ՀԱՅԱՍՏԱՆԻ ՀԱՆՐԱՊԵՏՈՒԹՅԱՆ ԿՐԹՈՒԹՅԱՆ ԵՎ ԳԻՏՈՒԹՅԱՆ

ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ

ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

ՄԱԳԻՍՏՐՈՍԱԿԱՆ ԹԵԶ

**ԹԵՄԱ`** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ՄԱԳԻՍՏՐԱՆՏ**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ՄԱՍՆԱԳԻՏՈՒԹՅՈՒՆ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ԿՐԹԱԿԱՆ ԾՐԱԳԻՐ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ՈՐԱԿԱՎՈՐՄԱՆ ԱՍՏԻՃԱՆ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_մագիստրոսի

ԵՐԵՎԱՆ 2018

Հ Ա Ս Տ Ա Տ Մ Ա Ն Թ Ե Ր Թ

**ԹԵՄԱ`\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Թեզի ղեկավար`\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ԱԱՀ, գիտ.աստիճան, կոչում

Մագիստրանտ`\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Գրախոս`\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ԱԱՀ, գիտ.աստիճան, կոչում

Ամբիոնի վարիչ`\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ԱԱՀ, գիտ.աստիճան, կոչում

Ինստիտուտի տնօրեն/Ֆակուլտետի դեկան`\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ԱԱՀ, գիտ.աստիճան, կոչում

Տ Վ Յ Ա Լ Ն Ե Ր Շ Ր Ջ Ա Ն Ա Վ Ա Ր Տ Ի Մ Ա Ս Ի Ն

Մագիստրանտ՝\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Մասնագիտություն`\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Կրթական ծրագիր`\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ծննդյան տարեթիվը՝\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Մինչ մագիստրոսական որակավորումը՝\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_բակալավր

Մասնագիտությունը`\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Կրթական ծրագիրը`\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Հրատարակված աշխատանքներ`\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ԻՆՍՏԻՏՈՒՏ/ՖԱԿՈՒԼՏԵՏ

(Ինստիտուտի/ֆակուլտետի անվանումը)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ԱՄԲԻՈՆ

(ամբիոնի անվանումը)

Մասնագիտություն\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_դասիչ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (գրել առանց հապավումների)

Կրթական ծրագիր \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(գրել առանց հապավումներ)

Թիվ\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ակադեմիական խմբի

(խմբի համարը)

(ուսանողի ազգանուն / անուն / հայրանուն)

**ՄԱԳԻՍՏՐՈՍԱԿԱՆ ԹԵԶԻ ԱՌԱՋԱԴՐԱՆՔ**

1. Թեման \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Հաստատված է ՀԱՊՀ 20 թ.-ի \_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_\_\_\_\_» թիվ \_\_\_\_\_\_\_ հրամանով:

2. Նախնական տվյալներ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Հաշվեբացատրագրի բովանդակություն (բաժիների և մշակման ենթակա հարցերի թվարկմամբ)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Թեզի կատարման օրացուցային պլան

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Հ/Հ | Թեզի կատարման փուլերը | | | Ծանոթություն |
| Անվանումը | Կատ.  ժամկ. | հաշ. ձևը |
| 1. |  |  |  |  |
| 2. |  |  |  |  |
| 3. |  |  |  |  |
|  | I ատեստավորում |  | 20 % |  |
| 4. |  |  |  |  |
| 5. |  |  |  |  |
| 6. |  |  |  |  |
|  | II ատեստավորում |  | 60 % |  |
| 7. |  |  |  |  |
| 8. |  |  |  |  |
| 9. |  |  |  |  |
| 10. |  |  |  |  |
|  | III ատեստավորում |  | 100% |  |
| 11. | Աշխատանքի ներկայացումը ամբիոն |  | Ավ. աշխ. |  |
| 12. | Նախնական պաշտպանություն |  |  |  |

6. Աշխատանքի պաշտպանության օրը \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

7. Ամբիոնի վարիչ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ա.Ա.Հ., ստորագրություն, ամսաթիվ)

8. Աշխատանքի ղեկավար \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ա.Ա.Հ., ստորագրություն, ամսաթիվ)

9. Աշխատանքի առաջադրանքը ստացա \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ուսանողի Ա.Ա.Հ., ստորագրություն, ամսաթիվ)

# ՀԱՄԱՌՈՏԱԳԻՐ

**ԹԵՄԱ՝ Կիսահաղորդչային տվյալների համար պահպանման մեթոդների հետազոտումն ու դրանց արագ հասանելիության կազմակերպումը**

Գասպարյան Լևոն Աշոտի

Սույն մագիստրոսական ատենախոսության շրջանակներում ուսումնասիրվել են կիսահաղորդչային տվյալների կառուցվածքը և օգտագործման` պահեստավորման ու մշակման, առանձնահատկությունները։ Ուսումնասիրվել, համեմատվել և փորձարկվել են մեծ տվյալների պահպանման և մշակման հիմնական խնդիրներ, տվյալների պահոցների տեսակներ, ոչ հարաբերական տվյալների բազաների լուծումներ։ Խնդրի դրվածքից և կիսահաղորդչային տվյալների կառուցվածքային ու օգտագործման առանձնահատկություններից ելնելով հետագա ուսումնասիրության ու փորձարկման համար ընտրվել է ոչ հարաբերական տվյալների բազաների տեսակը։ Կիսահաղորդչային տվյալները ոչ հարաբերական տվյալների բազայում պահպանման համար ստեղծվել, մշակվել և կիրառվել են երեք մեթոդներ։ Ստեղծվել է գործիք, որը իրականացուն և փորձարկում է այդ մեթոդները։ Կատարվել են փորձարկումներ և հետազոտություններ նշված մեթոդնորի համար, պարզելու տվյալների պահուստավորման արագության կախվածությունը՝ տվյալների քանակից, տվյալների տեսակից և դրանց փոփոխման արագությունից:

Բովանդակություն

[ՀԱՄԱՌՈՏԱԳԻՐ 6](#_Toc513989203)

[ՆԿԱՐՆԵՐԻ ՑԱՆԿ 10](#_Toc513989204)

[ԱՂՅՈՒՍԱԿՆԵՐԻ ՑԱՆԿ 11](#_Toc513989205)

[ՆԵՐԱԾՈՒԹՅՈՒՆ 12](#_Toc513989206)

[Հասկացություններ և սահմանումներ 13](#_Toc513989207)

[ԳՐԱԿԱՆՈՒԹՅԱՆ ԱԿՆԱՐԿ 15](#_Toc513989208)

[Հարաբերական տվյալների բազաներ 15](#_Toc513989209)

[Ոչ հարաբերական տվյալների բազաներ 16](#_Toc513989210)

[Ուղղահայաց մասշտաբավորում 17](#_Toc513989211)

[Հորիզոնական մասշտաբավորում 18](#_Toc513989212)

[Հարաբերական և ոչ հարաբերական տվյալների բազաների համեմատություն 18](#_Toc513989213)

[Hadoop համակարգ 20](#_Toc513989214)

[Hive համակարգ 21](#_Toc513989215)

[Տվյալների տիպեր 21](#_Toc513989216)

[SQL հրամաններ 23](#_Toc513989217)

[MapReduce 27](#_Toc513989218)

[Ընդարձակ նշման լեզու (XML) 31](#_Toc513989219)

[ԽՆԴՐԻ ԴՐՎԱԾՔ 34](#_Toc513989220)

[ՏԵՍԱԿԱՆ ԱՌՆՉՈՒԹՅՈՒՆՆԵՐ 35](#_Toc513989221)

[Կիսահաղորդչային թեստավորման տվյալների ձևաչափեր 35](#_Toc513989222)

[Տվյալների պահուստի ընտրություն 35](#_Toc513989223)

[Hive գործիքի ներդրված մեթոդներ 37](#_Toc513989224)

[SDFReader ներդրված մեթոդ 38](#_Toc513989225)

[SDF տվյալների հասանելիություն 40](#_Toc513989226)

[c\_sdf\_wrapper գրադարան 41](#_Toc513989227)

[c\_sdf\_wrapper գրադարանի իրականացումը 42](#_Toc513989228)

[HiveLoader գործիք 44](#_Toc513989229)

[Աշխատանքի սկզբունքները 44](#_Toc513989230)

[Պահուստի ստեղծում 46](#_Toc513989231)

[Աղյուսակների ստեղծում 47](#_Toc513989232)

[Աղյուսակների սխեմայի ձևափոխում 48](#_Toc513989233)

[Աղյուսակներում տվյալների ավելացում 49](#_Toc513989234)

[Հրամնանների դաշտ 50](#_Toc513989235)

[Կարգավորումների փաստաթուղթ 51](#_Toc513989236)

[Կարգավորումների փաստաթղթի նկարագրություն 52](#_Toc513989237)

[ՓՈՐՁԱՐԱՐԱԿԱՆ ՏԵԽՆԻԿԱ 57](#_Toc513989238)

[ՓՈՐՁՆԱԿԱՆ ՀԵՏԱԶՈՏՈՒԹՅՈՒՆՆԵՐ 58](#_Toc513989239)

[*SDF* ձևաչափի տվյալների պահուստավորումը *Hive* համակարգում 59](#_Toc513989240)

[Աշխատանքի ընթացքում ուսումնասիրվել են 59](#_Toc513989241)

[Մեթոդ 1 59](#_Toc513989242)

[Մեթոդ 2․ 60](#_Toc513989243)

[Մեթոդ 3․ 62](#_Toc513989244)

[ԵԶՐԱԿԱՑՈՒԹՅՈՒՆ 64](#_Toc513989245)

[ԳՐԱԿԱՆՈՒԹՅԱՆ ՑԱՆԿ 65](#_Toc513989246)

# ՆԿԱՐՆԵՐԻ ՑԱՆԿ

Նկար 1 – Հարաբերական և ոչ հարաբերական բազաների արագագոծության կախումը տվյալների ծավալից։

Նկար 2 – HiveLoader-ի կապը Hive-ի հետ։

Նկար 3 – HiveLoader-ի աշխատանքը։

Նկար 4 – HiveLoader գործիքի օգտագործումը հրամանների դաշտից։

Նկար 5 – Մեթոդ 1-ի աշխատանքի սխեման։

Նկար 6 – Մեթոդ 2-ի աշխատանքի սխեման։

Նկար 7 – Պահեստավորման ձևաչափերի համեմատության գրաֆիլ

Նկար 8 – Մեթոդ 3-ի աշխատանքի սխեման։

Նկար 9 – Մեթոդների համեմատության գրաֆիկ

# ԱՂՅՈՒՍԱԿՆԵՐԻ ՑԱՆԿ

Աղյուսակ 1 – Հարաբերական և ոչ հարաբերական բազաների տարբերությունները։

Աղյուսակ 2 – Hive-ի տվյալների պահեստավորման ձևաչափերը։

Աղյուսակ 3 – Hive, HBase և Redshift համակարգերի տարբերությունները։

Աղյուսակ 4 – HiveLoader գործիքի մուտքային արգումենտների նկարագրություն։

Աղյուսակ 5 – Կարգավորման փաստաթղթի <HiveLoader> տարրի բովանդակություն։

Աղյուսակ 6 – Կարգավորման փաստաթղթի <DB> տարրի բովանդակություն։

Աղյուսակ 7 – Կարգավորման փաստաթղթի <Table> տարրի բովանդակություն

Աղյուսակ 8 – Մեթոդ 1-ի արդյունքները

Աղյուսակ 9 – Մեթոդ 2-ի արդյունքները

Աղյուսակ 10 – Մեթոդ 3-ի արդյունքները

# ՆԵՐԱԾՈՒԹՅՈՒՆ

Մենք ապրում ենք ինֆորմացիոն դարաշրջանում։ Այնքան էլ հեշտ չէ տալ գնահատական թվային ինֆորմացիայի ամբողջ ծավալին, բայց IDC -ի գնահատականներով այն 2006թ․-ի դրությամբ եղել է 0.18 զետտաբայթ, 2011թ․-ին 1.8 զետտաբայթ, 2013թ․ին արդեն 4.4 զետտաբայթ և սպասում է, որ 2020թ․ ին այն կմոտենա 44 զետտաբայթ շեմին1: 1 զետտաբայթը համարժեք է 1021 բայթի և միլիարդ տերաբայթի։  
Մեծ ծավալով տվյալներ գալիս են բազմաթիվ աղբյուրներից։ Օրինակ՝

* Նյու Յորքի բաժնետոմսերի աճուրդը օրական գեներացնում է 4-5 տերաբայթ ինֆոմացիա։
* *Facebook*-ը պահում է ավելի քան 250 միլյարդ նկար, որոնք ամեն ամիս ավելանում են 7 պետաբայթով։
* *The Large Hadror* կոլայդերը, որը տեղակայված է Ժնեվում, Շվեյցարիա, օրական գեներացնում է մոտավորապես 30 պետաբայթ ծավալով տվյալ։
* Վերջին շրջանում, նաև կիսահաղորդչային միկրոսխեմաների ոլորտը մոտեցել է այս ծավալի տվյալների հետ աշխատանքին։

Այսպիսի մեծ ծավալի և տարբեր տեսակի տվյալները առաջացնում են բազմաթիվ խնդիրներ դրանք պահելու և մշակելու համար։ Այս խնդիրների լուծման համար գոյություն ունի «Մեծ տվյալ»-ը, որի մոտեցումները օգնում են հաղթահարել վերը նշված խնդիրները։

1 Վիճակագրկան տվյալները վերձված են այս աշխատանքից՝ [The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things.](https://www.emc.com/leadership/digital-universe/2014iview/index.htm)։

## Հասկացություններ և սահմանումներ

Մեծ ծավալի տվյալների հետ կապված խնդիրներում շատ հաճախ օգտագործվում են մի շարք տերմիններ և սահմանումներ որոնք հեշտացնում են նկարագրել խնդիրը։

**Մեծ Տվյալ** (անգլ․ Big Data) – նկարագրվում է հետևյալ երեք հատկանիշներով՝ էքստրեմալ մեծ ծավալի տվյալներ, տվյալների բազմազան ձևաչափեր և տվյալների մշակման անհրաժեշտ մեծ արագություն։

**Ծավալ** (անգլ․ Volume) - Մեծ տվյալ խնդրի շրջանակներում ասելով տվյալի ծավալ հասկանում ենք, որ այն չափվում է պետաբայթերով, էքսաբայթերով կամ զետաբայթերով։ Առաջացնում է տեխնիկական խնդիրներ ավանդական եղանակներով պահելու, փնտրելու, փոխանցելու, վերլուծության և ցուցադրման համար։

**Արագություն** (անգլ․ Velocity) - Տվյալների գեներացման, մշակաման և թարմացման նկարագիրն է։ Տվյալները փոփոխվում են հաճախակի և պետք է արձագանք տալ անմիջապես։ Ավանդական եղանակները չեն կարող կամ բավականաչափ արագ չեն մշակում տվյալներն իրական ժամանակում արձագանքելու համար։

**Բազմազանություն** (անգլ․ Variety) – Բնութագրում է տվյալների տեսակը և բույթը։  
տվյալներ կարող են լինել ստրուկտուրիզացված, կիսա-ստրուկտուրիզացված և չստրուկտուրիզացված։ Դրանք կարող են լինել ինչպես մարդու կողմից ստեղծված, այպես էլ մեքենաների կողմից գեներացված։

Խնդիրը կարող է համարվել Մեծ Տվյալի խնդիր, եթե բավարարում է վերը նշված բնութագրերին (առնվազն մեկին)։

**Մասշտաբավորում** (անգլ․ Scalability) –Մասշտաբավորում է կոչվումհամակարգի այն հատկությունը, որը թույլ է տալիս ներկայումս առկա տվյալներից կարգով ավելի մեծ արագություն և ծավալ ունեցող մուտքային տվյալները տեղավորել և մշակել առանց զգալի արագագործության կորուստի՝ համակարգում անհրաժեշտ քանակով ռեսուրսներ (հաշվարկային, տվյալների պահպանման, ցանցային և այլն) ավելացնելու հաշվին։

# ԳՐԱԿԱՆՈՒԹՅԱՆ ԱԿՆԱՐԿ

## **Հարաբերական տվյալների բազաներ**

Հարաբերական տվյալների պահուստներում տվյալները պահվում են մեկ կամ մի քանի աղյուսակների և այդ աղյուսակներում տողերի միջոցով։ Այս մեթոդը հիմնվում է հարաբերականության հանրահաշվական տեսության վրա։ Բոլոր հարաբերական բազաների ամենա կարևոր հատկանիշը այն է, որ դրանք բոլորը պահպանում են ACID (Ավտոմատացում (անգլ․ Automicity), Կայունութույն (անգլ․ Consistency), Մեկուսացում (անգլ․ Isolation) և Երկարակեցություն (անգլ․ Durability)) սկզբունքները, որը ապահվում է բոլոր գործողությունների հոսալիորեն կատարվում են։

**Ավտոմատացում**։ Եթե կատարվող գործողությունը չի հաջողվել ավարտել հաջողությամբ, ապա այս հատկությունը ապահվում է ամբողջ համակարգի հետարկվումը այնպես, որ տվյալները մնան անփոփոխ։

**Կայունություն**։ Տվյալների պահուստում գրվող բոլոր տվյալները ենթարկվում են նախապես սահամանված կանոնենրին։

**Մեկուսացում**։ Տվյալ գործողության կատարման ընթացքում կատարվող որևէ փոփոխություն չի ազդում առաջինիս վրա, մինչև դրա հաջող ավարտը։

**Երկարակեցություն**։ Կատարված փոփոխությունները պահպանվում են և մնում են հասանելի հեգատայում, եթե անգամ կատարման ընթացքում տվյալների պահուստն ապահովող մեքենան հոսանքազրկվում է։

Հարաբերական տվյալների պահուստներում տվյալները պահվում են խիստ ստրուկտուրիզացված։ Աղյուսակներում տվյալները պահվում են տողերի և սյուների միջոցով։ Յուրաքանչյուր սյուն ունի տվյալի տիպը։ Ստրուկտուրիզացված հարցումների լեզուն (անգլ․ Structured Query Langulage (SQL)) հարմար է, և օգտագործվում է տվյալնրի պահապանման և հարցման համար։ Հարցումները անգլերեն լեզվով գրված տեքստեր են։ Աղյուսակներն ունեն ֆիքսված քանակի սյուներ, չնայած դրանք կարող են ավելացվել անհրաժեշտության դեպքում։ Աղյուսակները կարող են կապակցված լինել միմյանց հետ՝ այսպիսով ապահովելով տվյալների ամբողջականությունը։ Հարաբերական բազաներ տրամադրում են հետևյալ ընկերությունները՝ *ORACLE, SQL Server, MySQL, PostGreSQL,* և այլն։

## **Ոչ հարաբերական տվյալների բազաներ**

Սրանքնախատեսված են մեծ արագությամբ աճող, չստրուկտուրիզացված տվյալները պահեստավորելու, հեշտությամբ այդ պահեստները ընդլայնելու և արագ հասանելիություն ապահովելու համար։ Այս համակարգերը ապահովում են ճկուն սխեմաների մեխանիզմ տվյակների պահման համար, այսպիսով հրաժարվելով այսպես կոչված «Հղումային ամբողջականություն» (*“Referential Integrity”*) սկզբունքից, որը բնորոշ է հարաբերական բազաներին։ Սրանք հետևվում են *CAP* սկզբունքներին, որոնք են՝ Կայունություն (անգլ․Consistency), Հասանելիություն (անգլ․ Availability) և Բաշխում (անգլ․ Partitioning)։ Ի տարբերություն *ACID* համակարգերի, այս երեք սկզբուքներից միայն երկուսն են մշտապես պահպանվում․ ոչ-հարաբերական տվյալների բազաները պահպանում են միայն «հիմանականը»՝ ծրագրերի հիմնական վիճակները և վերջնական կայուն վիճակը։ Այս տեսակի բազաների օրիանկներ են՝ *BigTable (Google), HBase (Yahoo), Cassandra (Facebook),* և այլն։

**Ոչ հարաբերական տվյալների բազաների տեսակները։** Սրանք բաժանվում են չորս հիմնական խմբերի, դրանք են՝

* Բանալի-արժեք տվյալների բազա (անգլ․ key-value database)
* Սյունային տվյալների բազա (անգլ․ column datebase)
* Փաստաթղթային տվյալների բազա (անգլ․ document database)
* Գրաֆային տվյալների բազա (անգլ․ graph database)։

**Բանալի-արժեք տվյալների բազաներ։** Սա *NoSQL* տվյալների բազաների պարզագույն տեսակ է, որտեղ յուրաքանչյուր արժեք ասոցացվում է մեկ բանալու հետ։

Օրինակ՝ *Redis*։

**Սյունային տվյալների բազաներ։** Այսպիսի բազաները ի վիճակի են պահեստավորել և մշակել մեծ քանակությամբ տվյալներ, օգտագործելով ցուցիչ, որը հղվում է կլաստերի վրա բաշխված բազմաթիվ սյունակներին։

Օրինակ՝ *HBase, Hive:*

**Փաստաթղթային տվյալների բազաներ։** Կարող են պարունակել բազմաթիվ առանցքային փաստաթղթեր մի քանի ներդրված մակարդակներով։ Փաստաթղթերը կարող են պահվել օրինակ *JSON* ձևաչափով։

Օրինակ՝ *MongoDB*:

**Գրաֆային տվյալների բազաներ։** Ավանդական տողերի և սյուների փոխարեն, այս բազաները օգտագործում են հանգույցներ (անգլ․ nodes) և կողեր (անգլ․ edges) գրաֆի ստրուկտուրան նկարագրելու և տվյալները պահեստավորելու համար։

Օրինակ ՝ *Neo4J*։

Ուղղահայաց մասշտաբավորում

Սա մեթոդ է, որի միջոցով մեծացվում է համակարգի հզորությունը, նրան ավելացվելով լրացուցիչ հաշվողական և հիշողության սարքեր։ Բոլոր հարաբերական տվյալների բազաները ապահովում են ուղղահայաց մասշտաբավորում։ Այս մասշտաբավորման մոտեցումը ենթադրում է, որ համակարգը պարունակում է միայն մեկ սերվեր մեքնա, որը կապված է բոլոր օգտագործողների հետ։ Կլաստերային համակարգերը ստեղծվում են հանգույցներից՝ կրկնելով տվյալները բոլոր հանգույցներում։ *ACID*-ի սկզբունքները պահպանելու համար բոլոր հանգույցները պետք է ունենան նույն տվյալները, որի պատճառով տվյալների սինխրոնիզացիան դառնում է բավականին բարդ պրոցես։

Մասշտաբավորման այս մեթոդի առավելությունը կայանում է նրանում, որ եթե գործող սերվերը ինչ-որ պատճառով շարքից դուրս է եկել, ապա այն կարող է հեշտությամբ փոխարինվել միևնույն կլաստերին պատկանող մեկ այլ հանգույցով, քանի որ համակարգի բոլոր հանգույցները կրկնում են միևնույն տվյալները։ Այսպիս համակարգերը հաճախ կոչում են անսխալ կլաստերներ (անգլ․ Fail-over clusters)։

## Հորիզոնական մասշտաբավորում

Այս մեթոդի ժամանակ արագ եկող տվյալները պահեստավորելու համար, ցանցին ավելացվում են նոր ռեսուրսներ (հաշվարկային և տվյալների կրիչ սարքեր)։ Բոլոր ոչ հարաբերական բազաները ապահովում են հորիզոնական մասշտաբավորում։ Դժվար չէ կլաստերի մեջ ավելացնել նոր հանգույցներ՝ տվյալների աճը սպասարկելու համար։ Տվյալները ավտոմատ կերպով բաժանվում են կլաստերի հանգույցների մեջ և մշակվում այնտեղ՝ ապահովելով նաև ուղղահայաց մասշտաբավորում (տվյալների հասանելիությունն ապահովելուվ միանգամից մի քանի հանգույցից)։ Սա բաշխված տվյալների համակարգ է։ Հադուպի բաշխված ֆայլային համակարգը (անգլ․ Hadoop Distributed File System (HDFS)) սրա կլասիկ օրինակ է։

Այս տեխնիկայի առավելությունը կայանում է ըստ հանգույցների տվյալների բաշխման և կրկնօրինակման մեջ՝ ապահովելով բարձր արագություն և հուսալիություն տվյալների զուգահեռ հարցման դեպքում։ Այս մեթոդը հարկապես լավ է աշխատում, երբ տարբեր հանգույցներում գտնվող տվյալների միջև կապերի (անգլ․ joins) անհրաժեշություն չկա և թույլ է տալիս տվյալները պահել աշխարհագական տարբեր վայրերում։

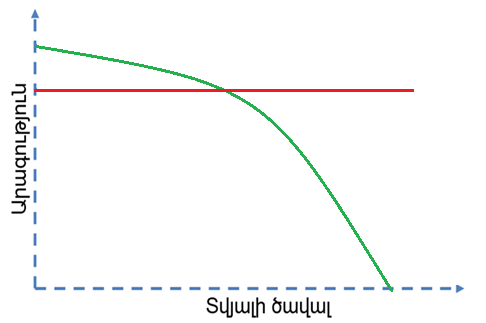
## **Հարաբերական և ոչ հարաբերական տվյալների բազաների համեմատություն**

Տվյալները պահեստավորելու և դրանք մշակելու համար կան գրականությունից հայտնի երկու մոտեցումներ, դրանք են՝

1. Հարաբերական տվյալների բազաներ (անգլ․ relational databases)
2. Ոչ հարաբերական տվյալների բազաներ (անգլ․none-relational databases)։

Աղ. 1 Հարաբերական և ոչ հարաբերական բազաների տարբերությունները։

|  |  |  |
| --- | --- | --- |
|  | **Հարաբերական բազաներ** | **Ոչ հարաբերական բազաներ** |
| **Տվյալի ծավալ** | Գիգաբայթ | Պետաբայթ |
| **Հասանելիություն** | Ինտերակտիվ և խմբաքանակով | Խմբաքանակով |
| **Թարմացումներ** | Գրել կարդալ բազմակի անգամ | Գրել մեկ անգամ, կարդալ բազմիցս |
| **Կառուցվածք** | Ստատիկ | Դինամիկ |
| **Ամբողջականություն** | Բարձր | Ցածր |
| **Մասշտաբավորում** | Ոչ գծային | Գծային |



Հարաբերական բազաներ

Ոչ հարաբերական բազաներ

Նկ․ 1 Հարաբերական և ոչ հարաբերական բազաների արագագոծության կախումը տվյալների ծավալից։

## Hadoop համակարգ

*Apache Hadoop* ծրագրային գրադարանը հանդիսանում է ամբողջական համակարգ, որը թույլ է տալիս մեծ տվյալների հետ կատարվող աշխատանքը բաշխել համակարգչային կլաստերների միջև՝ օգտագործելով ծրագրային պարզ մոդելներ։ Այն նախագծված է այնպես, որ կարողանա մասշտաբավորվել մեկ մեքենայից մինչև հազարավոր մեքենաների, որոնցից յուրաքանչյուրն ապահովում է լրացուցիչ հաշվողական հզորություն և հիշողություն։ Բացի վերոհիշյալից, Hadoop համակարգը կարողանում է նաև ավտոմատ կերպով հայտանբերել ծրագրային խափանումները հանգույցներում, ուղարկել ծանուցումներ համակարգի վիճակի մասին և կատարում է տվյալների վերաբաշխում առկա անխափան հանգույցների վրա՝ այսպիսով ապահովելով բարձր հասանելիություն տվյալ կլաստերից ։

*Hadoop*-ը բաղկացած է հետևյալ հիմնական մոդուլներից՝

* **Hadoop Common** – Ընդհանուր օգնական ծրագրեր, որոնք օգնում են Hadoop-ի մյուս մոդուլներին,
* **Hadoop Distributed File System (HDFS)** – Բաշխված ֆայլային համակարգ է, որը ապահովում է բարձր թողունակությամ հասանելիթույն ծրագրերի տվյալներին,
* **Hadoop YARN** – Աշխատանքների պլանավորման և կլաստերի ռեսուրսների կառավարման համակարգ է,
* **Hadoop MapReduce –** *YARN*-ի վրա հիմնված համակարգ է մեծ տվյալների զուգահեռ մշակման համար։

Hadoop-ի հետ կապված այլ նախագծերի օրինակներ են՝

* **Ambari –** *WEB* պորտալ է *HDFS, Hadoop MapReduce, Hive, HBase,* և այլ գործիքերի կառավածման և մոնիթորինգի համար,
* **HBase –** Մասշտաբավորվող տվյալների պահուստ է, որը նախատեսված է մեծ աղյուսակների պահաման համար,
* **Hive –** Տվյալների պահեստավորման և *SQL* հարցումներ իրականացման ենթակառուցվածք է։

### Hive համակարգ

**Apache Hive** -ը տվյալների պահուստի ծրագիր է, որը հեշտացնում է մեծ քանակի տվյալների ընթերցումը և գրառումը աղյուսակային բաշխված պահեստներում։ Տվյալների հասանելիությունն ապահովվում է *SQL* հրմանների և *JDBC* դրայվերի միջոցով։

### Տվյալների տիպեր

*Hive* -ը սապասարկում է հետևյալ տվյալների տիպերը՝

**Թվային տիպեր (Numeric types)**

* **TINYINT**  
  1 բայթ չափսի, ամբողջ թիվ, [-128, 127] միջակայքում
* **SMALLINT**  
  2 բայթ չափսի, ամբողջ թիվ, [-32,768, 32,767] միջակայքում
* **INT / INTEGER**  
  4 բայթ չափսի, ամբողջ թիվ, [-2,147,483,648, 2,147,483,647] միջակայքում
* **BIGINT**  
  8 բայթ չափսի, ամբողջ թիվ, [-9,223,372,036,854,775,808, 9,223,372,036,854,775,807] միջակայքում
* **FLOAT**4 բայթ չափսի, լողացող կետով թիվ
* **DOUBLE / DOUBLE PRECISION**  
  8 բայթ չափսի, կրկնակի ճշգրտությամ լողացող կետով թիվ
* **DECIMAL / NUMERIC**  
  38 թվանշանի ճշտությամբ ամբողջ թիվ

**Ամսաթիվ / Ժամ տիպեր (Date/ Time types)**

* **TIMESTAMP**

Այս տիպը իրենից ներկայացնում է *UNIX* համակարգի ժամանակցույցի համարժեքը Hive-ում։ Այն կարող է ունենալ նանովայրկյանային ճշտություն։ Այս ժամանակացույցները տեքստային ֆայլերում ներկայացվում են հետևյալ տեսքով՝   
yyyy-mm-dd hh:mm:ss[.f...]։

* **DATE**

Նկարագրում է որոշակի տարի/ամիս/օր հատևյալ ձևաչափում ՝  
YYYY-MM-DD։ Օրինակ՝ DATE '2018-01-01'։ Այս տիպը չունի օրվա ժամ բաղադրիչ։

* **INTERVAL**INTERVAL տիպը նկարագրում է որոշակի ժամանակահատված, օրինակներ՝
  + INTERVAL '1' DAY – 1 օրյա ժամանակահատված
  + INTERVAL '1-2' YEAR TO MONTH – 1 տարի + 2 ամիս ժամանակահատված

**Տողային տիպեր (String types)**

* **STRING**

Տողային տառատեսակները կարող են ներկայացվել կամ միայնակ մեջբերումներով (‘) կամ երկակի մեջբերումներով (“)։ *Hive*-ը C ծրագրավորման լեզվի տողերի նկարագրությունը**։**

* **VARCHAR**

Այս տիպը օգտագործվում է երկարության նշումով (1 ից մինչև 65535), որը սահմանում է սիմվոլների առավելագույ քանկը մեկ տողային տիպի օբյեկտում։

* **CHAR**

Այս տիպը նման է VARCHAR-ին բայց այն ունի հաստատուն երկարություն։

**Այլ տիպեր (Misc types)**

* **BOOLEAN**
* **BINARY**

**Բարդ տիպեր (Complex types)**

* **ARRAY<data\_type>** (Զանգվածներ)
* **MAP<primitive\_type, data\_type>** (Ասոցացիաներ)
* **STRUCT<col\_name: data\_type [COMMENT col\_comment], ...>** (Ստրուկտուրաներ)
* **UNIONTYPE<data\_type, data\_type, ...>** (Միավորումներ)

### SQL հրամաններ

Ստորև բերված են *Hive* -ի կողմից ապահովվող *SQL* հրամններից մի քանիսը, որոնք օգտագործվել են աշխատանքի ընթացքում ՝

**Պահեստի ստեղծում – Create Database**

CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database\_name  
  [COMMENT database\_comment]  
 [LOCATION hdfs\_path]  
 [WITH DBPROPERTIES (property\_name=property\_value, ...)];

**Աղյուսակի ստեղծում – Create Table**

CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db\_name.] table\_name [(col\_name data\_type [COMMENT col\_comment], ... [constraint\_specification])]  
  [COMMENT table\_comment]  
  [PARTITIONED BY (col\_name data\_type [COMMENT col\_comment], ...)]  
  [CLUSTERED BY (col\_name, col\_name, ...) [SORTED BY (col\_name [ASC|DESC], ...)] INTO num\_buckets BUCKETS]  
  [SKEWED BY (col\_name, col\_name)]  
     ON ((col\_value, col\_value, ...), (col\_value, col\_value, ...), ...)  
     [STORED AS DIRECTORIES]  
  [[ROW FORMAT row\_format]  
   [STORED AS file\_format]  
     | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (...)]]  
  [LOCATION hdfs\_path]  
  [TBLPROPERTIES (property\_name=property\_value, ...)]  
  [AS select\_statement];

**Պահեստավորման ձևաչափեր**

*Hive*-ը ապահովում է ներդրված և մասնավոր ստեղծված ֆայլային ձևաչափեր։ Աղյուսակ 2-ում ցուցադրված են ներդրված տիպերից մի քանիսը։

Աղ. 2 Hive-ի տվյալների պահեստավորման ձևաչափերը։

|  |  |
| --- | --- |
| **Պահեստավորման ձևաչափ** | **Նկարագրություն** |
| STORED AS TEXTFILE | Պահվում է որպես պարզ տեքստային ֆայլ, ներդրված ձևաչափ է |
| STORED AS SEQUENCEFILE | Պահվում է որպես սեղմված հաջորդականության ֆայլ |
| STORED AS ORC | Պահվում է ORC ձևաչափի ֆայլ, ապահովում է ACID գործարքներ։ Գրանցում է մետատվյալներ ըստ սյուների |
| STORED AS PARQUET | Պահվում է PARQUET-ի սյունային ձևաչափով |
| STORED AS AVRO | Պահվում է AVRO ձևաչափով |
| STORED AS RCFILE | Պահվում է որպես RCFile տվյալների սյնունային ֆայլ |
| STORED BY | Պահվում է ոչ ներքին աղյուսակային ձևաչափով |

**Աղ. 2 -ի շարունակություն։**

**Մասնատված աղյուսակներ**

Մասնատված աղյուսակներ կարելի է ստեղծել օգտագործելով PARTITIONED BY կետը աղյուսկներ ստեղծող հրամանի մեջ։ Աղյուսակը կարող է ունենալ մեկ և ավելի մասնատված սյուներ, և որոնցից յուրաքնչյուր արժեքների համադրության համար սյան մեջ ստեղծվում է առանձին պանակ։ Հետագայում աղյուսակները կամ նրա կտորները կարող են միավորվել օգտագործելով CLUSTERED BY «սյուներ» կետով, ինչպես նաև տվյալները կարող են տեսակավորվել SORT BY «սյուներ» կետի միջոցով։ Սա կարող է էապես լավարկել որոշակի տեսակի հարցումների արագագործությունը։

Օրինակ՝

CREATE TABLE page\_view (viewTime INT, userid BIGINT, page\_url STRING,

referrer\_url STRING, ip STRING COMMENT 'IP Address of the User')

  PARTITIONED BY (dt STRING, country STRING);

**Load - Ֆայլերի ներբեռնում աղյուսակների մեջ**

*Hive*-ը չի կատարում տվյալների որևէ ձևափոխություն դրանց աղյուսակների ներբեռնամն ժամանակ։ Ներբեռնում գործողությունը կատարվում է պարզ պատճենահանման (կամ տեղափոխման) գործողությամբ, որի արդյունքում տվյալները տեղափոխվում են Hive-ի աղյուսակների համապատասխան վանդակներ։

LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename [PARTITION (partcol1=val1, partcol2=val2 ...)]

**Insert - Տվյալների մուտքագրում աղյուսակների մեջ**

Տվյալները աղյուսակ մուտքագրելու համարօգտագործվում է **Insert** հրամանը։

INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...) [IF NOT EXISTS]] select\_statement1 FROM from\_statement;

INSERT INTO TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select\_statement1 FROM from\_statement;

INSERT OVERWRITE հրամանը կվերագրանցի աղյուսակում կամ դրա մասերում առկա տվյալները, եթե հրամանի մեջ առկա չէ IF NOT EXISTS կետը։

INSERT INTO հրամանը կավելացնի տվյալներ աղյուսակում կամ դրա մասերում՝ պահելով արդեն գոյություն ունեցողները։

Հնարավոր է նաև մի քանի գրառումների միաժամանակ ավելացում աղյուսակի մեջ հետրյալ հրամանների միջոցով։

Օրինակ՝

CREATE TABLE pageviews (userid VARCHAR(64), link STRING, came\_from STRING)

  PARTITIONED BY (datestamp STRING) CLUSTERED BY (userid) INTO 256 BUCKETS STORED AS ORC;

INSERT INTO TABLE pageviews PARTITION (datestamp = '2014-09-23')

  VALUES ('jsmith', 'mail.com', 'sports.com'), ('jdoe', 'mail.com', null);

## MapReduce

*MapReduce*-ը *Google*-ի կողմից նախագծված ծրագրավորման մոդել է, որը կատարում է մեծ տվյալների զուգահեռ մշակում համակարգիչների կլաստերների վրա։

Այս ալգորիթմը բաղկացած է երկու քայլերից՝ դրանք են ***Map***-ը և ***Reduce***-ը։

**Map** քայլով կատարվում է մուտքային տվյալների նախնական մշակում՝ գլխավոր հանգույցը (անգլ․ master node) ստանում է մուտքային տվյալները, բաժանում այն մասերի, և փոխանցում դրանք մնացած աշխատող հանգույց ներին (անգլ․ worker node)։

**Reduce** քայլի ժամանակ տեղի է ունենում արդեն մշակված տվյալների համաձուլում։ Գլխավոր հանգույցը ստանում է արդյունքները աշխատող հանգույցներից և դրանց հիման վրա կազմավորում է նախապես տրված առաջադրանքի արդյունքը։

*MapReduce*-ի առավելությունը կայանում է նրանում, որ այն թույլ է տալիս բաշխված կերպով կատարել մշակման և ձուլման աշխատանքները։ Նախնական մշակաման օպերացիաները կատարվում են միմյանցից անկախ և կարող են աշխատել միմյանց զուգահեռ (իրականում այն կախված է մուտքային տվյալների աղբյուրից և օգտագործվող պրոցեսորների քանակից)։ Նմանապես ձուլման աշխատանքը կարող է կատարվել բազմաթիվ աշխատող հանգույցների միջոցով՝ դրա համար միայն անհրաժեշտ է, որ նախնական մշակման այն արդյունքները որոնք ունեն միևնույն բանալին մշակվեն մեկ աշխատող հանգույցի միջոցով ժամանակի միևնույն պահին։ Այսպիսով *MapReduce*-ը կարող է օգտագործվել պետաբայթ ծավալ ունեցող տվյալների տեսակավորման (անգլ․ sorting) համար, դրա համար ծախսելով ընդամենը մի քանի ժամ։ Զուգահեռ կատարումը տալիս է նաև վերականգնվելու հնարավորություն, երբ տեղի է ունեցում որոշ սերվերների մասնակի խափանումներ․ եթե հանգույցներից մեկում աշխատանքի որևէ փուլում տեղի է ունենում խափանում, ապա նրա կատարվելիք աշխատանքը փոխանցվում է մեկ այլ հանգույցի։

*MapReduce* մոդելի կիրառման պրակտիկ օրինակ․

**Գտնել ընդհանուր ընկերներ։** Սա իրական օրինակ է, որը կիրառվում է Facebook ընկերության կողմից՝ երկու օգտատերի ընդհանուր ընկերների ցուցակի ստացման հարցման ժամանակը նվազացնելու համար։ Այսպիսով ինչպես հայտնի է Facebook-ը ունի ընկերների ցուցակ գաղափար (ընկերները այստեղ երկկողմանի են՝ այսինք եթե մի օգտատերը հանդիսանում է մյուսի ընկերը, ապա հակառակ պնդումը նույնպես ճիշտ է)։ *Facebook*-ն ունի բավականին շատ սկավառակային հիշողության տարածք և ամեն օր նրանք մշակում են հարյուր-միլիոնավոր հարցումներ։ Բավականին տարածված հարցում է հանդիսանում ընդհանուր ընկերների ցուցակի ցուցադրումը։ Այս ցուցակը փոխվում է ոչ հաճախ, և անիմաստ է ամեն անգամ հաշվարկներ կատարել այդ ցուցակը դուրս բերելու համար։ Այս խնդիրը կարելի է լուծել *MapReduce*-ի միջոցով օրական մեկ անգամ հաշվելով ընդհանուր ընկերներին և պահելով այդ արդյունքները հիշողության սկավառակների վրա՝ հետագայում կատարելով միայն պարզագույն փնտրման աշխատանք։ Ենթադրենք որ ընկերները պահպանվում են [Անձ] -> [Ընկերների ցուցակ] տեսքով:

A -> B C D  
B -> A C D E  
C -> A B D E  
D -> A B C E  
E -> B C D

Վերևում բերված յուրաքանչյուր տողը կլինի Map ֆունկցիայի մուտքային արժեք։ Ընկերներ ցուցակից յուրաքանչյուր ընկերոջ համար Map -ը դուրս կբերի բանալի-արժեք զույգ։ Բանալին կլինի ընկերը անձի հետ։ Արժեքը կլինի ընկերների ցուցակը։ Բանալին կպահվի տեսակավորված (անգլ․ sorted), որպեսզի բոլոր զույգերը հնարավոր լինի մշակել միևնույն Reducer ֆունկցիայի միջոցով։ Բոլոր Map-երի աշխատանքից հետո կունենանք հետեևյալ արդյունքները՝

map (A -> B C D)  
 (A B) -> B C D  
 (A C) -> B C D  
 (A D) -> B C D

map (B -> A C D E) (A-ն դրվում է B-ից առաջ)  
(A B) -> A C D E  
(B C) -> A C D E  
(B D) -> A C D E  
(B E) -> A C D E

map (C -> A B D E)  
 (A C) -> A B D E  
 (B C) -> A B D E  
 (C D) -> A B D E  
 (C E) -> A B D E

map (D -> A B C E)  
 (A D) -> A B C E  
 (B D) -> A B C E  
 (C D) -> A B C E  
 (D E) -> A B C E

map (E -> B C D)  
 (B E) -> B C D  
 (C E) -> B C D  
 (D E) -> B C D

Բանալի-արժեք զույգերը Reduce-ներին ուղարկելուց առաջ անհրաժեշտ է կատարել բանալիների խմբավորում՝

(A B) -> (A C D E) (B C D)

(A C) -> (A B D E) (B C D)

(A D) -> (A B C E) (B C D)

(B C) -> (A B D E) (A C D E)

(B D) -> (A B C E) (A C D E)

(B E) -> (A C D E) (B C D)

(C D) -> (A B C E) (A B D E)

(C E) -> (A B D E) (B C D)

(D E) -> (A B C E) (B C D)

Յուրաքանչյուր տող հանդիսանում է Reduce ֆունկցիային համար մուտքային արժեք։ Reduce ֆունկցիան պարզապես կատարում է արժեքների ցուցակների հատումներ՝ որպես ելքային արժեք տալով բանալին և իր արժեքների ցուցակների հատումը։ Օրինակ՝ reduce((A B) -> (A C D E) (B C D)) -ը ելքում կտա (A B) -> (C D) ՝ որը նշանակում է, որ A և B ընկերները ունեն C և D ընդհանուր ընկերներ։

Բոլոր Reduce-ների արդյունքները կլինեն ՝

(A B) -> (C D)

(A C) -> (B D)

(A D) -> (B C)

(B C) -> (A D E)

(B D) -> (A C E)

(B E) -> (C D)

(C D) -> (A B E)

(C E) -> (B D)

(D E) -> (B C)

Այսպսիսով, երբ D-ն այցելի B օգտատիրոջ էջը, մենք կկարողանանք փնտրել (B D)-ն և կտեսնենք, որ (A C E)-ն նրանց ընդհանուր ընկերներն են։

## Ընդարձակ նշման լեզու (XML)

Այս լեզուն սահմանում է մի շարք կանոներ՝ փաստաթղթերը մարդու և համակարգչի համար ընթեռնելի լեզվով կոդավորելու համար։ Այս կանոնները սահմանված են *XML* 1.0-ում։

Սույն աշխատանքում օգտագործված բոլոր կարգավորման փաստաթղթերը սահմանված են ընդարձակ նշման լեզվով։ Ստորև բերված է *XML* լեզվի կարճ նկարագրությունը, ինչը բացատրում է կարգավորման փաստաթղթերի ձևաչափի ընտրությունը։

*XML*-ի նախագծման հիմնական նպատականերն են պարզությունըն ու ընթեռնելիությունը, ընդհանրությունը և համացանցում օգտագործման հեշտությունը։ *XML*-ը տեքստային ինֆորմացիայի ձևաչափ է՝ Յունիկոդի ուժեղ աջակցությամբ։ Չնայած, որ *XML*-ի դիզայնը հիմնականում կենտրոնանում է փաստաթղթերի վրա, այն շատ է օգտագործվում կամայական տվյալների կառույցների ներայացման համար, օրինակ՝ համացանցում, կամ ինչպես այն արվել է HiveLoader գործիքի համար։

Բազմաթիվ ծրագրերի ծրագրավորման ինտերֆեյսը զարգացվել են *XML* տեղեկատվության մշակմանը օգնելու համար։ 2009-ից սկսած փաստաթղթերի հարյուրավոր ձևաչափեր են ստեղծվել, օրինակ՝ *RSS*, *Atom*, *SOAP* և *XHTML*։ Հետագայում *XML*-ի վրա հիմնվածները դարձան հիմնականը բազմաթիվ գրասենյակա-արտադրողականային գործիքների համար, ներառելով *Microsoft* *Office*-ը (Office Open *XML*), *OpenOffice*-ը, *LibreOffice*-ը (*OpenDocument*) և *Apple*-ի *iWork*-ը։ *XML* լեզուն լայնորեն օգտագործվում է նաև կոմունիկացիոն պրոտոկոլների համար, ինչպիսին է, օրինակ, *XMPP* -ն։

*XML* փաստաթուղթը, ըստ սահմանման, նշանների շարք է։ Յունիկոդի գրեթե բոլոր նշանները կարող են օգտագործվել *XML* փաստաթղթի մեջ։

**Նշում և բովանդակություն**

Նիշերը, որոնք պարունակում է *XML* փաստաթուղթը, լինում են նշման և բովանդակային և կարող են տարբերակվել սովորական սինտակտային կանոններով ծրագրով։ Հիմնականում, շարքերը, որոնք ստեղծում են նշում, սկսվում են «<» նիշով և վերջանում «>» նիշով, կամ սկսվում «&» նիշով և վերջանում «;» նիշով։ Նիշերի այն շարքերը, որոնք նշման չեն՝ բովանդակային են։

**Հայտանիշ (անգլ.՝ tag)**

Նշումներըման կառույցները, որոնք սկսվում են «<» և վերջանում են «>» կոչվում են հայտանիշեր։ Հայտանիշները լինում են 3 տեսակ՝

* + Սկսման հայտանիշեր։ Օրինակ՝ *<section>*
  + Վերջացնելու հայտանիշեր։ Օրինակ՝ *</section>*
  + Դատարկ-տարրային հայտանիշեր։ Օրինակ՝ *<line-break />*

**Տարր (անգլ․՝ element)**

Փաստաթղթի այն մասը, որը կա'մ սկսվում է սկսման հայտանիշով և վերջանում է վերջացնելու հայտանիշով, կա'մ էլ պարունակում է միայն դատարկ-տարրային հայտանիշեր, կոչվում է տարր։ Սկսման և վերջացնելու հայտանիշերի միջև գտնվող նիշերը, եթե առկա են, կոչվում են տարրերի բովանդակություն, և կարող են պարունակել նշում, ներառյալ այլ տարրեր, որոնք կոչվում են երեխա տարրեր։ Տարրի օրինակ է՝ *<Greeting>Hello, world.</Greeting>,* կամ՝ *<line-break />*։

**Բնորոշիչ (անգլ․՝ attribute)**

Նշման կառույցները, որոնք պարունակում են անուն/արժեք զույգ, որը ունի սկսման և դատարկ-տարրային հայտանիշ, կոչվում են բնորոշիչներ։ Ստորև ներկայացված օրինակում img տարրը ունի երկու բնորոշիչ՝ src և alt։

*<img src="madonna.jpg" alt='Foligno Madonna, by Raphael'/>*

Մեկ այլ օրինակ, որտեղ բնորոշիչի անունը line է, իսկ արժեքը 5։

*<data line=”5”>Data of the line five</data>*

*XML*-ի բնորոշիչը կարող է ունենալ մեկ արժեք և ամեն բնորոշիչը կրող է հայտնվել ամենաշատը մեկ անգամ ամեն տարրում։

**XML-ի հայտարարումը**

*XML* փաստաթղթերը կարող են սկսվել իրենց մասին որոշ տեղեկատվությամբ, ինչպես հետևալ օրինակում՝

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

**Մեկնաբանություններ (անգլ․՝ comments)**

Մեկնաբանությունները կարող են հայտնվել ամեն տեղ փաստաթղթում՝ նշումից դուրս։ Մեկնաբանությունները չեն կարող հայտնվել *XML* հայտարարւթյունից առաջ։ Մեկնաբանությունները սկսվում են «<!—» և վերջանում «-->»։, ինչպես նաև կրկնակի «--» շարքը չի թույլատրվում մեկնաբանությունների ներսում։ Ամպերսանդը չունի ինչ-որ հատուկ նշանակություն մեկնաբանությունների համար։ Մեկնաբանության օրինակ՝

*<!—Some comment can be here -->*

# ԽՆԴՐԻ ԴՐՎԱԾՔ

«Մեծ տվյալ» լուծման կիրառումը կիսահաղորդչային տվյալների համար:

Տվյալների պահուստավորման արագության կախվածությունը՝

* + տվյալների քանակից
  + տվյալների տեսակից
  + փոփոխման արագությունից:

# ՏԵՍԱԿԱՆ ԱՌՆՉՈՒԹՅՈՒՆՆԵՐ

## Կիսահաղորդչային թեստավորման տվյալների ձևաչափեր

**Standart Test Data Format (STDF)** – Ստանդարտ թեստավորման տվյալների ձևաչափ է, որը մասնավորապես օգտագործվում է կիսահաղորդիչների թեստավորման ինֆորմացիայի պահման համար, ի սկզբանե այն մշակվել է Teradyn-ի կողմից, բայց այժմ այն փաստացի հանդիսանում է ստանդարտ ձևաչափ այդ ոլորդի համար և լայնորեն կիրառվում է LTX-Credence, Roos Instruments, Teradyne, Advantest և այլ ընկերությունների ավտոմատացված թեստավորման համակարգերի կողմից (անգլ․ automated test equipment (ATE))։

**KLA-Tencor Results File (KLARF)** – Օգտագործվում է ֆիզիկական թերությունների ստուգման (անգլ․ inspection) և վերանայման (անգլ․ review) տվյալների պահման համար։

## Տվյալների պահուստի ընտրություն

Կիսահաղորդչային տվյալների պահուստավորման խնդրի լուծման համար ընտրվել է մեծ տվյալներ լուծումը, քանի որ, այս տեսակի տվյալները բավարարում են մեծ տվյալ կոչվելու համար անհրաժեշ երեք հիմնական պահանջներին, ինչպես նկարագրված է ներածություն բաժնում՝

* տվյալները սովորաբար ներկայացվում են բազմաթիվ ձևաչափերով (օրինակ՝ STDF, KLARF, և այլն),
* գեներացվում են տարբեր աղբյուրների կողմից (օրինակ՝ ծրագրեր, գործիքներ կամ մարդիկ) ,
* վերջին ժամանակաշրջանում տեխնոլոգիաների զարգացումը բերեց այս տվյալների ծավալների կտրուկ աճի։

Ինչես փորձը ցույց տվեց ավանդական հարաբերական տվյալները բավականաչափ լավ չեն հաղթահարում այս տվյալների մշակուման և պահեստավորուման խնդիրնեը և այդ պատճառով տվյալ աշխատանքի շրջանակներում որոշվել է օգտագործել ոչ հարաբերական տվյալների բազաներ։ Քանի որ վերը նշված տվյալները սովորաբար հարմար է լինում պահեստավորել և ցուցադրել աղյուսկային տեսքով՝ աշխատանքի հետագա կատարման համար դիտարկվել են հետևյալ երեք լայնապես օգտագործվող պահեստների տեսակները՝

* HBase (Apache Hadoop),
* Hive (Apache Hadoop),
* Redshift (Amazon)։

Աղյուսակ 3-ում ցուցադրված է նշված երեք տարբերակների մոտեցումները և դրանց տարբերությունները։

Աղ․ 3 Hive, Hbase և Redshift համակարգերի տարբերությունները։

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Hive** | **HBase** | **Redshift** |
| **Ծավալային սահմանափակում** | Անսահման | Անսահման | 16TB \* 100 հանգույցներ (nodes) |
| **Ծառայության արժեքը AWS-ում** | Ավելի քան  900$/TB/տարի | Ավելի քան  900$/TB/տարի | Ավելի քան  1000$/TB/տարի |
| **Հնարավոր է տեղադրել լոկալ** | Այո | Այո | Ոչ |
| **SQL** | Մասնակի սպասարկում | Ոչ | Այո |
| **Հասանելիության մեթոդներ** | JDBC, ODBC, Thrift | Ներքին API | JDBC, ODBC |
| **Սպասակվող ծրագրավորման լեզուներ** | C++, Java, PHP, Python | Java | Բոլոր լեզուները որոնք սպասարկում են JDBC/ODBC |
| **Ներմուծվող տվյալների ձևաչափեր** | CSV, Text, RC, ORC… | Չուսումնասիրված | CSV, Text |
| **Տվյալների ինտեգրացիա (Տվյալների աղբյւրներ)** | Լոկալ հիշողություն, Հարաբերական տվյալների պահուստ, Cloud | Չուսումնասիրված | Amazon S3 կամ Dynamo DB |

Աղ․3-ի շարունակություն

Աղյուակի վրա հիմնվելով՝ որպես տվյալների պահուստ ընտրվել է **Hive**-ը, քանի որ այն հասանելի է անվճար, տրամադրում է C++, Java և այլ լեզուներով աշխատելու հնարավորություն և ապահվովում է SQL հարցումների լեզվով աղյուսակային տվյալների հետ աշխատելու միջոց։ Մեր դեպքում տվյալները պահպանվում են SDF աղյուսակային ձևաչափով, ինչը բավականին կարևոր հանգամանք է իրական կիրառելիություն ապահովելու համար։

## Hive գործիքի ներդրված մեթոդներ

Նախագծվել և իրագործվել են *Hive* գործիքի ներդրված մեթոդներ, որոնք *Hive*-ին թույլ են տալիս աշխատել *Hadoop* համակարգում առկա *SDF* ձևաչափի տվյալների հետ՝ գրել, կարդալ և փոփոխել այն որպես աղյուսակային տվյալ։ Մեթոդները իրականցվել են *Java* ծրագրավորման լեզվով։ *c\_sdf\_wrapper* C-ական գրադարանը *Java* լեզվով գրված մեթոդների հետ կապելու համար օգտագործվել է *Java Native Access (JNA)* գրադարանը։ Այն հնաևավորություն է տալիս Java լեզվով գրված ծրագրերը հեշտությամբ օգտվել հարազատ (native) համօգտագործման ընդհանուր (shared) գրադարաններից` առանց Java Native Interface*(JNI)*-ի օգտագործման:

*Hive*-ն օգտագործում է *Hadoop* համակարգի գրադարանների կողմից տրամադրված մի շարք ինտերֆեյսներ, դասեր և մեթոդներ տվյալների հետ աշխատելու համար։ *SDF* ձևաչափի տվյալները *Hive*-ում տեսանելի դարձնելու և նրա հետ համակարգի աշխատանքը հնարավոր դարձնելու համար ժառանգվել և իրագործվել են անհրաժեշտ ինտերֆեյսներ, դասեր և մեթոդներ։ Ստորեվ ներկայացված են այդ մեթոդներից երկու հիմնականները։

### SDFReader ներդրված մեթոդ

package com.SDFReader;

import java.io.IOException;

import org.apache.hadoop.io.BytesWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapred.InputSplit;

import org.apache.hadoop.mapred.RecordReader;

import org.apache.hadoop.mapred.FileSplit;

import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.mapred.LineRecordReader;

import org.apache.hadoop.mapred.Reporter;

import com.sun.jna.Native;

import com.sun.jna.Library;

import com.sun.jna.Pointer;

public class SDFRecordReader implements RecordReader<LongWritable, BytesWritable> {

private long pos;

private boolean fileProcessed = false;

byte[] binaryData;

int i = 0;

int from = 0;

LineRecordReader reader;

Pointer HANDLE;

public interface SDFWrapper extends Library {

SDFWrapper INSTANCE = (SDFWrapper)Native.loadLibrary("SDFWrapper", SDFWrapper.class);

long Next(Pointer HANDLE);

void Value(Pointer HANDLE, byte[] arrBuffer, long nLength);

void Close(Pointer HANDLE);

Pointer Load(String strFileName, String strTableName, String strColumnSplitter, String strEscapeSimbole);

}

private SDFWrapper sdfWrapper = SDFWrapper.INSTANCE;

SDFRecordReader(InputSplit genericSplit, JobConf job, Reporter reporter) throws IOException {

FileSplit split = (FileSplit) genericSplit;

split.getPath();

HANDLE = sdfWrapper.Load("/usr/synopsys/data/", "\t", "\t");

binaryData = new byte[1024];

}

@Override

public boolean next(LongWritable key, BytesWritable value) throws IOException {

// Read from the output byte array until a newline is found

byte[] arrBuffer = new byte[2048];

while (true)

{

long nTo = sdfWrapper.Next(HANDLE);

if(nTo <= 0) break;

// Set the key to the row number (arbitrary - key is not used by Hive)

key.set(i);

// Set the value to the fullrow read from this part of the output byte array

sdfWrapper.Value(HANDLE, arrBuffer, 1024);

value.set(arrBuffer, 0, (int) nTo);

// Increment rowcount

++i;

// There is still more data...

return true;

}

this.fileProcessed = true;

return false;

}

// Generate new key object

@Override

public LongWritable createKey() {

return new LongWritable();

}

// Generate new value object

@Override

public BytesWritable createValue() {

return new BytesWritable();

}

// Return position in file

@Override

public long getPos() throws IOException {

return pos;

}

// Close input streams (raw and uncompressed)

@Override

public void close() throws IOException {

sdfWrapper.Close(HANDLE);

}

// Progress for this file is 1 when finished processing, 0 otherwise

@Override

public float getProgress() throws IOException {

if (fileProcessed) { return 1.0f; }

else { return 0.0f; }

}

}

Ներկայացված է *SDFReader* դասը ժառանգված *RecordReader* դասից ծածկելով *next, createKey, createValue, getPos, close, getProgress* մեթոդները։ Օգտագործվել է *c\_sdf\_wrapper* գրադարանը և նրա *Next, Value, Close, Load*  մեթոդները։ Դասը հնարավորություն է տալիս *Hive* գործիքին տեսնել և ներկայացնել *Hadoop* համակարգում առկա *SDF*  տվյալներն որպես աղյուսակային տվյալներ։

## SDF տվյալների հասանելիություն

Java ծրագրավորման լեզվով գրված Hive գործիքի ներդրված ֆունկցիաներին՝ SDF ձևաչափի տվյալներին հասանելիություն ապահովելու համար մշակվել և իրականցվել է C++ ծրագրավորման լեզվով c\_sdf\_wrapper C-ական գրադարանը։ Այս գրադարանը տրամադրում է հետևյալ հիմնական ֆունկցիաները՝

* Load
* Next
* Value
* Close

**Load** ֆունկցիան որպես մուտքային արգումենտներ ստանում է մշակվող ֆայլը և այն աղյուսակի անունը, որը ցանկում ենք կարդալ։ Ֆունկցիան կատարում է պահանջվող ֆայլի և աղյսուակի առկայության ստուգում և դրանց առկայության դեպքում նախապատրաստում է աղյուսակը հետագա ընթերցման համար։ Այս ֆունկցիան հաջող ավարտի պարագայում վերադարձնում է HANDLE տիպի օբյեկտ, որը անհրաժեշտ է մյուս մեթոդներին։

**Next** ֆունկցաին որպես միակ մուտքային արգումենտ ստանում է Load-ի կողմից վերդարձված HANDLE օբյեկտը։ Այս ֆունկցաին նախապատրաստում է տրված HANDLE օբյեկտի աղյուսակի մեկ տողը ընթերցման համար՝ այն տեղավորելով ընթերցման բուֆերի մեջ։ Next ֆունկցիան տողի՝ առկայության դեպքում, վերադարձնում է բուֆերում առակ բայթերի քանակը՝ որով կարելի է ստուգել, արդյոք աղյուսակի ընթերցումը ավարտվել է, թե ոչ։

**Value** ֆունկցիան ստանում է HANDLE օբյեկտը, C-ական char-երի զանգված (char\*) որպես ընթերցման բուֆեր և ընթերցվելիք բայթերի քանակը։ Որպես ընթերվելիք բայթերի քանակ ցանկալի է տալ Next -ի վերադարձրած արժեքը։ Այս ֆունկցիան կատարում է HANDLE-ի բուֆերից տվյալների ընթերցում դեպի տրված բուֆեր։

**Close** ֆունկցիան ստանում է HANDLE տիպի արգումենտ և կատարում է այդ օբյեկտի կողմից օգտագործվող հիշողության ազատում։ Այս մեթոդը անհրաժեշտ է օգտագործել տվյալների ընթերցման ավարտից հետո։

### c\_sdf\_wrapper գրադարան

#ifndef C\_SDF\_WRAPPER\_H

#define C\_SDF\_WRAPPER\_H

// Includes

#include "c\_sdf\_wrapper\_global.h"

#include <sym\_sdf.h>

#include <QByteArray>

//! Types

using t\_buffer = QByteArray;

/\*

struct SDFWrapperData.

Data need for API

\*/

struct SSDFWrapperData

{

::SYM\_SDF::CSDFSegment const\* m\_pcSegment = nullptr;

qint64 m\_nRow = -1;

char m\_chColumnSpliter = '\t';

char m\_chEscapeSymbol = '/';

t\_buffer m\_aBuffer;

};

using HANDLE = SSDFWrapperData\*;

/\*

Declaration of API functions

\*/

extern "C" {

HANDLE C\_SDF\_WRAPPER\_API Load(char const\* chszFileName, char const\* chszTableName, char chColumnSpliter, char chEscapeSymbol);

void C\_SDF\_WRAPPER\_API Close(HANDLE pHandle);

size\_t C\_SDF\_WRAPPER\_API Next(HANDLE pHandle);

void C\_SDF\_WRAPPER\_API Value(HANDLE pHandle, char\* pBuffer, size\_t nLenght);

}

#endif // C\_SDF\_WRAPPER\_H

### c\_sdf\_wrapper գրադարանի իրականացումը

//! Includes

#include "c\_sdf\_wrapper.h"

#include <cctype>

#include <cstring>

#include <algorithm>

HANDLE Load(char const\* chszFileName, char const\* chszTableName, char chColumnSpliter, char chEscapeSymbol)

{

HANDLE pHandle = nullptr;

MYM\_TRY

if ( chszFileName == nullptr || chszTableName == nullptr )

return nullptr;

// create new handle

pHandle = new(::std::nothrow) SSDFWrapperData();

if ( pHandle == nullptr )

return nullptr;

// set split and escape symbols

pHandle->m\_chColumnSpliter = chColumnSpliter;

pHandle->m\_chEscapeSymbol = chEscapeSymbol;

::SYM\_SDF::CSDFVector oSDFVector;

::SYM\_SDF::CSDFFilter oFilter;

// set segment filter

{

::SYM\_SDF::CSDFSegmentFilter\* pSegmentFilter = oFilter.GetSegmentFilter();

MYM\_CHECK\_PTR(pSegmentFilter);

pSegmentFilter->SetName(chszTableName);

}

// load SDF

MYM\_CHECK(oSDFVector.Load(chszFileName, &oFilter));

// get segment index

qint64 idxSegment = oSDFVector.GetSegmentIndexByName(chszTableName);

MYM\_CHECK\_IDX(idxSegment, oSDFVector.GetSegmentCount());

// take segment

pHandle->m\_pcSegment = oSDFVector.RemoveSegment(idxSegment, false);

if ( pHandle->m\_pcSegment == nullptr )

{

// segment not exists, close HANDLE

Close(pHandle);

return nullptr;

}

MYM\_CATCH\_CBEGIN(oExp)

Close(pHandle);

pHandle = nullptr;

MYM\_CATCH\_END

return pHandle;

}

void Close(HANDLE pHandle)

{

if ( pHandle != nullptr )

{

if ( pHandle->m\_pcSegment != nullptr )

delete pHandle->m\_pcSegment;

delete pHandle;

}

}

size\_t Next(HANDLE pHandle)

{

if ( pHandle == nullptr || pHandle->m\_pcSegment == nullptr )

return -1;

if ( (++pHandle->m\_nRow) >= pHandle->m\_pcSegment->GetRowCount() )

return -1;

pHandle->m\_aBuffer.clear();

{

QString const sReplace = QString("%1%2").arg(pHandle->m\_chEscapeSymbol).arg(pHandle->m\_chColumnSpliter);

qint64 const nColCount = pHandle->m\_pcSegment->GetColumnCount();

for ( qint64 nCol = 0; nCol < nColCount; ++nCol )

{

// read cell data into the temporary buffer

qint64 nBufferSize = 0;

char const\* pszBuffer = reinterpret\_cast<char const\*>(pHandle->m\_pcSegment->fastGetValue(pHandle->m\_nRow, nCol, &nBufferSize));

QByteArray aBuffer(pszBuffer, nBufferSize);

// set escape symbol before column splitter symbols

aBuffer.replace(pHandle->m\_chColumnSpliter, sReplace);

// write cell data into the buffer

pHandle->m\_aBuffer.push\_back(aBuffer);

pHandle->m\_aBuffer.push\_back(pHandle->m\_chColumnSpliter);

}

if ( !pHandle->m\_aBuffer.isEmpty() )

pHandle->m\_aBuffer.resize(pHandle->m\_aBuffer.size() - 1); // remove last splitter symbol

}

return pHandle->m\_aBuffer.size();

}

void Value(HANDLE pHandle, char\* pBuffer, size\_t nLenght)

{

if ( pHandle == nullptr || pHandle->m\_pcSegment == nullptr )

return;

::std::memmove(reinterpret\_cast<void\*>(pBuffer), reinterpret\_cast<void const\*>(pHandle->m\_aBuffer.data()), sizeof(t\_buffer::value\_type) \* qMin<size\_t>(nLenght, pHandle->m\_aBuffer.size()));

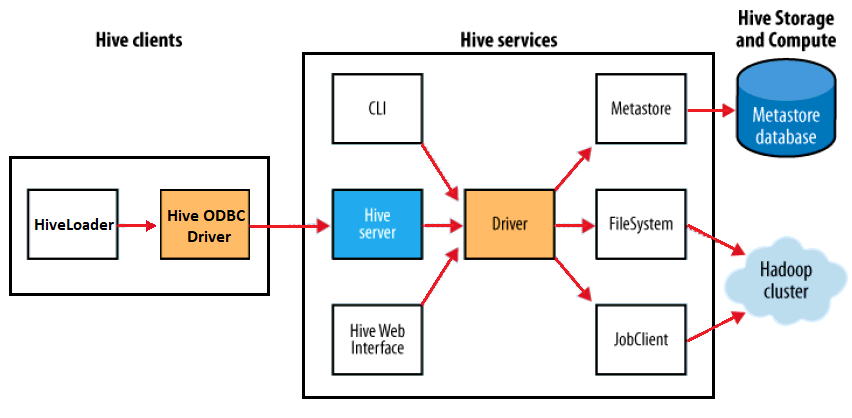
}

## HiveLoader գործիք

*HiveLoader* գործիքը նախագծվել և իրագործվել է աղյուսակային ձևաչափերով ներկայացվող տվյալները Hive-ի պահուստներում պահուստավորելու համար։ Ծրագիրը իրականցվել է *C++* ծրագրավորման լեզվով և նրա *Qt* գրադարանի միջոցով, այդ պատճառով գործիքը հանդիսանում է օպերացիոն համակարգից անկախ և կարող է աշխատել օրինակ ինչպես *Windows* այնպես էլ *Linux* օպերացիոն համակարգերում։ Սրա միջոցով հնարավոր է տվյալները տեղափոխել օրինակ *Windows*-ի ֆայլային համակարգից դեպի *Linux* միջավայրում աշխատող *Hive*-ի պահուստներ, ինչը կարևոր հանգամանք է գործիքի իրական կիրառելիության համար։

### Աշխատանքի սկզբունքները

*HiveLoader* գործիքը հանդիսանում է *Hive* համակարգի համար հաճախորդ (անգլ․՝ client) ծրագիր։ *Hive* ծրագարյին համակարգի հետ *SQL* հարցումներով աշխատելու համար հաճախորդ ծրագրերը պետք է օգտագործեն *Hive Thrift Client, Hive JDBC Driver* կամ *Hive ODBC Driver* կառավարիչ (անգլ․՝ driver) ծրագրերից մեկը։ Գործիքի կապը *Hive*-ի հետ պատկերված է նկար 3-ում։



Նկ․ 2 HiveLoader-ի կապը Hive-ի հետ։

Ինչպես երևում է նկարից՝ *HiveLoader*-ը աշխատում է *Hive ODBC Driver*-ի միջոցով, մասնավորապես աշխատանքի կատարման ժամանակ օգտագործվել է *Hortonworks* *Hive ODBC Driver*-ը։

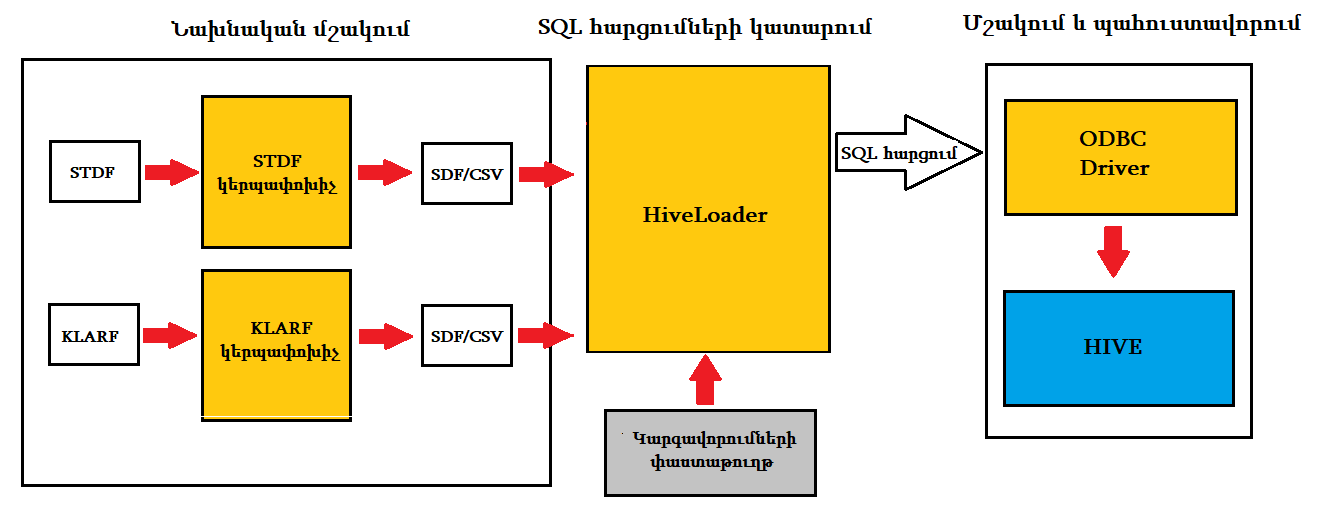
HiveLoader գործիքը տալիս է հնարավորություն տվյալները *Hive*-ում պահուստավորելու մի քանի եղանակներով։ Դրանցից առաջինի դեպքում տվյալները վերափոխվում են *SQL* գրառման և փոխանցվում են *Hive*-ին, որը և կատարում է այդ տվյալների հետագա մշակումը, իսկ երկրորդ դեպքում տվայլները տեղափոխվում են *HDFS* համակարգ, որտեղ և կատարվում է տվյալների հետագա մշակումը *Java* ծրագրավոման լեզվով նախապես իրականացված հատուկ ֆունկցիաների միջոցով։

Գործիքը կարողանում է աշխատել *SDF* և *CSV* ձևաչափի տվյալների հետ։ Այսպիսով մնացած բոլոր տվյալների պահուստավորումն ապահովելու համար անհրաժեշտ է դրանք նախապես կերպափոխել վերը նշված ձևաչափերից որևէ մեկին։

*HiveLoader*-ի միջոցով կարելի է պահուստավորվել կամայական ձևաչափի տվյալներ, պարզապես կատարելով տվյալների նախնական ձևափոխություն։

*HiveLoader*-ի աշխատանքը կարգավորվում է հատուկ մշակված կարգավորումների փաստաթղթի միջոցով, որտեղ նկարագրվում է, թե որ տվյալները որ պահուստներում, աղյուսակներում և ինչ մեթոդով պետք է պահվեն։

Կիսահաղորդչային տվյալների պահեստավորման ամբողջ ճանապարհը պատկերված է նկար 4-ում։



Նկ․ 3 HiveLoader-ի աշխատանքը։

*HiveLoader* գործիքը *Hive*-ի և նրա պահուստների հետ աշխատելու համար տրամադրում է հետևյալ հնարավորությունները՝

* Պահուստի ստեղծում
* Աղյուսակների ստեղծում
* Աղյուսակների սխեմայի ձևափոխում
* Աղյուսակներում տվյալների ավելացում

### Պահուստի ստեղծում

Գործիքը տվյալները պահեստավորելուց առաջ ստուգում է տրված պահուստի առկայությունը *Hive*-ում, և դրա բացակայության դեպքում ավտոմատ կերպով ստեղծում է այն՝ *Hive*-ի համապատասխան *SQL* հրամանի միջոցով։

Օրինակ`

CREATE DATABASE IF NOT EXISTS my\_database:

### Աղյուսակների ստեղծում

Սա թերևս այս գործիքի կարևորագույն հնարավորությունն է։ Գործիքը նախապես ստեղծելով տվյալների պահուստը (եթե իհարկե մինչ այդ այն գոյություն չուներ) ստուգում է տվյալ անունով աղյուսակի առկայությունը պահուստում։ Աղյուսակի բացակայության դեպքում, օգտագործելով համապատասխան *SQL* հրամանը, ստեղծում է աղյուսակը՝ օգտագործելով իրական տվյալի (օրինակ ՝ *SDF* կամ *CSV* ձևաչափի) աղյուսակի սխեման և կատարելով կարգավորումների փաստաթղթում նկարագրված ձևափոխությունները (oրինակ՝ սյան անվանափոխություն կամ սյան տիպի փոփոխում)։ Սյան անվանափոխության հնարավորությունը անհրաժեշտ է, քանի որ *Hive*-ում գոյություն ունեն բանալի բառեր որոնք թույլ չեն տալիս ունենալ այդ անունով սյունակներ աղյուսակներում, ինչպես օրինակ “order” բառը։

Տվյալներին հասանելիության բարձր արագություն ապահովելու համար *Hive*-ը հնարավորություն է տալիս տվյալները մասնատել՝ համապատասխան որոշակի սյուների արժեքների։ Սա էականորեն փոքրացնում է տվյալների փնտրման ժամանակը՝ հահկապես եթե հասցումն արվում է տվյալ սյան արժեքները ֆիլտրելով․ տվյալ մասնատված սյան արժեքի համար միարժեքորեն որոշվում է համապատասխան աղյուսակի մասը, և փնտրման գործողությունը շարունակվում է միայն գտնված փոքր աղյուսակներում զուգահեռացված։

Կիսահաղորդիչային աղյուսակային տվյալների առանցքային մաս են կազմում այսպես կոչվող «կոնտեքստային» սյուները, որոնցից են *Lot, Wafer, Parameter․․․* Սրանցով են կազմվում հիմնական հարցումները տվյալների ստացման և մշակման համար։ Այդ պատճառով կիսահաղորդչային տվյալները հարմար է պահուստավորել մասնատված ըստ կոնտեքստային սյուների։ *HiveLoader* գործիքը տալիս է հնարավորություն յուրաքանչյուր աղյուսակի համար սահմանել մասնատման սյունակները։ Գործիքը, օգտագործելով կարգավորումների փաստատթղթում տրված մասնատման սյունակների անունները, վերածում է այն համապատասխան *SQL* հարցման։ Օրինակ, եթե ուզում ենք տվյալը մասնատել ըստ *Lot* և *Wafer* սյուների արժեքների, ապա *HiveLoader*-ը *Hive*-ում կատարում է հետևյալ *SQL* հարցումը՝

CREATE TABLE BinData (HBIN INT, SBIN INT, …)

PARTITIONED BY (LOT STRING, WAFER STRING);

### Աղյուսակների սխեմայի ձևափոխում

Տվյալների պահուստավորման ժամանակ հանդիպում են դեպեքեր, երբ աղյուսակը որի մեջ մետք է ավելացվեն նոր տվյալները՝ իր կառուցվածքով ամբողջապես չի համապատասխանում *Hive*-ի պահուստում արդեն գոյություն ունեցող նույն անունով աղյուսակի կառուցվածքին։ Այս դեպքում առաջանում է անհրաժեշտություն ձևափոխել պահուստում առկա աղյուսակի կառուցվածքը։ Այս հնարավորթույնը նույնպես ապահովված է *HiveLoader* գործիքի կողմից։ Անհրաժեշտության դեպքում գործիքը կատարում է աղյուսակի ձևափոխման *SQL* հարցում, որից հետո միայն կատարվում է տվյալների ավելացում արդեն ձևափոխված աղյուսակում։ Հաճախ հանդիպող դեպք է, երբ պետք է լինում ավելացնել նոր սյուն, այդպսիպ *SQL* հրամանի օրինակ ՝

ALTER TABLE table\_name ADD COLUMNS (col\_name data\_type, ...)

Alter հրամանի միջոցով հնարավոր է, նաև ավելացնել նոր մասնատում աղյուսակում, օրինակ ՝

ALTER TABLE table\_name ADD [IF NOT EXISTS] PARTITION partition\_spec   
[, partition\_spec]

Որտեղ՝

partition\_spec: (partition\_column = partition\_col\_value,

partition\_column = partition\_col\_value, ...)

Այս երկու հարամանները կազմվում են և իրականացվում *HiveLoader*-ում։ Անհրաժեշտության դեպքում *HiveLoader*-ում հեշտությամբ կարելի է իրականցվել նաև ավելի բարդ տիպի աղյուսակների ձևափոխություններ։

### Աղյուսակներում տվյալների ավելացում

*Hive*-ը տրամադրում է տվյալների ավելցման մի քանի հրամններ, դրանք են՝

* Load

Այս դեպքում՝ բեռնվող տվյալների հետ Hive-ը չի կատարում որևէ ձևափոխություններ դրանք աղյուսակներում տեղադրելու համար։ Load հրամանը կատարում է հասրակ պատճենահնում (անգլ․՝ copy/paste), որը տվյալների փաստաթղթերը տեղադրում է աղյուսակներին համապատասխան հասցեներում (HDFS ֆայլային համակարգի վրա)։ HiveLoader-ը այս SQL հրամանը կատարում է այն ժամանակ, երբ աղյուսակը նոր է ստեղծվում և դեռ չի պարունակում տվյալներ։

* Insert  
  Այս հարցումը թույլ է տալիս տվյալները բեռնել ուղղիղ *SQL* -ից։ Insert հրամանը հնարավորություն է տալիս ինչպես վերագրանցել (անգլ․՝ overwrite) տվյալները, այնպես էլ ավելացնել նոր տողեր արդեն գոյություն ունեցողներին։ *HiveLoader*-ը օգտագործում է վերոհիշյալ *SQL* հրամանը՝ գոյություն ունեցող և նոր տվյալներին համապատասխան կառուցվածք ունեցող աղյուսակների մեջ նոր տողեր ավելացնելու համար։ Տվյալ մոտեցումը հարմար է և կիրառելի միայն փոքր ծավալի տվյալների համար։ Փորձը ցույց է տալիս, որ քսան և ավելի սյուն պարունակող աղյուսկների դեպքում այս հրամանի կատարումը հարյուրից ավել տողեր պարունակող տվյալների համար կիրառելի չէ, քանի որ այն տևում է մի քանի րոպե։

Սրա պատճառներից մեկը այն է, որ աղյուսակի տողերը վերածվում են տողային տեսքի և փոխանցվում են *Hive*-ին որպես *SQL* հարցում, իսկ *Hive*-ն իր հերթին այն գրում է տվյալ աղյուսակին համապատասխանող փաստաթղթում կամ փաստաթղթերում (եթե աղյուսակը պահված է մասնատված)։

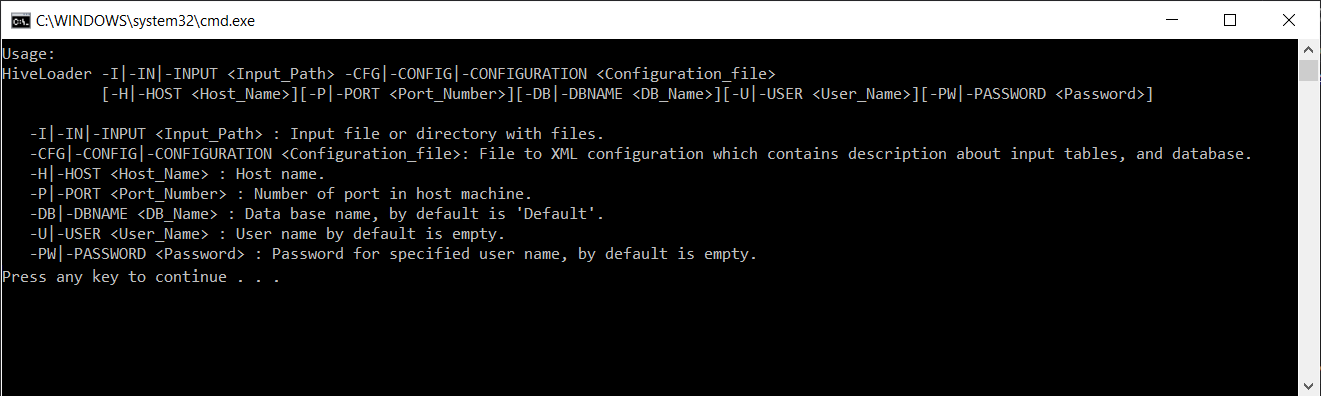
### Հրամնանների դաշտ

Այս գործիքը աշխատում է օպերացիոն համակարգերի հրամանների դաշտից և չունի գրաֆիկական ինտերֆեյս։ Գործիքի մուտքային արգումենտները նկարագրված են աղյուսակ 4 -ում՝

Աղ․ 4 HiveLoader գործիքի մուտքային արգումենտների նկարագրություն։

|  |  |  |
| --- | --- | --- |
| **Անուն** | **Նկարագրություն** | **Պարտադիր է** |
| -I|-IN|-INPUT | Մուտքային ֆայլ(SDF ձևաչաձ) կամ ֆայլերի պանակ | Այո |
| -CFG|-CONFIG| -CONFIGURATION | Կարգավորումների ֆայլ (XML ձևաչափ) | Այո |
| -H|-HOST | Հյուրնկալող սերվեր մեքենայի IP հասցեն է, որի վրա գործում է Hive-ը | Ոչ, բայց այո, եթե բացակայում է կարգավորումների ֆայլում |
| -P|-PORT | Սերվերի պորտի համարն է, որը  սպասարկում է HiveServer-ը | Ոչ, բայց այո, եթե բացակայում է կարգավորումների ֆայլում |
| -DB|-DBNAME | Hive-ի վրա գոյություն ունեցող  (կամ ստեղծվելիք) տվյալներ բազայի անունն է։ Լռելիությամբ սրա արժեքը Default է։ | Ոչ, բայց այո, եթե բացակայում է կարգավորումների ֆայլում |
| -U|-UESER | Օգտագործողի անունն է  Hive-ում | Ոչ, բայց այո, եթե բացակայում է կարգավորումների ֆայլում |
| -PW|-PASSWORD | Օգտագործողի գաղտնաբառն է  Hive-ում | Ոչ, բայց այո, եթե բացակայում է կարգավորումների ֆայլում |

Աղ․ 4-ի շարունակություն



Նկ․ 4 HiveLoader գործիքի օգտագործումը հրամանների դաշտից։

### Կարգավորումների փաստաթուղթ

*HiveLoader*-ի տվյալների և *Hive*-ի հետ աշխատանքը կարգավորվում է հատուկ մշակված կարգավորումների փաստաթուղթի միջոցով, որը ծրագրին տրվում է հրամանների դաշտից (ավելի մանրամասն նկարագրված է աղյուսակ 4-ում)։ Կարգավորմների ֆայլը իրենից ներկայացնում է *XML* (ընդարձակ նշման լեզու) ձևաչափով ներկայացվող որոշակի կարգավորումների խումբ։

### Կարգավորումների փաստաթղթի նկարագրություն

*HiveLoader*-ի կարգավորումների փաստաթուղթը *XML* ձևաչափով պահվող ստրուկտուրաների համախումբ է։

Փաստաթուղթը պարունակում է մեկ գլխավոր տարր*` <HiveLoader>*։

*<HiveLoader Version=”1.0.0.0”>  
 <DB>…</DB>  
 <Tables>…</Tables>  
 </HiveLoader>*

Աղ․ 5 Կարգավորման փաստաթղթի <HiveLoader> տարրի բովանդակություն։

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Անուն** | **Տիպ** | **Արժեքի տիպ** | **Նկարագրություն** | **Պարտադիր է** |
| Version | Բնորոշիչ | Տող (String) | Կարգավորումների տարբերակը ցույց տվող դաշտ է (օրինակ՝ 1.0.0.0) | Ոչ |
| DB | Տարր |  | Տվյալների պահուստի մասին ինֆորմացիա | Ոչ |
| Tables | Տարր |  | Աղյուսկների պահեստավորելու նկարագրությունների ցուցակ | Այո |

*<DB>* տարրի մանրամասն նկարագրությունը տրված է աղյուսակ 6-ում:

*<DB>*

*<Host>10.116.19.15</Host>*

*<Port>11001</Port>*

*<DBName>Hive\_Horton</DBName>*

*<User>hive</User>*

*<Password></Password>*

*</DB>*

Աղ․ 6 Կարգավորման փաստաթղթի <DB> տարրի բովանդակություն։

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Անուն** | **Տիպ** | **Արժեքի տիպ** | **Նկարագրություն** | **Պարտադիր է** |
| Host | Տարր | Տող (String) | Հյուրնկալող սերվեր մեքենայի IP հասցեն է, որի վրա գործում է Hive-ը | Այո |
| Port | Տարր | Թվանշան | Սերվերի պորտի համարն է, որը  սպասարկում է HiveServer-ը | Այո |
| DBName | Տարր | Տող (String) | Hive-ի վրա գոյություն ունեցող  (կամ ստեղծվելիք) տվյալներ բազայի անունն է։ | Այո |
| User | Տարր | Տող (String) | Օգտագործողի անունն է Hive-ում | Այո |
| Password | Տարր | Տող (String) | Օգտագործողի գաղտնաբառն է  Hive-ում, եթե օգտագործողը չունի գախտնաբառ, ապա կարող է լինել դատարկ տող | Այո |

Աղ․ 6 -ի շարունակություն

*<Tables>* տարրը հանդիսանում է մակ կամ ավելի *<Table>* տարրերի ցուցակ։ *<Table>* տարրի նկարագրությունը տրված է աղյուսակ 7-ում։

*<Tables>*

*<Table Name="BinData" Format=" ORC">  
 <Partitions>Lot,Wafer</Partitions>  
</Table>  
<Table Name="BinData" HiveTableName="BinData\_TEXTFILE" Format="TEXTFILE">  
 <Partitions>Lot</Partitions>  
 <ColumnMapping>  
 <Column PreviousName="Order" NewName="Order\_"/>  
 </ColumnMapping>  
</Table>*

*</Tables>*

Աղ․ 7 Կարգավորման փաստաթղթի <Table> տարրի բովանդակություն։

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Անուն** | **Տիպ** | **Արժեքի տիպ** | **Նկարագրություն** | **Պարտադիր է** |
| Name | Բնորոշիչ | Տող (String) | Աղյուսակի անունն է, որի համար տրվուն է կարգավորումը | Այո |
| HiveTableName | Բնորոշիչ | Տող (String) | Աղյուսակի անունն է, Hive-ում, բացակայության դեպքում նույն է ինչ անունը տվյալի աղբյուրում | Ոչ |
| Format | Բնորոշիչ | Տող (String) | Hive-ի կողմից սպասրկվող պահուստավորման ձևաչափ (օրինակ՝parquet), կամ custom: | Ոչ, լռելիությամբ ձևաչափը orc է |
| InputFormat | Բնորոշիչ | Տող (String) |  | Ոչ |
| OutputFormat | Բնորոշիչ | Տող (String) |  | Ոչ |
| Partitions | Տարր | Տող (String) | Մասնատնման սյունակների ցուցակ է՝ բաժանված ,-ով։ | Ոչ |
| ColumnMapping | Տարր |  | Ասոցացիա է տվյալում առկա սյան անվան և Hive-ում պահվելիք անվան միջև | Ոչ |

Աղ․ 7 -ի շարունակություն

Կարգավորումների փաստաթղթի ամբողջական օրինակ՝

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

*<HiveLoader Version="1.0.0.0">*

*<DB>*

*<Host>10.116.19.15</Host>*

*<Port>11001</Port>*

*<DBName>Hive\_Horton</DBName>*

*<User>hive</User>*

*<Password></Password>*

*</DB>*

*<Tables>*

*<Table Name="BinData" HiveTableName="BinData\_Avro" Format="Avro">*

*<Partitions>Lot</Partitions>*

*<ColumnMapping>*

*<Column PreviousName="Order" NewName="Order\_"/>*

*</ColumnMapping>*

*</Table>*

*<Table Name="ParameterData" HiveTableName="ParameterData" Format="ORC">*

*<Partitions>Lot,Parameter</Partitions>*

*<ColumnMapping>*

*<Column PreviousName="ORDER" NewName="Order\_"/>*

*</ColumnMapping>*

*</Table>*

*</Tables>*

*</HiveLoader>*

# ՓՈՐՁԱՐԱՐԱԿԱՆ ՏԵԽՆԻԿԱ

Նախագծման համար օգտագործված սերվերի տեխնիկական տվյալները.

հիշողություն - 320Gb, օպերատիվ հիշողություն - 72Gb, տիպը – HP ProLiant DL360 G7 x64, պրոցոսոր – 2x Intel64 Family 6 Model 44 Stepping 2 GenuineIntel ~1599 Mhz, օպերացիոն համակարգ – Microsoft Hyper-V Server 2016:

Սերվերի վրա պատրաստված տեղադրված վիրտուալ մեքենաներ. 2x հիշողություն - 60Gb, օպերատիվ հիշողություն - 34Gb, տիպը – վիրտուալ x64, պրոցոսոր – Intel64 Family 6 Model 44 Stepping 2 GenuineIntel ~1599 Mhz, օպերացիոն համակարգ – CentOS Linux release 7.3.1611:

Օգտագործողի մեքենայի տեխնիկական տվյալները. հիշողություն - 250Gb, օպերատիվ հիշողություն - 8Gb, տիպը – Dell Latitude E7450 x64, պրոցոսոր – Intel64 Family 6 Model 61 Stepping 4 GenuineIntel ~2594 Mhz, օպերացիոն համակարգ – Microsoft Windows 10 Enterprise:

Օգտագործողի մեքենայի և սերվերի միջև կապի միջոցները, տեխնիկական տվյալները. կապի միջոց - WiFi կամ Ethernet, կապ տեսակներ – Windows native Remote Desktop Managers, Server Manager, Hyper-V Manager, MobaXterm 10.4 (SSH սեսիա օգտագործելով NAT մեթոդը վիրտուալ մեքենան սերվերից դուրս ներկայացնելու համար):

# ՓՈՐՁՆԱԿԱՆ ՀԵՏԱԶՈՏՈՒԹՅՈՒՆՆԵՐ

Փորձ է արվել ստեղծել *Hadoop* համակարգի հանգույցներ մի քանի մեքենաների վրա, օգտագործել *Hive* գործիքը՝ աղյուսակային *SDF* կիսահաղորդչային ձևաչափի տվյալների հետ աշխատելու համար, կատարել տարբեր արագագործության չափումներ, հետազոտություններ և փորձարկումներ, մշակել և կիրառել մեթոդներ, որի արդյունքում հնարավոր կլինի ապահովել *SDF* տվյալների կիրառությունը *Hadoop* համակարգում։ Դրա համար սերվերի վրա ստեղծվել են վիրտուալ մեքենաներ՝ ապահովելով դրանց հասանելիությունը սերվերից դուրս *Microsoft PowerShell*-ի *New-VMSwitch, New-NetIPAddress, New-NetNat, Add-NetNatStaticMapping․․․* հրամանների միջոցով։ Մեքենաների վրա տեղադրվել է *Hortonworks*համակարգը՝ նախատեսված *Hadoop* համակարգի և դրա հետ կապված գործիքների հետ հարմար աշխատելու համար։ Համակարգը հնարավորություն է տալիս կատարել բազմաբնույթ գործողություններ՝ ինչպես համակարգի աշխատանքի ընթացքում այնպես էլ դրա գործարկումից առաջ և հետո․ *Hortonworks*համակարգն ունի հնարավորություն ստեղծելու *Hadoop* համակարգի հանգույցներ ինչպես մեկ մեքենայի այնպես էլ ցանցում գտնվող բոլոր նմանատիպ մեքենաների վրա։ Մեր հեսազոտությունների շրջանակներում *Hortonworks*համակարգը կարգավորվել է *Hive*գործիքի հետ աշխատելու համար։

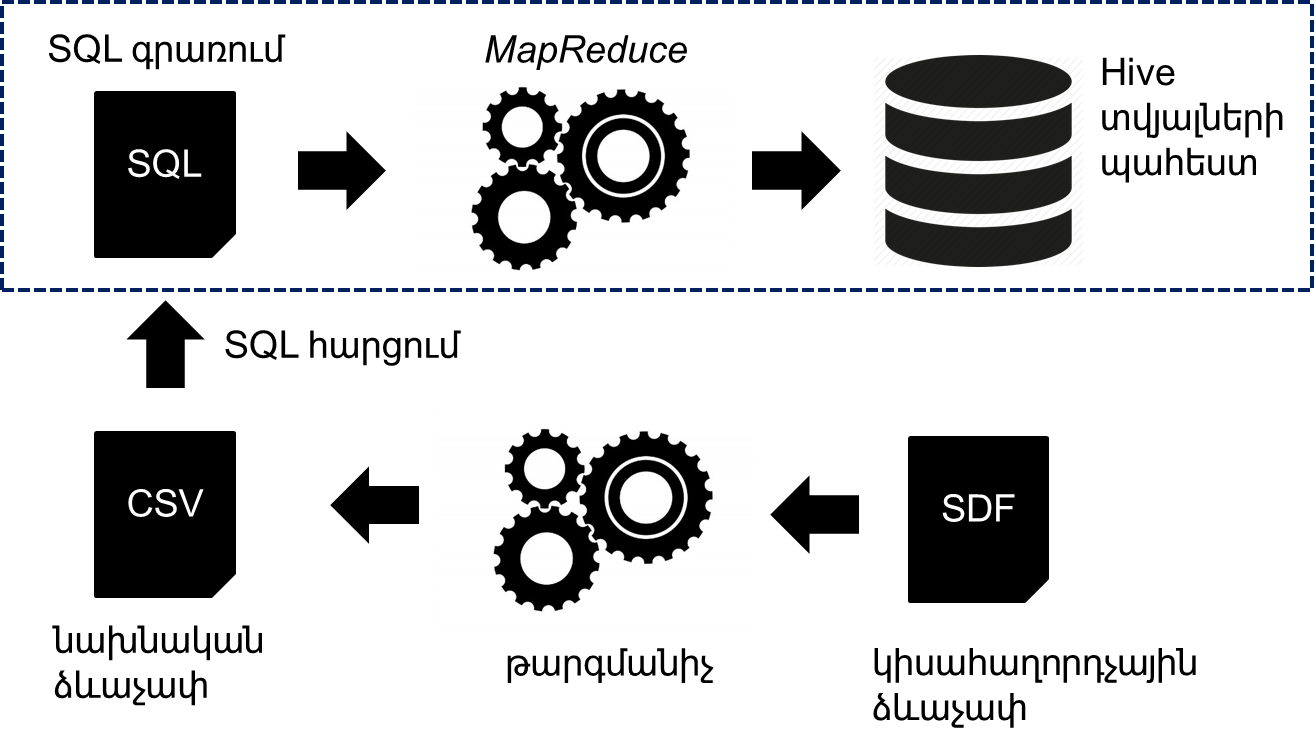
Օգտագործողների մեքենաների վրա տեղադրվել է *Hortonworks Hive ODBC Driver***՝** *Hortonworks*համակարգի և վիրտուալ մեքենաներում աշխատող մեծ տվյալների բազայի կապն ապահովելու համար։

Մշակվել և իրականցվել է *HiveLoader* գործիքը, որի արդյունքում հնարավոր է դարձել իրականացնել և փորձնականում հետազոտել մշակված պահուստավորման մեթոդները։ Գործիքը ապահովում է կապը օգտագործող մեքենաների և *Hive* գործիքի միջև և տրամադրում է տվյալների պահուստավորման արագության մասին տվյալներ։

## *SDF* ձևաչափի տվյալների պահուստավորումը *Hive* համակարգում

Աշխատանքի ընթացքում ուսումնասիրվել են *Hive* համակարգում մեծ տվյալներ մուտքագրելու մի քանի մեթոդներ, ինձպես նկարագրված է ստորև․

Մեթոդ 1․ *Hive* գործիքի միջոցով *Hadoop* համակարգում առկա աղյուսակներից համապատասխան *SDF* ձևաչափի տվյալի պահուստավորման համար նախ *SDF* ձևաչափի տվյալը թարգմանիչի միջոցով կերպափոխվում է *CSV* ձևաչափի, որը *SQL* հարցման միջոցով պահուստավորվում է *Hive*-ի համապատասխան աղյուսակում։ Թարգմանիչն իրենից ներկայացնում է արդեն գոյություն ունեցող ծրագրային միջոց։



Նկ. 5 Մեթոդ 1-ի աշխատանքի սխեման։

Սույն մեթոդի կիրառմամբ ստացել ենք հետևյալ փորձնական արդյունքները․  
Օգտագործվել է 2 հանգույցով մեքենայական կլաստեր՝ ընդհանուր 10 պրոցեսոր և 34 Գբ․ օպերատիվ հիշողությամբ։

Աղ 8․ Մեթոդ 1-ի արդյունքները։

|  |  |
| --- | --- |
| Ծավալ (Մոտավոր տողերի քանակ) | Պահուստավորման տևողություն (Վայրկան) |
| 1000 | 2.3 |
| 10000 | 18.6 |
| 100000 | 124.9 |
| 1000000 | 526.2 |

Մեթոդ 2․ *Hive* գործիքի միջոցով *Hadoop* համակարգում *SDF* ձևաչափի տվյալների պահուստավորման համար նախ *SDF* ձևաչափի տվյալը թարգմանչի միջոցով բերվում է մեկ *SQL* հարցման։ Նման մեծ հարցումների մրջոցով հնարավոր է տվյալների բազաում ավելացնել միանգամից մի քանի գրառում (BULK INSERT)։ Ստացված *SQL* գրառումը ուղարկվում *Hive*-ին այն մշակելու համար։ Թարգմանիչն իրենից ներկայացնում է արդեն գոյություն ունեցող ծրագրային միջոց՝ օրինակ *HiveLoader*։



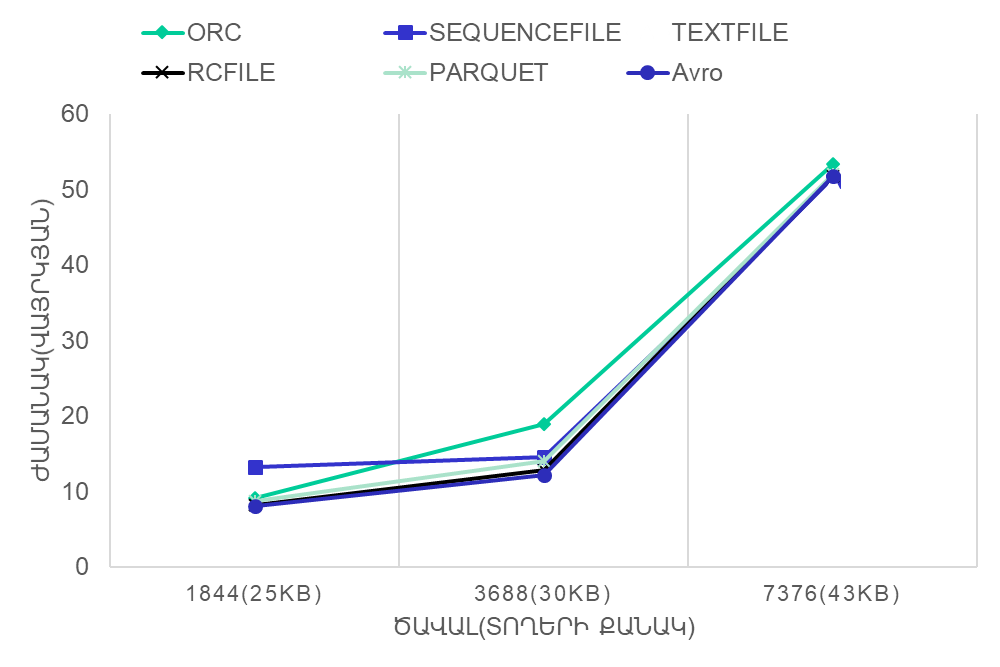
Նկ. 6 Մեթոդ 2-ի աշխատանքի սխեման։

Սույն մեթոդի կիրառմամբ ստացել ենք հետևյալ փորձնական արդյունքները․  
Օգտագործվել է 2 հանգույցով մեքենայական կլաստեր՝ ընդհանուր 10 պրոցեսոր և 34 Գբ․ օպերատիվ հիշողությամբ։

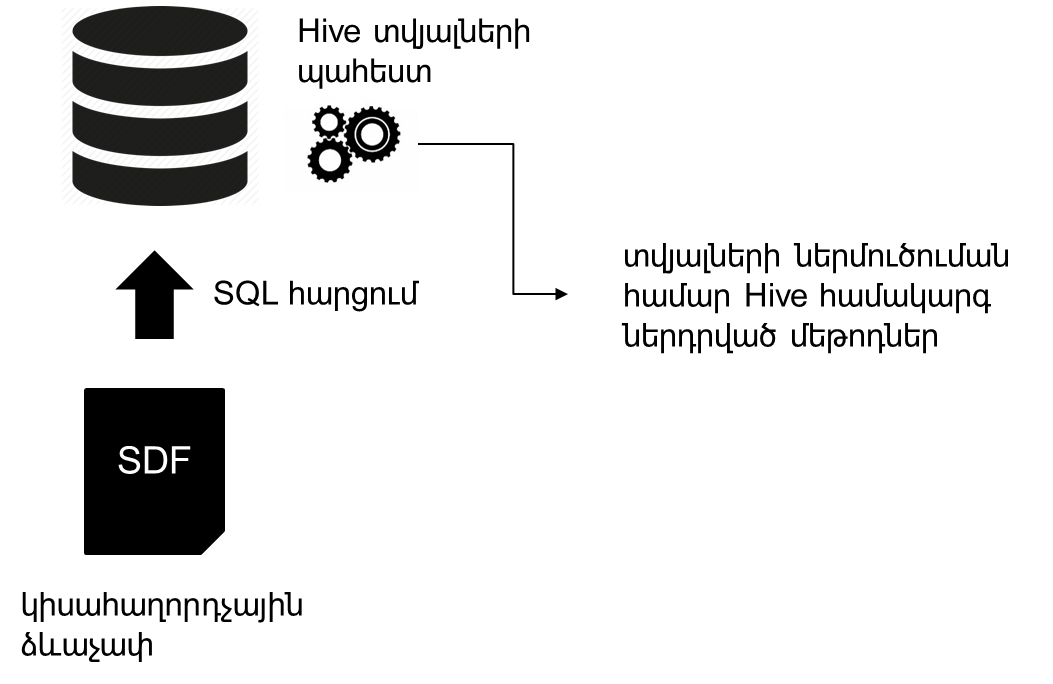
Աղ 9․ Մեթոդ 2-ի արդյունքները։

|  |  |
| --- | --- |
| Ծավալ (Մոտավոր տողերի քանակ) | Պահուստավորման տևողություն (Վայրկան) |
| 1000 | 1.4 |
| 10000 | 14.7 |
| 100000 | 98.6 |
| 1000000 | 345.3 |

Ստորև բերված է *Hive*-ի ներդրված պահուստավորման փաստաթղթային ձևաչափերի համեմատության գրաֆիկը մեթոդ 2-ի համար՝

  
**Նկ․ 7 Պահեստավորման ձևաչափերի համեմատության գրաֆիլ**

Մեթոդ 3․ *Hive* գործիքի միջոցով *Hadoop* համակարգում *SDF* ձևաչափի տվյալին համապատասխան աղյուսակների ստեղծումը և դրանցում տվյալների պահուստավորմը իրագործելու համար նախ ստեղծվել են *Hive* համակարգի կողմից օգտագորվող ներդրված մեթոդներ, որոնք *Hive*-ին թույլ են տալիս աշխատել *Hadoop* համակարգում *SDF* ձևաչափով տվյալների հետ՝ գրել, կարդալ և փոփոխել դրանք, աշխատելով դրանց հետ որպես աղյուսակային տվյալ։ Հաջորդիվ *HiveLoader* գործիքը ուղարկում է համապատասխան *SQL*հարցումը, որը նշում է նախապես մշակված ներդրված մեթոդը և տվյալի գտնվելու հասցեն։ Ներդրված մեթոդը կատարում է *SDF* տվյալի մշակումը և պահուստավորումը։



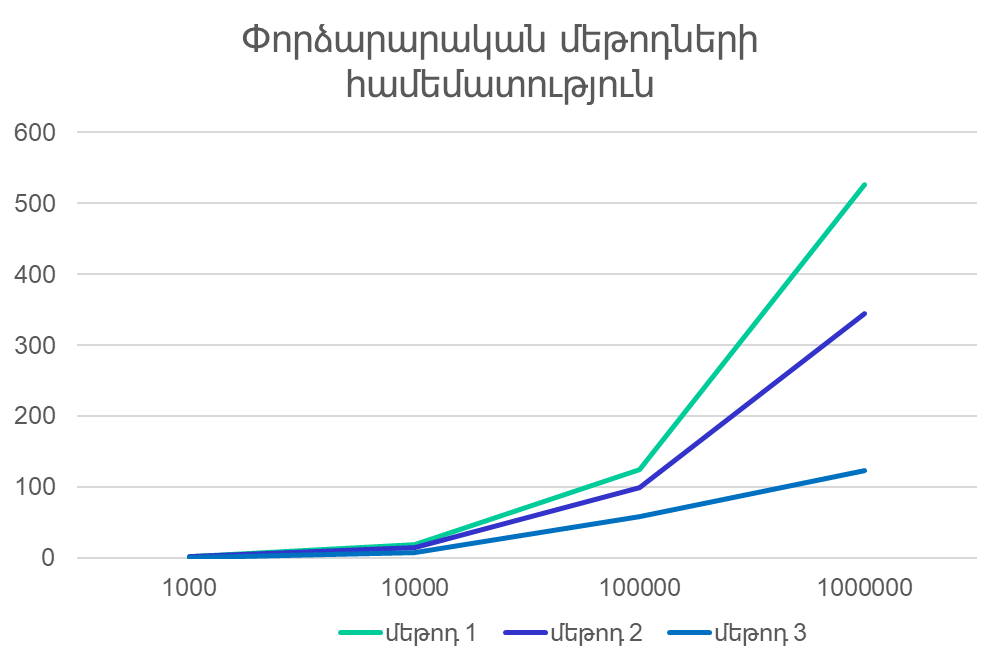
Նկ. 8 Մեթոդ 3-ի աշխատանքի սխեման։

Սույն մեթոդի կիրառմամբ ստացել ենք հետևյալ փորձնական արդյունքները․  
Օգտագործվել է 2 հանգույցով մեքենայական կլաստեր՝ ընդհանուր 10 պրոցեսոր և 34 Գբ․ օպերատիվ հիշողությամբ։

Աղ 10․ Մեթոդ 3-ի արդյունքները։

|  |  |
| --- | --- |
| Ծավալ (Մոտավոր տողերի քանակ) | Պահուստավորման տևողություն (Վայրկան) |
| 1000 | 1 |
| 10000 | 8.1 |
| 100000 | 58.3 |
| 1000000 | 123.4 |

Ստորև բերված է վերը նկարագրված 3 մեթոդների համեմատության գրաֆիկը՝

  
**Նկ․ 9 Մեթոդների համեմատության գրաֆիկ**

# ԵԶՐԱԿԱՑՈՒԹՅՈՒՆ

Մշակվել են մեթոդներ կիրառելով ոչ հարաբերական տվյալների պահոցներում կիսահաղորդչային ձևաչափի տվյալների պահուստավորման համար: Կատարվել են փորձարկումներ և համեմատական հետազոտություններ նշված մեթոդնորի արագագործությունը և կիրառելիուրյունը պարզելու համար։ Մեթոդներից առավել արդյունավետ է *Hive*-ում ներդրված և *SDF* ձևաչափի համար հատուկ նախագծված լուծումը։ Մշակվել է *HiveLoader* գործիքը*,* որը հնարավորություն է տալիս իրական ժամանակում *SDF*  տվյալների օպտիմալ պահուստավորում *Hive* համակարգում՝ հետագայում նրանց արագ հասանելիությունն ապահովելու համար։

Ստացված արդյունքները ցույց են տալիս, որ համակարգը կարելի է ընդլայնել և կիրառել կիսահաղորդչային տվյալների մեծ ծավալների պահեստավորման և մշակման համար։

# ԳՐԱԿԱՆՈՒԹՅԱՆ ՑԱՆԿ

* Tom White: Hadoop: The Definitive Guide, 3th Edition. // O'Reilly Media. -2012. -P. 623.
* Tom White: Hadoop: The Definitive Guide, 4th Edition. // O'Reilly Media. -2015. -P. 756.
* <http://bigdata-madesimple.com/relational-vs-non-relational-databases-part-1/>
* <http://bigdata-madesimple.com/relational-vs-non-relational-databases-part-2/>
* <http://stevekrenzel.com/finding-friends-with-mapreduce>
* <https://cwiki.apache.org/confluence/display/Hive/>
* <https://en.wikipedia.org/wiki/XML>