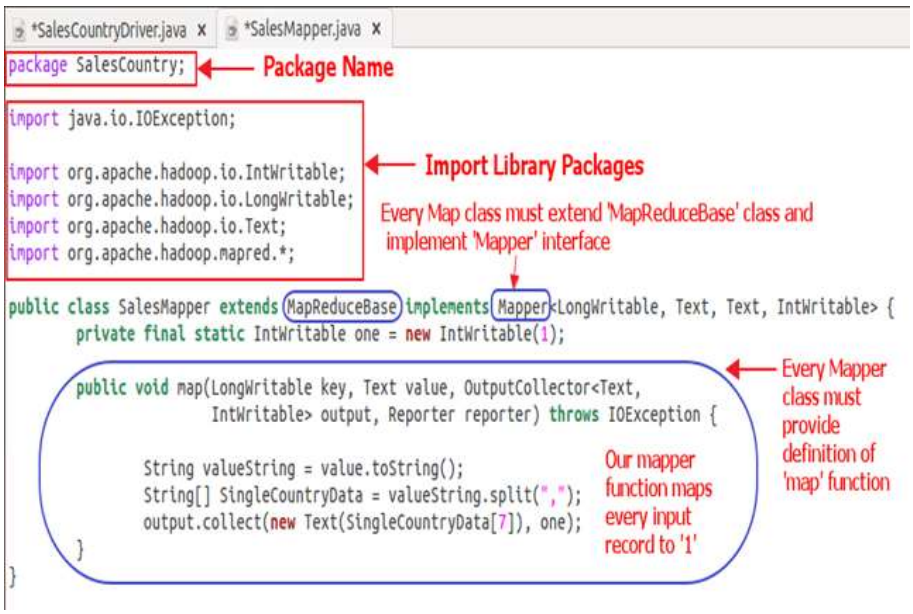
SalesMapper.java

```
package SalesCountry;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;
public class SalesMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text,
IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output,
Reporter reporter) throws IOException {
                String valueString = value.toString();
                String[] SingleCountryData = valueString.split(",");
                output.collect(new Text(SingleCountryData[7]), one);
        }
}
```

SalesMapper კლასის განმარტება

1. დავიწყოთ ჩვენი კლასის პაკეტის სახელის განსაზღვრით, ანუ გვექნება **SalesCountry**. კომპილაციის გამოსავალზე **SalesMapper.class** შევა დირექტორიაში **SalesCountry** პაკეტის სახელწოდებით. აქედან გამომდინარე, ჩვენ შემოგვაქვს პაკეტების ბიბლიოთეკა.

ქვემოთ მოყვანილია **SalesMapper** კლასის მაგალითი:



1. **SalesMapper** კლასის განსაზღვრა:

public class SalesMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {

ყოველი mapper კლასი გამომდინარეობს **MapReduceBase** კლასიდან და აისახება **Mapper** ინტერფეისში.

2. '**map**' ფუნქციის განსაზღვრა

```
public void map(LongWritable key,
        Text value,
OutputCollector<Text, IntWritable> output,
         Reporter reporter) throws IOException
```

Mapper კლასის მთავარი ნაწილი არის '**map()**' მეთოდი, რომლისთვისაც მისაღებია ოთხი არგუმენტი.

'**map()**' მეთოდი იძახებს **key-value** წყვილს ('**key**' და '**value**' კოდში).

'**map()**' მეთოდი იწყებს შემავალი ტექსტის დაყოფას, სტრიქონების სიტყვებად (ლექსემებად) დანაწევრებას.

```
String valueString = value.toString();
String[] SingleCountryData = valueString.split(",");
```

აქ, '**,**' ნიშნავს გამყოფს.

ამის შემდეგ, წყვილი ფორმირდება '**SingleCountryData**' მასივის მე-7 ინდექსის მქონე ჩანაწერის გამოყენებით და მნიშვნელობით '**1**'.

```
output.collect(new Text(SingleCountryData[7]), one);
```

ჩვენ შევარჩიეთ ჩანაწერი მე-7 ინდექსით, რადგან მონაცემები **Country** არის განთავსებული '**SingleCountryData**' მასივის მე-7 ინდექსით. ამასთან, საწყისი ინდექსი არის 0:

Transaction_date,Product,Price,Payment_Type,Name,City,State,**Country**,Account_Created,Last_Login,Latitude,Longitude

mapper გამოსავალზე გვაქვს კვლავ **key-value** წყვილი, რომლის გამოსატანად გამოიყენება '**OutputCollector**'-ის '**collect()**' მეთოდი.

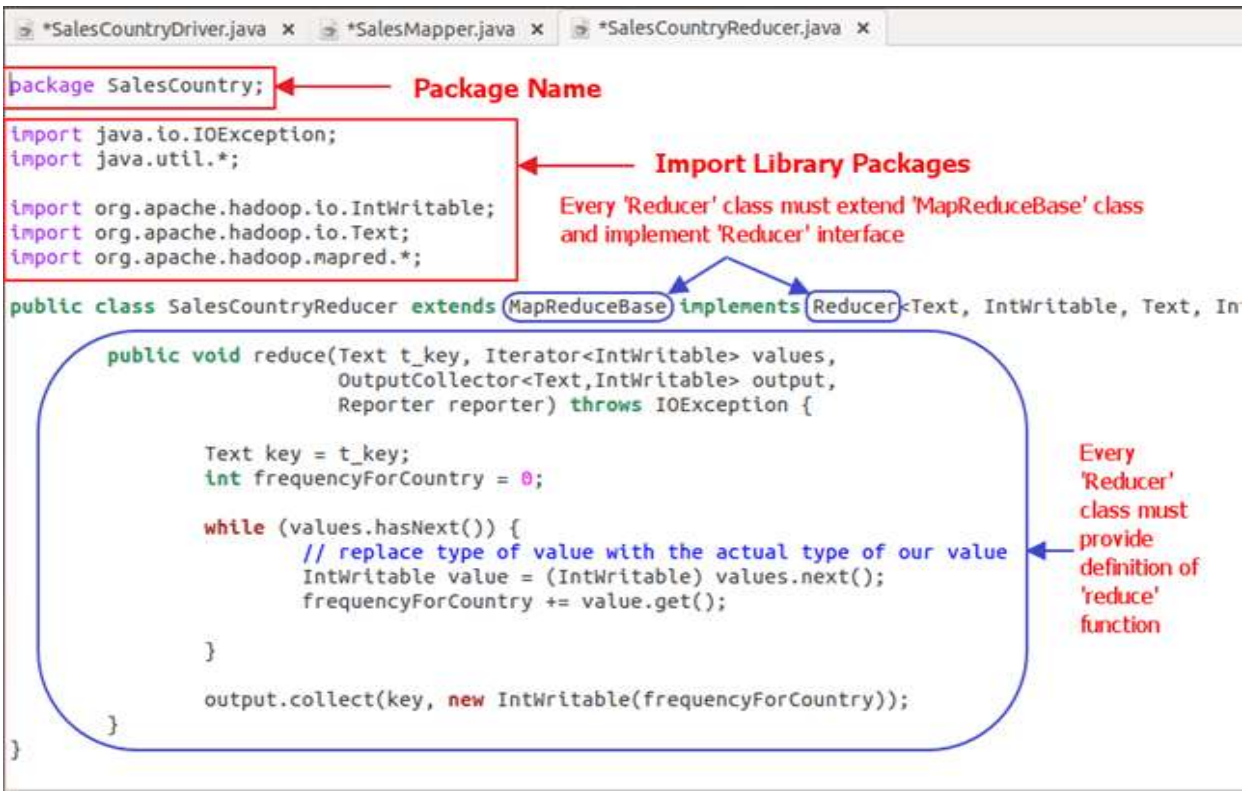
SalesCountryReducer.java

```
package SalesCountry;
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;
public class SalesCountryReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text,
IntWritable> {
        public void reduce(Text t_key, Iterator<IntWritable> values, OutputCollector<Text,IntWritable>
output, Reporter reporter) throws IOException {
                Text key = t_key;
                int frequencyForCountry = 0;
                while (values.hasNext()) {
                        // replace type of value with the actual type of our value
                        IntWritable value = (IntWritable) values.next();
                        frequencyForCountry += value.get();
                }
                output.collect(key, new IntWritable(frequencyForCountry));
        }
}
```

SalesCountryReducer კლასის განმარტება

1. **SalesCountry** არის გამომავალი პაკეტის სახელი. კომპილაციის გამოსავალზე **SalesCountryReducer.class** შევა დირექტორიაში **SalesCountry** პაკეტის სახელწოდებით. აქედან გამომდინარე, ჩვენ შემოგვაქვს პაკეტების ბიბლიოთეკა.

ქვემოთ მოყვანილია **SalesCountryReducer** კლასი.



1. SalesCountryReducer კლასის განსაზღვრა:

public class SalesCountryReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {

აქ, პირველი ორი მონაცემის ტიპი '**Text**' და '**IntWritable**' არის reducer-ში შემავალი key-value მონაცემთა ტიპები.

mapper-ის გამოსავალი ფორმატი იქნება <CountryName1, 1>, <CountryName2, 1>. mapper-ის გამოსავალი გარდაიქმნება reducer-ის შესავლად. შესაბამისად, **Text** და **IntWritable** აქ გამოიყენება მონაცემთა ტიპებად. ბოლო ორი მონაცემთა ტიპი 'Text' და 'IntWritable' არის reducer-ის მიერ გამოსავალზე გენერირებული key-value წყვილის მონაცემთა ტიპები.

ყოველი reducer კლასი გამომდინარეობს **MapReduceBase** კლასიდან და აისახება **Reducer** ინტერფეისზე.

2. 'reduce' ფუნქციის განსაზღვრა:

```
public void reduce( Text t_key,
        Iterator<IntWritable> values,
        OutputCollector<Text,IntWritable> output,
        Reporter reporter) throws IOException {
```

reduce() მეთოდის შესავალზე გასაღები იქნება მრავლობითი მნიშვნელობების სიასთან ერთად. მაგალითად, ჩვენს შემთხვევაში იქნება:

<United Arab Emirates, 1>, <United Arab Emirates, 1>, <United Arab Emirates, 1>,<United Arab Emirates, 1>, <United Arab Emirates, 1>, <United Arab Emirates, 1>.

მოცემული reducer როგორც: **<United Arab Emirates, {1,1,1,1,1,1}>**

არგუმენტების ასეთი ფორმის მიღების შემთხვევაში, პირველი ორი მონაცემთა ტიპი გვექნება **Text** და **Iterator<IntWritable>**. **Text** არის გასაღების ტიპი,
ხოლო **Iterator<IntWritable>** მონაცემთა ტიპები იქნება მრავლობითი მნიშვნელობების სიისათვის.

OutputCollector<Text,IntWritable> ტიპისათვის გვექნება შემდეგი არგუმენტი, რომელიც აგროვებს reducer ფაზის გამოსავლებს.

reduce() მეთოდი იწყებს გასაღების მნიშვნელობის კოპირებას და სიხშირის ინიციალიზაციას 0-დან.

```
Text key = t_key;
int frequencyForCountry = 0;
```

მაშინ '**while**' ციკლის გამოყენებით, გამოვითვლით მნიშვნელობათა ჯამურ სიხშირეს.

```
while (values.hasNext()) {
    // replace type of value with the actual type of our value
    IntWritable value = (IntWritable) values.next();
    frequencyForCountry += value.get();
}
```

ახლა, უკვე შედეგები გამოგვაქვს კოლექტორის გამოსავალზე **key** სახით და ვიღებთ **frequency count**. ქვემოთ მოყვანილია შემდეგი კოდი:

```
output.collect(key, new IntWritable(frequencyForCountry));
```

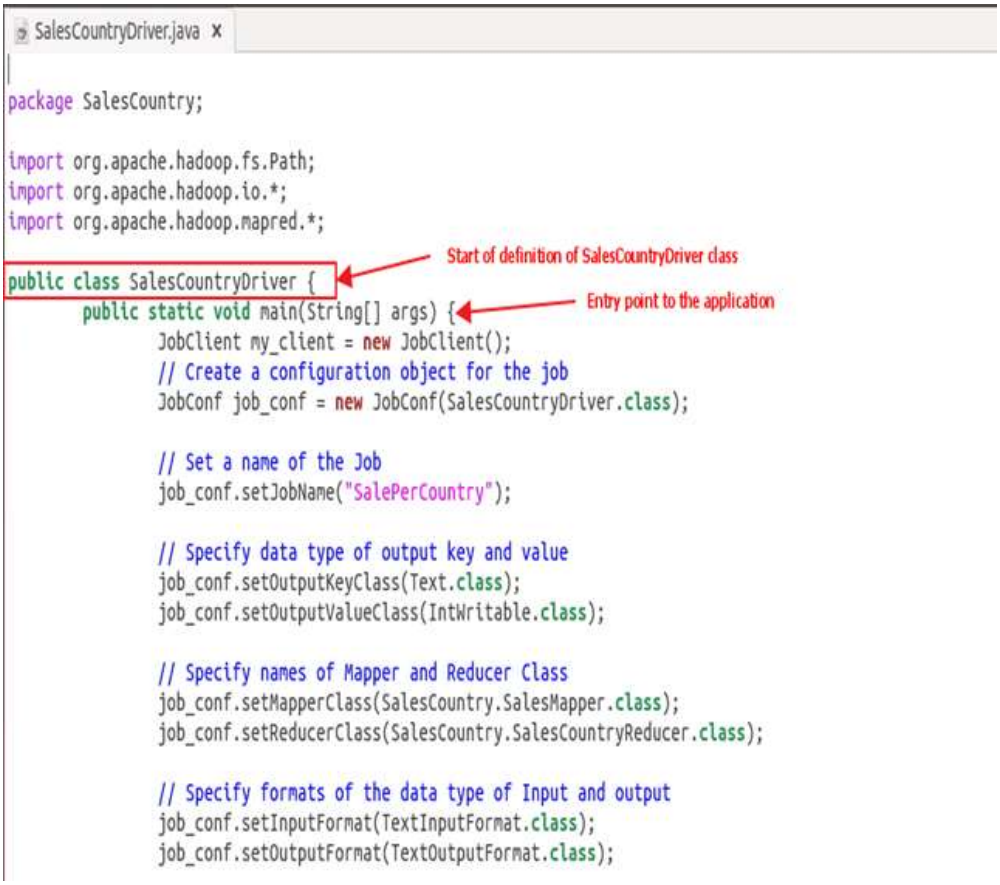
SalesCountryDriver.java

```
package SalesCountry;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
public class SalesCountryDriver {
	public static void main(String[] args) {
		JobClient my_client = new JobClient();
		// Create a configuration object for the job
		JobConf job_conf = new JobConf(SalesCountryDriver.class);

		// Set a name of the Job
		job_conf.setJobName("SalePerCountry");

		// Specify data type of output key and value
		job_conf.setOutputKeyClass(Text.class);
		job_conf.setOutputValueClass(IntWritable.class);

		// Specify names of Mapper and Reducer Class
		job_conf.setMapperClass(SalesCountry.SalesMapper.class);
		job_conf.setReducerClass(SalesCountry.SalesCountryReducer.class);

		// Specify formats of the data type of Input and output
		job_conf.setInputFormat(TextInputFormat.class);
		job_conf.setOutputFormat(TextOutputFormat.class);

		// Set input and output directories using command line arguments,
		//arg[0] = name of input directory on HDFS, and arg[1] =  name of output directory to be
created to store the output file.

		FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
		FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));
		my_client.setConf(job_conf);
		try {
			// Run the job
			JobClient.runJob(job_conf);
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

SalesCountryDriver კლასის განმარტება

1. **SalesCountry** არის გამომავალი პაკეტის სახელი. კომპილაციის გამოსავალზე **SalesCountryDriver.class** შევა დირექტორიაში **SalesCountry** პაკეტის სახელწოდებით. აქედან გამომდინარე, ჩვენ შემოგვაქვს პაკეტების ბიბლიოთეკა.

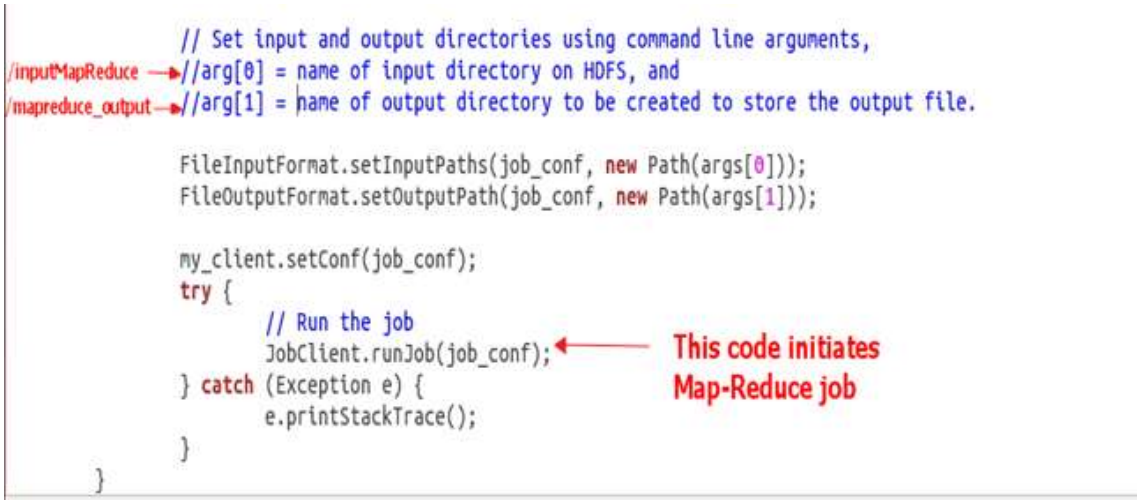
2. განვსაზღვროთ driver კლასი, რომელიც ქმნის ახალ client job-ს, ახდენს ობიექტის კონფიგურირებას და აცხადებს Mapper და Reducer კლასებს. driver კლასი ახდენს ჩვენი MapReduce job-ის გაწყობას Hadoop-ში შესრულებაზე გაშვებისათვის. ამ კლასში ჩვენ განვსაზღვრავთ **job name, data type of input/output and names of mapper and reducer classes**.



3. ქვემოთ კოდის მონაკვეთში, ჩვენ განთავსებულია შეტანისა და გამოტანის დირექტორიები, რომლებიც იყენებენ მონაცემთა სიმრავლეებს და შესაბამისად ქმნიან გამოსავალს.

arg[0] და **arg[1]** არის MapReduce შეტანა-გამოტანის არგუმენტები.

$HADOOP_HOME/bin/hadoop jar ProductSalePerCountry.jar /inputMapReduce /mapreduce_output_sales



4. ჩვენი job-ის ტრიგერი

ქვემოთ მოყვანილი კოდი იწყებს MapReduce job-ის შესრულებას:

```
try {
  // Run the job
  JobClient.runJob(job_conf);
} catch (Exception e) {
  e.printStackTrace();
}
```

პრაქტიკული N3

მონაცემთა ბაზების მართვა. OLTP. ETL. მონაცემთა საცავები. OLAP კუბები

თემა: ინფორმაციული ტექნოლოგიები უმაღლეს განათლებაში

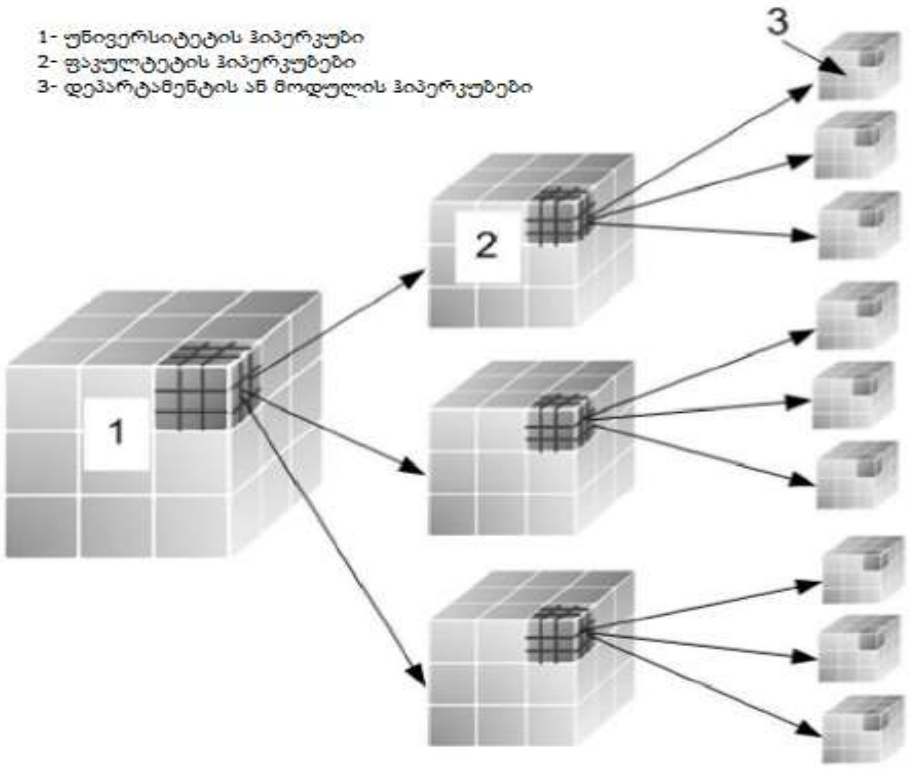
უმაღლესი განათლების სისტემაში, ინფორმაციული ტექნოლოგიების გამოყენების თვალსაზრისით, მართვის ამოცანები შეიძლება განვიხილოთ სამდონიან იერარქიაში: უნივერსიტეტი > ფაკულტეტი > დეპარტამენტი.

- *უნივერსიტეტის დონეზე:*

- ერთიანი საგამოცდო ცენტრის ორგანიზება და ფუნქციონირება საუნივერსიტეტო სერვერული კლასტერების ბაზაზე;

- დისტანციური სწავლების ცენტრის ორგანიზება დიდი მონაცემებისა და ღრუბლოვანი ტექნოლოგიების გამოყენებით;

- უნივერსიტეტის სერვერზე OLAP ტექნოლოგიის იერარქიული კუბის გამოყენებით (ნახ.1) შიდასაუნივერსიტეტო ანუ ფაკულტეტთაშორისი ყოველწლიური რეიტინგული კლასიფიკაცია (Ranking), რომლის შედეგად შეიძლება განისაზღვროს ფაკულტეტების სასწავლო მაჩვენებლები.



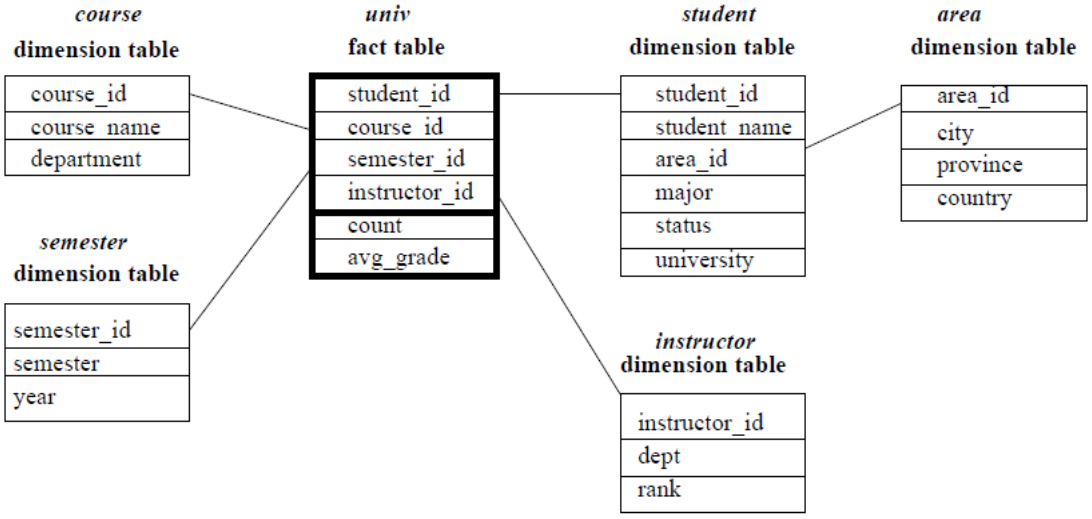
ნახ. 1

➢ *ფაკულტეტის და დეპარტამენტის დონეზე:*

- შრომის ბაზარზე ცალკეული სპეციალობისადმი მოთხოვნების მიხედვით კურიკულუმებისა თუ სილაბუსების ადაპტაციის განხორციელება;

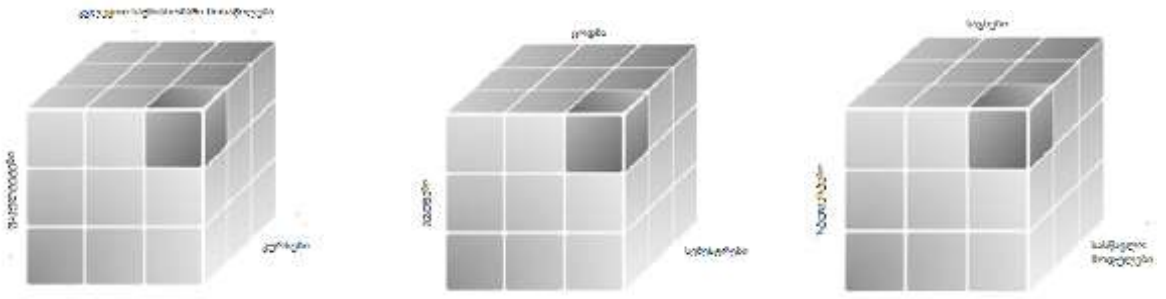
- ვირტუალური ლექციებისა და ლაბორატორიების შემუშავება;

- ხარისხის მართვის თვალსაზრისით, საფაკულტეტო სერვერზე Data Warehouse ტექნოლოგიის (ვარსკვლავისებრი ან ფიფქისებრი სქემით) გამოყენება (ნახ.2).



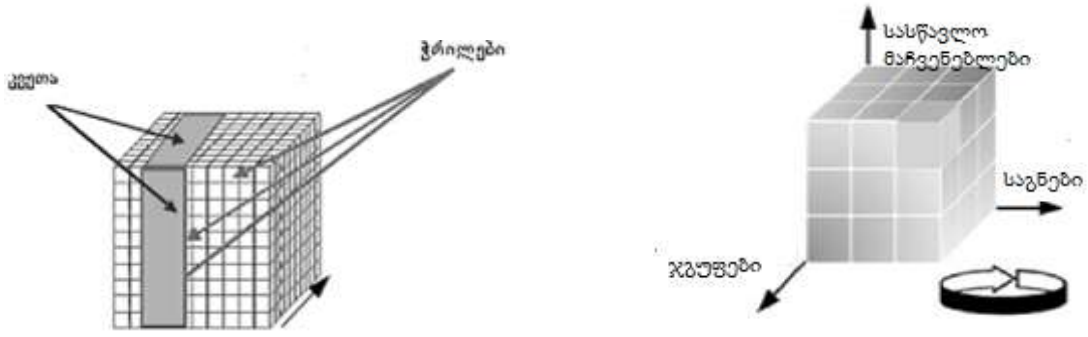
ნახ. 2

OLAP ჰიპერკუბებისათვის (მაგალითად, ნახ.3-ა,ბ,გ), კერძოდ, სხვადასხვა ფაქტორის მიხედვით სრულდება კვეთისა და ბრუნვის ოპერაციები (ნახ.4-ა,ბ).



ა) ბ) გ)

ნახ. 3



ა) ბ)

ნახ. 4

შედეგად შეიძლება გაკეთდეს ცალკეული ფაკულტეტის, სასწავლო ჯგუფისა, თუ სტუდენტის აკადემიური მაჩვენებლების, აგრეთვე კვლევით სამუშაოებში მონაწილეობის ანალიზი. OLAP კუბის გამოყენებით შესაძლებელია აგრეთვე პედაგოგთა რეიტინგული კლასიფიკაცია, რომელიც პედაგოგებისათვის საკონკურსო პირობების შესრულების ან სახელფასო განაკვეთის განსაზღვრის საფუძველი შეიძლება გახდეს.

მონაცემთა საცავების როლი გადაწყვეტილების მიღების პროცესში

უნივერსიტეტებში ხარისხის უზრუნველყოფა მას შემდეგ გახდა საკვანძო საკითხი, როდესაც მან მოიცვა არამხოლოდ კვლევა და სწავლება, არამედ ყველა შიდა პროცესი, რომელიც აუცილებელია სტუდენტთა ეფექტური განათლების მისაღებად. ხარისხის უზრუნველყოფის სისტემის ძირითადი მიზანია მუდმივი გაუმჯობესებისკენ სწრაფვა, რაც მოითხოვს მონაცემების შეგროვებასა და სისტემატურ დამუშავებას მიმდინარე სიტუაციის რეალური სურათის მისაღებად. მონაცემთა საცავებს შეუძლიათ წარმოგვიდგინონ მნიშვნელოვანი ინფორმაცია ხარისხის უზრუნველყოფის ყველა საფეხურზე.

უმაღლეს სასწავლებლებში პროცესების მონიტორინგისა და აღრიცხვიანობის დროს აღძრული ზოგიერთი პრობლემა დაკავშირებულია სხვადასხვა მონაცემთა წყაროებიდან სანდო მონაცემების შეგროვებასთან. აღნიშნული პრობლემის გადალახვა საჭიროებს მონაცემთა

საცავისა და ინტეგრირებული საინფორმაციო სისტემის მხარდაჭერას, რომელიც შეიცავს განახლებულ და საიმედო ინფორმაციას სასწავლო პროგრამებზე, სასწავლო გეგმებზე, პედაგოგებზე, რესურსებზე, სტუდენტებზე და სხვა.

მონაცემთა საცავი და მონაცემთა ინტელექტუალური ანალიზის სისტემები ხასიათდებიან ინფორმაციაზე წვდომის სიმარტივით და სისწრაფით. საცავში განთავსებული ინფორმაცია უნდა პასუხობდეს განსაზღვრულ მოთხოვნებს, როგორიცაა: საგნობრივი ორიენტაცია, ინტეგრირებულობა, ქრონოლოგიის მხარდაჭერა და შეუცვლელობა. მრავალ ფუნქციათა შორის, რომელსაც მონაცემთა საცავი ასრულებს, ძირითადი დანიშნულებაა მონაცემთა და ინფორმაციის ზუსტი წარმოდგენა უმოკლეს ვადებსა და მინიმალური დანახარჯებით. საცავში მონაცემთა ორგანიზაციის სქემა ნაჩვენებია მე-5 ნახაზზე.

ნახ.5. საცავში მონაცემთა ორგანიზაციის სქემა

არქიტექტურული გადაწყვეტის თვალსაზრისით მონაცემთა საცავები ახორციელებს თავის ფუნქციებს მონაცემთა ვიტრინების მეშვეობით. მონაცემთა კლასიკური საცავის დადებითი მხარეებია:

- საერთო სემანტიკა;
- ცენტრალიზებული, მართვადი გარემო;
- მონაცემთა ამოღებისა და ბიზნეს-ლოგიკის პროცესების გამოყენების შეთანხმებული კრებული;
- განთავსებული ინფორმაციის არაწინააღმდეგობრიობა;
- შაბლონების მიხედვით ადვილად შექმნადი და შევსებადი მონაცემთა ვიტრინები;
- მეტამონაცემების ერთიანი რეპოზიტორიუმი;
- მონაცემთა დამუშავებისა და წარმოდგენის მექანიზმების მრავალფეროვნება.

ასეთი საცავებისათვის ნაკლოვანებად შეიძლება ჩაითვალოს რეალიზაციის დიდი დანახარჯები, მაღალი რესურსტევადობა მთელი ორგანიზაციის მასშტაბით, რთული სერვისული სისტემების არსებობისადმი მოთხოვნილება, რისკის შემცველი განვითარების სცენარი, როცა ყველა მონაცემი და მეტამონაცემი იმყოფება ერთ რეპოზიტორიუმში და არახელსაყრელ შემთხვევაში შეიძლება დაიკარგოს.

გარდა ამისა, „ნედლი“ მონაცემების ფილტრაციის, აგრეგირებისა და რაფინირების დროს ბევრი ინფორმაცია იკარგება, რომელიც მეტად სასარგებლო შეიძლება იყოს ანალიზის დროს. ამასთან დაკავშირებით ჩამოყალიბდა აზრი, რომ გარდა მონაცემთა განთავსებისა და ამოღების მექანიზმებისა, რეპოზიტორიუმებისა და ვიტრინებისა, უნდა არსებობდეს შესაბამისი სივრცე

„ნედლი“ მონაცემებისა და მათი რეალურ დროის რეჟიმში მრავალგანზომილებიანი ანალიზისათვის, რასაც წარმოადგენს - OLAP (On Line Analytical Processing).

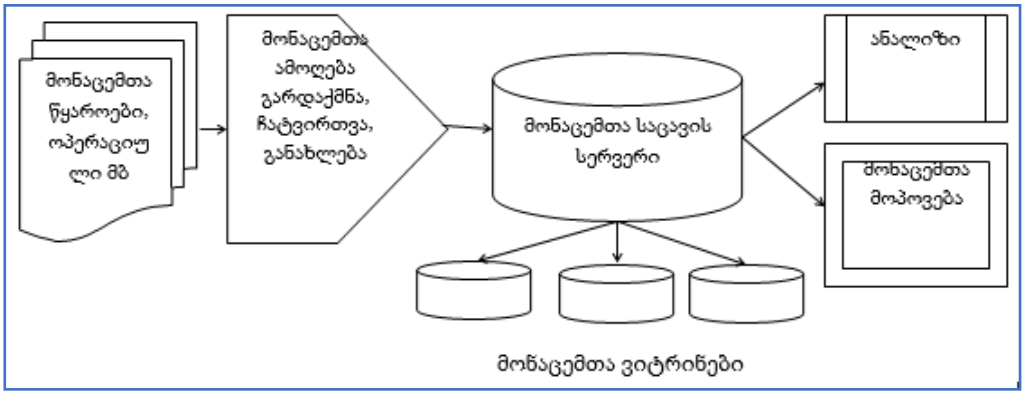
OLAP მონაცემთა სხვადასხვა განზომილებაში გამოკვლევის საშუალებას იძლევა. მომხმარებლებს შეუძლიათ აირჩიონ, თუ რომელი მაჩვენებლების ანალიზი და განზომილება რა სახით უნდა აისახოს კროს-ცხრილში, მოხდეს სტრიქონებისა და სვეტების გაცვლა, შემდგომ ამოჭრები განზომილებათა გარკვეულ კომბინაციაზე კონცენტრირებისთვის. დეტალიზაციისა და გამსხვილების დონეებზე გადაადგილებით შესაძლებელია შეიცვალოს მონაცემთა დეტალიზაციის ხარისხი.

მონაცემთა საცავი შეიცავს რამდენიმე ოპერატიული მონაცემთა ბაზიდან შეგროვებულ ინფორმაციას. იგი იქმნება სპეციალურად გადაწყვეტილების მიღების მხარდამჭერი დანართისთვის და წარადგენს გარკვეულ დროში დაგროვებულ, შემაჯამებელ და კონსოლიდირებულ მონაცემებს, რომლებიც გაცილებით მისაღებია ანალიზისთვის, ვიდრე დეტალური ინდივიდუალური ჩანაწერები. მუშა დატვირთვა შედგება არასტანდარტული, რთული მოთხოვნებისგან, რომლებიც მიმართავენ ჩანაწერების დიდ რაოდენობას და ასრულებენ სკანირების, გაერთიანებისა და აგრეგირების ოპერაციათა აურაცხელ რაოდენობას. მოთხოვნაზე პასუხის დრო მოცემულ შემთხვევაში გაცილებით მნიშვნელოვანია, ვიდრე გამტარუნარიანობა.

გადაწყვეტილების მიღება უნდა ეყრდნობოდეს სამართავი ობიექტის შესახებ რეალურ მონაცემებს. ასეთი მონაცემები ჩვეულებრივ ინახება ოპერატიულ მონაცემთა ბაზაში. თუმცა ოპერატიული მონაცემები ვერ დააკმაყოფილებს ანალიზის მიზნებს, რამდენადაც ანალიზისა და გადაწყვეტილების მისაღებად ძირითადად საჭიროა აგრეგირებული ინფორმაცია. გარდა ამისა, ანალიზის მიზნებისთვის საჭიროა ინფორმაციით სწრაფად მანიპულირების, მისი სხვადასხვა ასპექტში წარმოდგენის, სხვადასხვა არარეგლამენტირებული მოთხოვნების წარმოების შესაძლებლობა, რაც რთულად რეალიზებადია ოპერატიულ მონაცემებზე ტექნოლოგიური სირთულეების გამო. მონაცემთა საცავის მოცულობა, როგორც წესი, გაცილებით დიდია ოპერატიულ მონაცემთა ბაზის მოცულობაზე. საცავს აქვს ოპერატიული მონაცემთა ბაზისგან დამოუკიდებელი მხარდაჭერა, რადგან ანალიტიკური დანართების ფუნქციონალურობასა და მწარმოებლურობაზე მოთხოვნა განსხვავდება ტრანზაქციული სისტემებისადმი მოთხოვნებისგან.

მონაცემთა საცავის ბაზაზე შესაძლებელია ანგარიშგებების მომზადება, მონაცემთა ანალიზი OLAP ტექნოლოგიისა და მონაცემთა ინტელექტუალური ანალიზის (Data Mining) დამხარებით. ისინი საშუალებას გვაძლევენ ჩავატაროთ პრობლემის გაცილებით სრული და ღრმა ანალიზი, მივიღოთ მეტად საფუძვლიანი გადაწყვეტილება.

მე-6 ნახაზზე წარმოდგენილია გადაწყვეტილების მიღების მხარდამჭერი სისტემის არქიტექტურა, რომელიც შედგება სამი ძირითადი კომპონენტისგან: მონაცემთა საცავის სერვერისგან, მონაცემთა ანალიზის და მოპოვებს ინსტრუმენტებისგან და მონაცემთა საცავის საბაზო საშუალებებისგან:

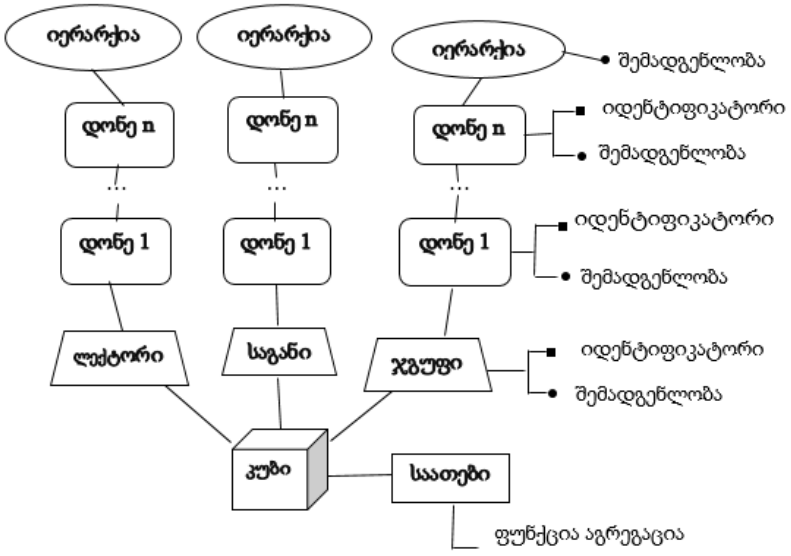


ნახ. 6. გადაწყვეტილების მიღების მხარდამჭერი სისტემის არქიტექტურა

OLAP კუბებისა და ჰიპერკუბის აგება

განზომილებებისა და ფაქტების ცხრილების შექმნის შემდეგ ავაგოთ კუბი, რისთვისაც ვირჩევთ ფაქტების ცხრილს მომავალი კუბისთვის. ჩვენ შემთხვევაში ეს არის ცხრილი - shedule. შემდეგ ფაქტების ცხრილიდან ავირჩიოთ ერთი ან რამდენიმე ველი, რომლის საფუძველზეც გამოითვლება კუბის ზომები. ავირჩიოთ ველები: group_number, subject_name, hours. შემდეგი ბიჯი არის კოლექტიური განზომილებების არჩევა, რომელიც გამოიყენება მოცემულ კუბში. ავირჩიოთ lecturer_name, lecturer_lastname, lecture_type. გარდა ამისა დავამატოთ ახალი, კერძო განზომილება department. ამდენად, ჩვენ განვსაზღვრეთ კუბის მეტამონაცემები. კუბების რედაქტორში შეგვიძლია შევიტანოთ ცვლილებები. საჭიროების შემთხვევაში შეიძლება შესწორებების შეტანა კუბის განსაზღვრაში. მაგალითად, დავამატოთ ან წავშალოთ განზომილებები ან ზომები, შევქმნათ გამოთვლითი მნიშვნელობები.

კუბის განზომილებები დავყოთ იერარქიებად, რაც აუცილებელია მაჩვენებლების მნიშვნელობების აგრეგაციისა და დეტალიზაციისთვის იერარქიული სტრუქტურის შესაბამისად. მე-7 ნახაზზე წარმოდგენილია კუბის იექრარქიებად დაშლის განზოგადოებული სქემა:



ნახ. 7. დატვირთვის კუბის შემადგენელი ელემენტები

ჰიპერკუბი ნახაზზე გამოსახულია კუბის სიმბოლოთი. მასთან დაკავშირებულია კუბის განზომილება, ასევე ზომა. მოცემულია იერარქია და იერარქიის დონეები.

გამოვიყენოთ ბალანსირებული იერარქია, რადგან ჩვენ ამოცანაში დონეთა რიცხვი განსაზღვრულია მისი სტრუქტურით და შეუცვლელია, ხოლო იერარქიული ხის ცალკეული ტოტი შეიცავს ობიექტებს ცალკეული დონიდან. ლექტორების დატვირთვის ამოცანაში: ცალკეული ლექტორი ასწავლის რამდენიმე საგანს, ხოლო ცალკეული საგანი ისწავლება რამდენიმე ჯგუფში. ამდენად, შეიძლება ითქვას, რომ ლექტორების დატვირთვის ამოცანაში საქმე გვაქვს ამ ობიექტების სამდონიან იერარქიასთან. მოცემულ შემთხვევაში იერარქიის პირველ დონეზე განთავსებიან ლექტორები, მეორეზე - საგნები, ხოლო მესამეზე - ჯგუფები.

ჯგუფების დატვირთვის ამოცანაში: ცალკეულ ჯგუფში ისწავლება რამდენიმე საგანი, ხოლო ცალკეულ საგანს შეიძლება ასწავლიდეს რამდენიმე ლექტორი. მაგალითად, ლექციას ატარებდეს ერთი, ხოლო ლაბორატორიულს ან პრაქტიკულს - მეორე.

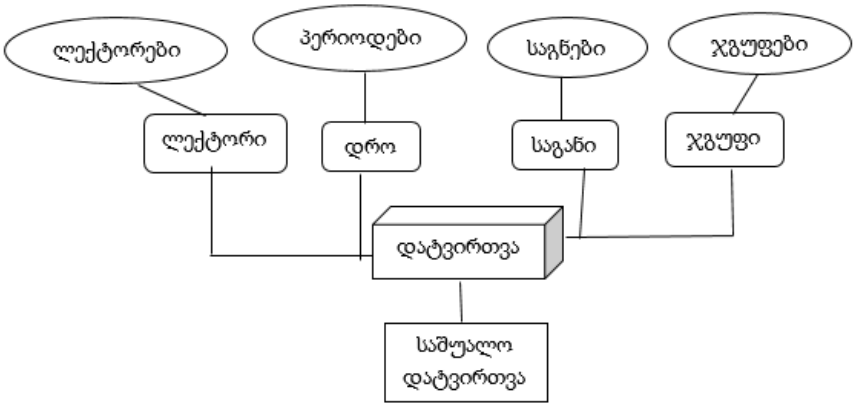
როგორც ვხედავთ ბალანსირებული იერარქიის ფორმირებისთვის საჭიროა „ერთი-მრავალთან“ კავშირის არსებობა ნაკლებად დეტალური დონის ობიექტებსა და გაცილებით დეტალური დონის ობიექტებს შორის. პრინციპში ბალანსირებული იერარქიის ცალკეული დონე შეიძლება წარმოვადგინოთ, როგორც ცალკე მარტივი განზომილება, მაგრამ მაშინ ეს განზომილებები აღმოჩნდებიან დამოკიდებულები, ამდენად საჭიროა კუბის ჭრის შესაძლებლობის ამაღლება.

გამსხვილებული კოორდინატების განზომილებებთან გათანაბრება მრავალგანზომილებიანი მოდლირების მნიშვნელოვანი თავისებურებაა. ეს ეს ელემენტები შეიძლება გამოყენებულ იქნან განზომილებათა კოორდინატების სახით ჰიპერკუბის უჯრედების ადრესაციისთვის ზუსტად ისევე, როგორც ჩვეულებრივი ფაქტ-კოორდინატები. გამსხვილებულ კოორდინატებს ჰიპერკუბში შეესაბამება ეგრეთწოდებული აგრეგირებული ანუ შემაჯამებელი უჯრედები, ფაქტების უჯრედებისგან განსხვავებით, რომელთა ყველა კოორდინატი წარმოადგენს ფაქტ-კოორდინატებს. მნიშვნელობები, რომლებიც შედიან შემაჯამებელ უჯრედებში და რომლებიც შეესაბამებიან იერარქიის რომელიმე დონის გამსხვილებულ კოორდინატებს, ფორმირდებიან იერარქიის ქვედა დონის კოორდინატების შესაბამისი უჯრედების მნიშვნელობების ბაზაზე.

ჰიპერკუბში გამსხვილებულ კოორდინატებს შეესაბამება აგრეგირებული უჯრედები, რომლის ყველა კოორდინატი არის ფაქტ-კოორდინატი. შემაჯამებელ უჯრებში განთავსებული მნიშვნელობები ფორმირდება იმ უჯრების მნიშვნელობების საფუძველზე, რომლებიც იმყოფებიან იერარქიის ქვედა დონეზე.

ზოგიერთი განზომილება შეიძლება გავლენას არ ახდენდეს ჯამურ მაჩვენებელზე, ამიტომ შეიძლება არ იქნას გათვალისწინებული აგრეგაციის პროცესში. ეს ეფექტი განსაკუთრებით ვლინდება ფესვური უჯრების შეჯამებისას.

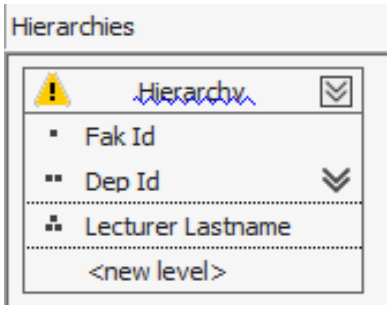
მე-8 ნახაზზე წარმოდგენილია ჰიპერკუბის მოდელი დატვირთვის შესაფასებლად. ზომა: „საშუალო დატვირთვა“ ეფუძვნება ფაქტს: „დატვირთვა“. მასში მონაწილეობს ოთხი განზომილება: „ლექტორი“, „საგანი“, ჯგუფი“, „დრო“. დავუშვათ, არ გვაქვს შუალედური იერარქიები. განვიხილოთ ფესვური უჯრების შეჯამების შემთხვევა: ვთქვათ, გვინდა გამოვითვალოთ ჯამური მაჩვენებელი, რომელიც არის ფესვური „ლექტორი“ და „ჯგუფი“ განზომილების მიხედვით.



ნახ. 8. ჰიპერკუბის მოდელი დატვირთვის შესაფასებლად

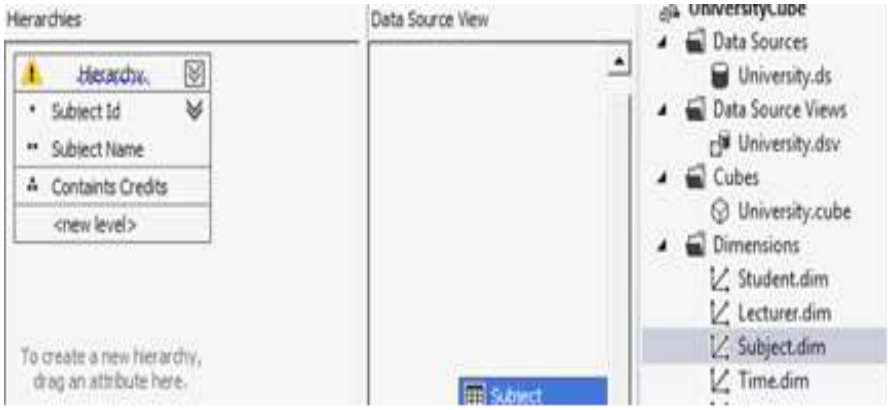
მაგალითად, გამოვითვალოთ კონკრეტული ლექტორის საშუალო ჯგუფური დატვირთვა. აქედან გამომდინარე შესაძლებელია გამოვითვალოთ დეპარტამენტის ყველა ლექტორის საშუალო დატვირთვა. მაგრამ, როდესაც გვინდა გამოვითვალოთ მაგალითად, ლექტორების დატვირთვა დაპარტამენტების, ფაკულტეტების მიხედვით, მაშინ საჭიროა განზომილებების დაყოფა იერარქიებად.

მე-9 ნახაზზე წარმოდგენილია ლექტორების იერარქიის სტრუქტურა, სადაც პირველ დონეზე არის ფაკულტეტი, მეორეზე - დეპარტამენტი, ხოლო მესამე დონეზე - ლექტორი:



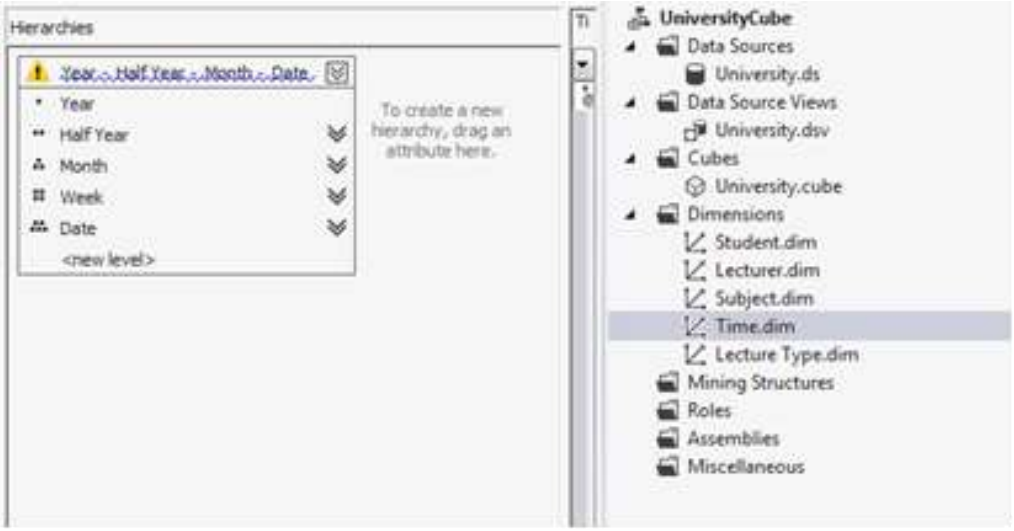
ნახ. 9. „ლექტორები“ განზომილების იერარქია

შესაბამისად, წარმოვადგინოთ საგნის იერარქია, სადაც პირველ დონეზეა საგნის კოდი, მეორეზე - საგნის დასახელება, ხოლო მესამეზე - საგნისთვის განკუთვნილი კრედიტების რაოდენობა. ნახ.10.-ზე წარმოდგენილია საგნის იერარქიის სტრუქტურა:



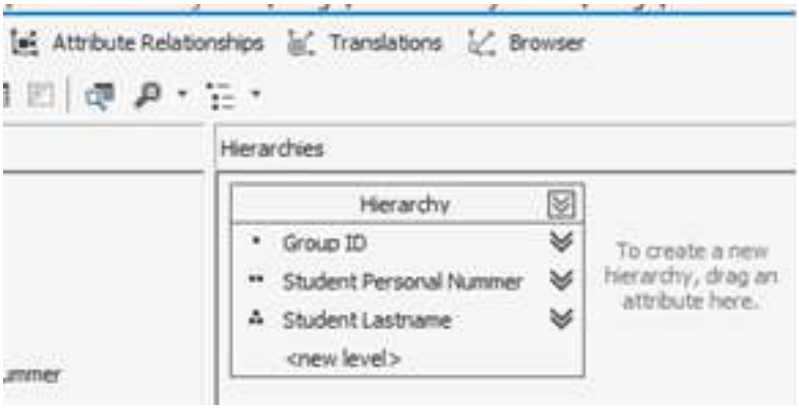
ნახ. 10. „საგანი“ განზომილების იერარქია

დროის იერარქია წარმოდგენილია შემდეგი დონეების სახით: წელი, სემესტრი ანუ წელიწადის ნახევარი, თვე, კვირა და დღე. ნახ.11.-ზე წარმოდგენილია დროის (პერიოდის) იერარქიის სტრუქტურა:



ნახ. 11. „დრო“ განზომილების იერარქია

ჯგუფური მეცადინეობების გარდა, რაშიც მოიაზრება ლექციები, პრაქტიკული და ლაბორატორიული მეცადინეობები, ჯგუფისთვის გათვალისწინებულია საკონსულტაციო საათები და ასევე ინდივიდუალური მუშაობა სტუდენტებთან, რაც მოითხოვს „სტუდენტი“ განზომილების დაყოფას იერარქიებად. მე-12 ნახაზზე წარმოდგენილია „სტუდენტი“ განზომილების იერარქიული სტრუქტურა, სადაც იერარქიის პირველ დონეზე არის ჯგუფი, ხოლო შემდეგ დონეზე - სტუდენტი:

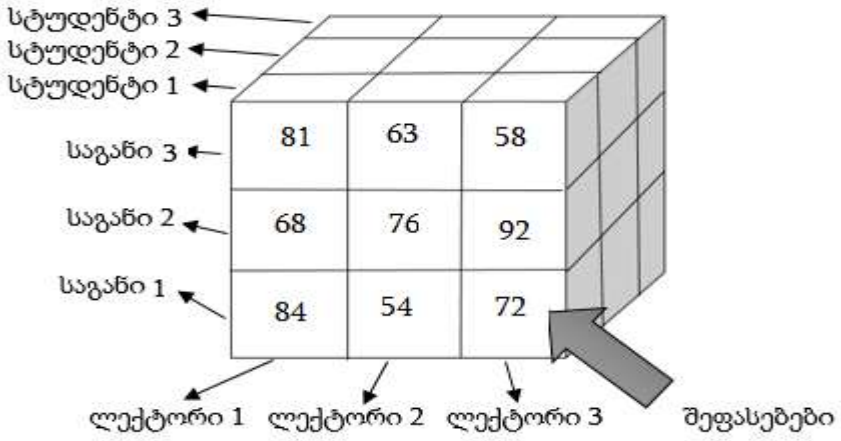


ნახ.12. „სტუდენტი“ განზომილების იერარქია

როგორც დასაწყისში ავღნიშნეთ, ჩვენ ვამუშავებთ ორ ამოცანას. მეორე ამოცანა არის სტუდენტთა აკადემიური მოსწრების ანალიზის ამოცანა. კუბის ასაგებად ავირჩიოთ ფაქტების ცხრილი მომავალი კუბისთვის. მოცემული ამოცანისთვის ეს იქნება ცხრილი - Rating. შემდეგ ფაქტების ცხრილიდან ავირჩიოთ ერთი ან რამდენიმე ველი, რომლის მიხედვითაც შემდგომში გამოითვლება კუბის ზომები მოცემული ამოცანისთვის. ავირჩიოთ ველები: Subject_name, Student_number, Rating_point. შემდგომ ეტაპზე ავირჩიოთ განზომილებები, რომლებიც გამოიყენება მოცემულ კუბში: Lecturer_name, lecturer_lastname, group_number. გარდა ამისა

დავამატოთ ახალი, კერძო განზომილება: Faculty. ამრიგად, ჩვენ განვსაზღვრეთ კუბის მეტამონაცემები. საჭიროების მიხედვით კუბების რედაქტორში შეგვიძლია შევიტანოთ ცვლილებები. დავამატოთ განზომილებები და ფაქტები ან წავშალოთ ზედმეტად რაც მიგვაჩნია. შევქმნათ გამოთვლითი მნიშვნელობები.

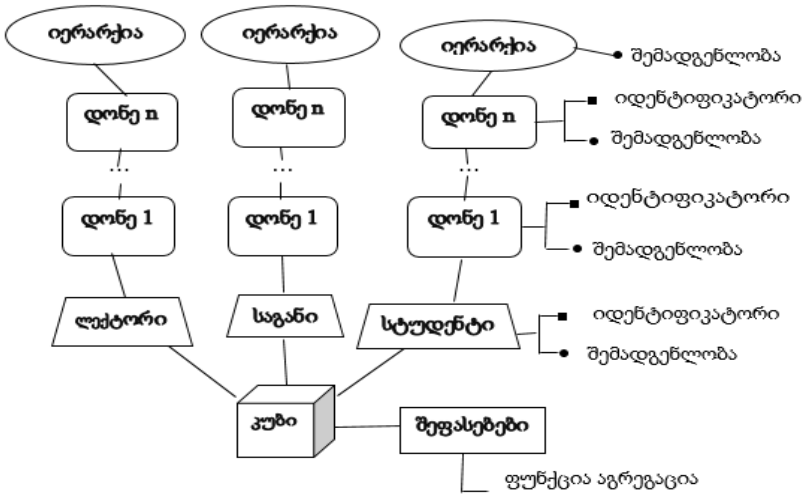
კუბში წარმოდგენილია სამი განზომილება, რომელიც შეესაბამება x, y და z კოორდინატებს. განზომილებებად აღებულია: Lecturer, Subject_name, Student_number:



ნახ. 13.

კუბი განზომილებებით: Lecturer, Subject_name, Student_number

მნიშვნელობების აგრეგაციისა და დეტალიზაციისთვის საჭიროა კუბის განზომილებები დავყოთ იერარქიულ სტრუქტურებად.

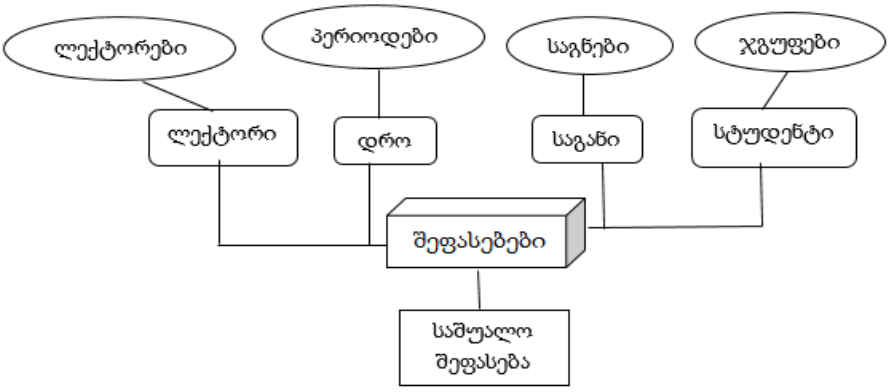


ნახ. 14. შეფასებების კუბის შემადგენელი ელემენტები

ჩვენი ამოცანისთვის გამოვიყენოთ ბალანსირებული იერარქია, რადგან მოცემულ შემთხვევაში დონეთა რიცხვი განსაზღვრულია მისი სტრუქტურით და შეუცვლელია, ხოლო იერარქიული ხის ცალკეული ტოტი შეიცავს ობიექტებს ცალკეული დონიდან.

როგორც წინა ამოცანაში განვიხილეთ, მოცემულ ამოცანაში ანალოგიურად გვექნება აგრეგირებული უჯრედები. შემაჯამებელ უჯრებში განთავსდებიან მნიშვნელობები, რომელთა ფორმირება მოხდება იერარქიის ქვედა დონეზე განთავსებული უჯრების მნიშვნელობების საფუძველზე. ის განზომილებები, რომლებიც გავლენას არ ახდენენ ჯამურ მაჩვენებელზე, აგრეგაციის პროცესში არ იქნება გათვალისწინებული.

მე-15 ნახაზზე წარმოდგენილია ჰიპერკუბის მოდელი აკადემიური მოსწრების შესაფასებლად:

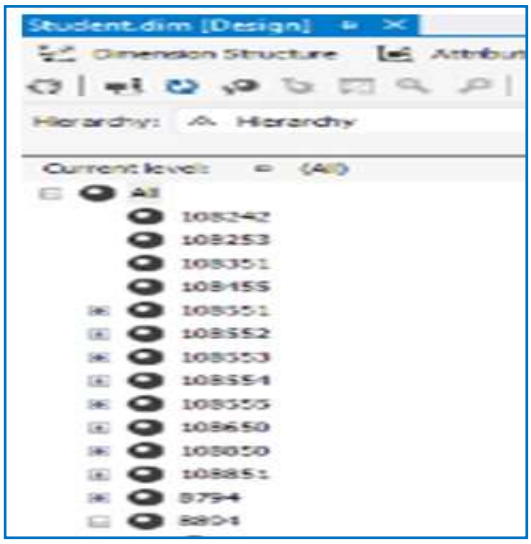


ნახ.15. ჰიპერკუბის მოდელი აკადემიური მოსწრების შესაფასებლად

ზომა „საშუალო ქულა“, ეფუძვნება ფაქტს „შეფასება“, მასში მონაწილეობს ოთხი განზომილება: Lecturer, Subject, Student, Time. ვთქვათ არ გვაქვს შუალედური იერარქიები. განვიხილოთ ფესვური უჯრების შეჯამების ორი შემთხვევა: ვთქვათ, გვინდა გამოვითვალოთ ჯამური მაჩვენებელი, რომელიც არის ფესვური Student და Subject განზომილების მიხედვით. მაგალითად, გამოვითვალოთ საშუალო ქულა შეფასებული განსაზღვრული მასწავლებლის მიერ. ეს იქნება გასაშუალებული ქულა ყველა სტუდენტთან მიმართებით.

ერთ მასწავლებელს გამოცდაზე შეუძლია მიიღოს მრავალი სტუდენტი, ამიტომ ჯამური მაჩვენებლის მიღებისთვის საჭიროა Rating მაჩვენებლის აგრეგირება. აგრეგირებაში მონაწილეობს განზომილება: Lecturer. მეორე შემთხვევაში გვინდა გამოვითვალოთ სტუდენტის ჯამური მაჩვენებელი, მაგალითად, სტუდენტის საშუალო ქულა ყველა საგანში. ამ შემთხვევაში აგრეგაციაში მონაწილეობს განზომილებები: Student და Subject. თითქოს მოთხოვნა ფორმულირებულია წინას ანალოგიურად, მაგრამ სიტუაცია აქ სხვაა. კონკრეტულ სტუდენტს კონკრეტულ საგანში აქვს ერთი შეფასება, რომელიც შეფასებულია ერთ-ერთი მასწავლებლისგან ერთ რომელიმე კონკრეტულ დღეს. ამიტომ აგრეგაცია მასწავლებლის და თარიღის მიხედვით არ მოითხოვება.

ამოცანაში ცალკეული სტუდენტის მონაცემების აგრეგირების გარდა საჭიროა მთელი ჯგუფის მონაცემების აგრეგირება. ამ შემთხვევაში უნდა მოხდეს განზომილების დონეების მიხედვით მონაცემების გაერთიანება. შვილი ელემენტების აგრეგირების შედეგი გადაეცემა მშობელ ელემენტს. სტუდენტის იერარქიის ზედა დონეზე არის ჯგუფი. გამოვითვალოთ ყველა საგნის მიხედვით ჯგუფის საშუალო ქულა ანუ გასაშუალებული ქულა სტუდენტებთან მიმართებით. მე-16 ნახაზზე წარმოდგენილია ერთი-მრავალთან იერარქია ჯგუფები-სტუდენტები, სადაც იერარქიის ზედა დონეზე განთავსებულია ჯგუფი.



ნახ.16. იერარქია ჯგუფი-სტუდენტი

მოცემული იერარქიის გათვალისწინებით 1-ცხრილში ნაჩვენებია კუბის ასახვის ვარიანტი აგრეგირებით. მშობელ ელემენტებში აგრეგირდება მონაცემები ყველა შვილი ელემენტებიდან. აგრეგირების მეთოდია - სტუდენტების შეფასებათა საშუალო არითმეტიკული ცალკეული საგნის მიხედვით.

მონაცემთა აგრეგირების მაგალითი *ცხრ.1*

	საგანი 1	საგანი 2	საგანი 3	საგანი 4	საგანი 5
ჯგუფი	67.5	68.75	69.5	65.5	74.25
სტუდენტი1	56	65	70	58	87
სტუდენტი2	66	58	72	70	52
სტუდენტი3	64	71	58	54	82
სტუდენტი4	84	81	78	80	76

რეკომენდებული ლიტერატურა:

1. Srinath Perera, Thilina Gunarathne. Hadoop MapReduce Cookbook. Published by Packt Publishing Ltd.Livery Place 35 Livery Street Birmingham B3 2PB, UK. ISBN 978-1-84951-728-7, CD-3117

2. Alex Holmes. Hadoop in Practice. ©2012 by Manning Publications Co. All rights reserved., ISBN 9781617290237, CD-3117

3. მეფარიშვილი ბ. დიდ მონაცემთა შენახვა და დამუშავება. 2018. -176 გვ. ISBN 978-9941-27-789-4. ბიბლ. ინდ. 004.65(02)

4. ჯანელიძე გ, მეფარიშვილი ბ, მონაცემთა საცავებში ბიზნესანალიზი Sql Server Analysis Services გამოყენებით, ISBN 978-9941-27-767-2 160გვ. ბიბლ.ინდ. 004.65(02)5.

5. Big Data Fundamentals: Concepts, Drivers & Techniques. 2016.

6. https://b-ok.org/book/2646614/4336af

7. https://www.researchgate.net/publication/272402027_Big_Data_A_Revolution_That_Will_Transform_How_We_Live_Work_and_Think

8. https://b-ok.org/book/2149893/012202

9. https://rutracker.org/forum/viewtopic.php?t=5218444

10. Beridze N., Janelidze G. Organizing Decision-Making Support System Based Multi-Dimensional Analysis of the Educational Process Data. Journal of ICT, Design, Engineering and Technological Science (JITDETS) VOL. 4, NO. 1, pp. 1-5, 2020 DOI: https://doi.org/10.33150/JITDETS-4.1.1

11.https://b-ok.org/book/2189674/308d9f

12. მეფარიშვილი ბ., ცერცვაძე გ., ჯანელიძე გ. დიდი მონაცემების ანალიტიკა, სტუ, 2020. 244 გვ. ISBN 978-9941-8-2161-5. ბიბლ. ინდ. 004.65(02)/12. CD-5677

გადაეცა წარმოებას 12..11.2020. ხელმოწერილია დასაბეჭდად 20.11.2020. ოფსეტური ქაღალდის ზომა 60X84 1/16. პირობითი ნაბეჭდი თაბახი 2. ტირაჟი 50 ეგზ.

სტუ-ს „IT კონსალტინგის ცენტრი“ (თბილისი, მ.კოსტავას 77)

ISBN 978-9941-8-2869-0