**Բովանդակություն**

Ներածություն………………………………………………………………………․․………

Գլուխ 1 Գրականության ակնարկ………………………………………………․․……….

1.1 Գոյություն ունեցող նմանատիպ գործիքամիջոցների ուսումնասիրություն․․

1․2 Օգտագործվող գործիքամիջոցների նկարագրություն………………․․․․․․․․․․․․

1.3 Խնդրի դրվածք……………………………………………………………………․․

Գլուխ 2 Նախագծի աշխատանքի նկարագրություն ․․……………………………․․․․

## 2․1 Սերվերային աշխատանքի կազմակերպում․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․

## 2․2 Տվյալների բազայի կառավարման համակարգեր…………………………․․․․

## 2․3 Օգտագործված տվյալների բազաների կառուցվածքը…………………………

## 2․4 Իրական ժամանակի ռեժիմում հաղորդագրություններ փոխանակելու համակարգ․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․

2․5 Ալգորիթմի նկարագրություն․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․

## Գլուխ 3 Ծրագրային համակարգի նկարագրություն…………………………………․․

3․1 Կառուցվածքային մասի նկարագրություն ․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․

3․2 Կատարման ընթացքի նկարագրություն․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․

Գլուխ 4 Տնտեսագիտությամ մաս․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․

Գլուխ 5 Բնապահպանության մաս․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․

Գլուխ 6 Կենսաանվտանգության մաս․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․

Եզրակացություն․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․

Գրականության ցանկ․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․

**Ներածություն**

Վեբ հավելված ստեղծելիս անհրաժեշտ է լինում աշխատել տվյալների հետ։ Օրինակ՝ տվյալներ, որոնք օգտագործվում են մուտքագրման ժամանակ, գեներացվում են որպես ելքային արդյունք և ընթացքում են առաջանում։ Երբ տվյալների քանակը քիչ է, կարելի է այն պահել տեքստային ֆայլերում կամ նույնիսկ երկուական ռեժիմում։ Երբ տվյալների քանակը բավականին շատ է, անհրաժեշտություն է առաջանում գտնել տվյալները պահպանելու և հեշտությամբ օգտագործելու լուծում ։ Օրինակ՝ ստեղծվել է վեբ խանութ և պետք է հետևել հազարավոր ապրանքների, պահել բոլոր ապրանքների մասին տեղեկատվություն և գուցե նույնիսկ յուրաքանչյուր ապրանքի պատկերը: Տվյալների քանակը կարող է նույնիսկ գերազանցել համակարգչի հիշողության ծավալը, ուստի այն չի կարող պահվել հիշողության մեջ:

Կարող է դժվարություններ առաջանալ նաև անհրաժեշտ տվյալները գտնելիս : Եթե անհրաժեշտ է գտնել որևէ տվյալ 10.000 գրառումների ցանկում, ուրեմն սովորաբար ստիպված պետք է անցնել ամբողջ ցուցակով և համեմատել յուրաքանչյուր գրառումը անհրաժեշտ տեղեկատվության հետ: Դա կլինի ժամանակի վատնում։ Այսպիսով, հիմնականում օգտագործվում են տվյալների բազաներ, եթե պետք է մեծ քանակությամբ տվյալներ պահել և անհրաժեշտ է գտնել տվյալները համեմատաբար արագ: Երբ անհրաժեշտ է օգտվել այդ տվյալներից տարբեր հարթակներում, առաջանում են շատ ավելի բարդ խնդիրներ։

Միգրացիան ձեռնարկատիրական և վեբ հավելվածների տեսանկյունից նշանակում է տեղափոխել մի պլատֆորմից մյուսը։ Տվյալների բազաների տեղափոխումը հատկապես բարդ է, քանի որ կան ծրագրային հարթակի փոփոխման դժվարություններ։ Դրանցից են որոշ հին ֆունկցիաների և հատկանիշների բացակայությունը կամ այլ կերպ աշխատելը, և որոշ նոր հնարավորությունների ի հայտ գալը, որոնցից հնարավոր է օգտվել:

Տվյալների բազաների հետ աշխատելիս որոշ ծրագրավորողներ փորձում են ստեղծել տվյալների բազայից անկախ ծրագրեր, հատկապես օգտագործելով ORM- ները (օբյեկտ ռելացիոն քարտեզագրողներ): Առաջին հայացքից սա առավել արդյունավետ տարբերակ է՝ կառուցել հավելվածը միայն ստանդարտ բաղադրիչներներ ու ֆունկցիաներ օգտագործելով, ապա հեշտությամբ տեղափոխել այլ հարթակ : Ցավոք, այդ լուծումը այնքան էլ պարզ չէ: Տվյալների բազայից անկախ հավելվածները պարտադիր աշխատում են բոլոր տվյալների բազայի հարթակների ամենացածր ընդհանուր համատեքստում, այսպիսով՝ իջեցնելով այն հարթակի արտադրողականության և ֆունկցիոնալության մակարդակը, որը օգտագործվում է: Վեբ ծրագրերի ճնշող մեծամասնությունը ի վերջո բախվում է որոշ կատարողական խնդիրների հետ՝ կապված տվյալ ռելացիոն տվյալների բազայի նախնական կառավարման համակարգի լուծումներից։ Հաճախ անհրաժեշտ է լինում կատարել տվյալների միգրացիա մի համակարգից մյուսը։

Ահա մի քանի դեպքեր.

1. Ստեղծել հավելված, որին անհրաժեշտ է մի քանի տարբեր տվյալների բազաների աջակցություն՝ օգտագործողների կայքերի համար։
2. Ստեղծել հավելված և մշակման փուլում թեստավորել հայեցակարգին համապատասխան լինելը: Հետագայում հնարավոր է փոխարինել այլ տվյալների բազայով:
3. Չսահմանափակել օգտագործողներին, չնայած ներկայումս հավելվածն անհրաժեշտ է միայն մեկ հարթակի համար:

Սրանք բոլորը լավ հիմքեր են տալիս մտածելու տվյալների բազաների միգրացիոն գործիք ստեղծելու համար:

**Գլուխ 1 Գրականության ակնարկ**

**1․1 Գոյություն ունեցող նմանատիպ գործիքամիջոցների ուսումնասիրություն**

Տվյալների բազաների միգրացիա կազմակերպող այլ համակարգերում կան մի քանի ռազմավարական եղանակներ։ Ահա դրանց ուսումնասիրության արդյունքում ընտրված ամենաօպտիմալ տարբերակը։

Նախ ընտրվում է ճիշտ RDBMS- ը։ Նախքան տվյալների բազայի տեղափոխումը DBMS հարթակ, պետք է ուշադիր գնահատել եւ ընտրել համապատասխան մատակարարը: Չնայած հիմնական գործառույթները տարածված են, շատ համակարգերներ տարբեր են իրենց աջակցությամբ.

* Սիմետրիկ բազմապատկման (SMP) եւ զանգվածային զուգահեռ վերամշակման (MMP) ճարտարապետություն:
* Ընթացակարգային լեզու:
* Տվյալների բեռնման եւ բեռնաթափման համար ծառայություններ, ինչպես նաեւ փոփոխված տվյալների հավաքագրում:
* Աուդիտի կառավարում:
* Արտադրողականության եւ բեռնման կառավարում: Այս ոլորտում ուշադրություն դարձնող որոշ հատկանիշներ ներառում են նյութականացված տեսակետներ, ինդեքսավորում, աստղային միավորումներ եւ ռեսուրսների բաշխում:
* Հատուկ տվյալների տեսակները եւ դրանց սահմանափակումները:
* Աջակցություն առցանց վերլուծական մշակման եւ ETL- ին:

Պլատֆորմ ընտրելուց հետո տվյալների բազայի միգրացիոն գործընթացը պետք է բաղկացած լինի հետեւյալ քայլերից.

1. Սահմանել միգրացիոն ռազմավարություն, որը ներառում է փուլային (կրկնող), սկզբնական տվյալների ռազմավարություն, տվյալների որակի կառավարում, վավերացման մեթոդներ եւ ժառանգական համակարգերի հեռացում:
2. Ծրագրային բազայի օբյեկտների նախագծում, որը հիմնված է նպատակային պահանջների վրա եւ թիրախային DBMS հարթակի լավագույն փորձի հիման վրա:
3. Կատարել հնացած տվյալների համակարգերի հակառակ տեխնիկա, որպեսզի ոչ միայն հասկանալի լինի ռեկորդի կառուցվածքը, այլեւ ռեկորդ ստեղծելու տրամաբանությունը, քանի որ այն ոչ միայն տվյալների, այլ նաեւ այն տրամաբանությունն է, որը նրանք ներկայացնում են, որոնք պետք է փոխանցվեն նոր պլատֆորմին:
4. Սկզբնական և նպատակային կառուցվածքների համապատասխանեցում, երկու տիպերի տվյալների կառուցվածքների ձևափոխման կանոնների սահմանում։
5. Աղբյուրի տվյալների պրոֆիլացում, նպատակային կառույցների հետ տեխնիկական եւ փոխգործունակության հասկանալու համար:
6. Միգրացիոն գործընթացը, որը բաղկացած է ժառանգական համակարգերից տվյալների վերծանումից, ստուգումից, մաքրումից, թիրախային տվյալների ձեւաչափի համաձայն դրանց իրականացումից թիրախային տվյալների բազայի վրա:
7. Միգրացիայի սցենարներ ստեղծելու փոքր քանակությամբ տվյալների համար, որոնք չեն պահանջում խոշոր փոփոխություններ: Սակայն ETL գործիքները կարող են օգտակար լինել ավելի բարդ փոխակերպումների համար:
8. Արտադրության որակի եւ ծավալների տվյալների եւ վավերացման արդյունքների օգտագործմամբ կատարել փորձարկում:
9. Գործարկել տվյալների միգրացիայի սցենարներ արտադրական միջավայրում եւ ամբողջական անցում դեպի նպատակային RDBMS:

Արդյունավետ տվյալների միգրացիայի ցուցանիշները

Տվյալների բազայի միգրացիայի ծրագրի հաջողությունը հիմնված է մի քանի այլ գործողությունների վրա.

* Տվյալների կառավարում եւ միգրացիայի ծրագրային գործընթացի սկզբնավորման ժամանակ փոփոխության կառավարում, տվյալների որակի, նախնական տվյալների միգրացիայի, տվյալների քարտեզագրման կոնֆլիկտների եւ փորձարկման տվյալների պահանջները կարգավորելու համար:
* Միգրացիան չպետք է դիտվի որպես «տեխնոլոգիական վարժություն»: Նոր գործառույթը պետք է բացահայտվի և իրականացվի համակարգում`գործարար գնումների խթանման համար:
* Միգրացիան պետք է լինի վերահսկվող եւ կրկնվող գործընթաց, որը պետք է կատարվի փուլային, վերափոխման եղանակով:
* Նպատակային տվյալների կառուցվածքը չպետք է լինի աղբյուրի կրկնօրինակը, այլ պետք է հիմնված լինի թիրախային հարթակի կիրառման պահանջներին եւ լավագույն փորձին:
* Հատուկ ուշադրություն պետք է դարձնել վարպետության տվյալների վրա, քանի որ RDBMS- ն նպաստում է ավելի լավ վարպետության տվյալների կառավարմանը: Միգրացիոն գործընթացում պետք է հեռացվեն հիերարխիկ մոդելի կրճատումները:
* Օրենսդրական պահանջների հիման վրա համապատասխան պատմությունը պետք է պահպանվի, քանի որ այն ոչ միայն նվազեցնում է ջանքերը տվյալների շուկայում եւ տեղափոխվելու համար, այլեւ նվազեցնում է փոխակերպման բարդությունը:
* Փորձարկման տվյալների ռազմավարությունը պետք է հիմնված լինի տվյալների որակի վրա: Թեստավորման համար օգտագործվող տվյալների ձևաչափը եւ ծավալը չպետք է պատահական լինի եւ պետք է հիմնված լինի տվյալների պահանջի վրա, տվյալների հստակեցման արդյունքների եւ միգրացիայի կատարողականության քննադատության վրա:

Համապատասխան եւ այլ տվյալների բազաների միջեւ հիմնարար տվյալների ճարտարապետական ​​տարբերությունների շնորհիվ, ժառանգական տվյալների միգրացիան կարող է լինել բարդ խնդիր, ռիսկերով լի: Այնուամենայնիվ, դրանք կարելի է կառավարել եւ մեղմել լավ սահմանված տվյալների բազայի միգրացիայի ռազմավարության եւ վավերացման պլանի հետ:

**1․2 Գործիքամիջոցների նկարագրություն**

* SQL լեզու

SQL-ը կամ հարցումների կառուցվածքավորված լեզուն ստանդարտ լեզու է, որը նշանակված է տվյալների ռելյացիոն բազաների հետ փոխգործակցելու համար։ Դրա նախատիպը մշակվել էր IBM-ում, դոկտոր Կոդդի (E.F.Codd) “Ռելյացիոն մոդելի օգտագործումը տվյալների բաժանված բանկերում” հոդվածի հիման վրա, յոթանասունական թվականներին։ SQL-ի առաջին ծրագրային արտադրանքը՝ Oracle ՏՌԲԿՀ-ն (տվյալների ռելյացիոն բազայի կառավարման համակարգ), ստեղծվեց 1979 թ. Relational Software, Inc. կորպորացիայի (հետագայում այն վերաանվանվեց Oracle Corporation) կողմից։ Ծրագրային արտադրանքի շուկայում այդ ՏՌԲԿՀ-ն մեծ հաջողությունը ստիպեց մյուս խոշոր ֆիրմաներին (Sybase, Informix, Microsoft) թողարկել SQL-ի վրա հիմնված ՏՌԲԿՀ-ների սեփական տարբերակները։ Քանի որ SQL-ը այն ժամանակ նորույթ էր, և դեռ ստեղծված չէին ստանդարտներ, ապա յուրաքանչյուր ֆիրմա արտադրում էր ՏՌԲԿՀ SQL-ի սեփական իրացումով։ Միայն 1989 թ. ANSI-ն (American National Standards Institute) մշակեց SQL-ի համընդհանուր ստանդարտը։ Սակայն այն հաշվի չէր առնում SQL-ի տեխնիկական կառուցվածքը, և անցումը լեզվի մեկ իրացումից մյուսին բավականաչափ բարդ էր։ 1992 թ. ստեղծվեց ստանդարտ, որը ընդգրկում է SQL-ի հնարավորին չափ շատ ընդլայնումները։ Այն անվանում են SQL92 կամ SQL2։ Նշենք, որ SQL2-ի մեծ ծավալի պատճառով ՏՌԲԿՀ-ները, որոնք իրացնում են լրիվ ստանդարտը, բավականաչափ բարդ են և աշխատատար։

* SQL լեզվի հրամանների տեսակները

SQL-ն ունի հրամանների վեց հիմնական խմբեր՝

1. DDL (Data Definition Language) - տվյալների սահմանման լեզվի հրամաններ։
2. DML (Data Manipulation Language) - տվյալների փոխակերպումների մանիպուլյացիաների) լեզվի հրամաններ։
3. DQL (Data Query Language) – հարցումների լեզվի հրամաններ։
4. DCL (Data Control Language) – տվյալների ղեկավարման լեզվի հրամաններ։
5. DAC - Տվյալների ադմինիստրացման հրամաններ։
6. TCC - Տրանզակցիաների ղեկավարման հրամաններ։

1) Տվյալների սահմանման հրամանների միջոցով օգտվողները կարող են ստեղծել և փոփոխել տվյալների օբյեկտների կառուցվածքը, օրինակ՝ ստեղծել և հեռացնել աղյուսակներ։ Այդ խմբի հիմնական հրամանները հետևյալներն են՝

* create table,
* alter table,
* drop table։

2) Այս խմբի հրամանները օգտագործվում են բազայի օբյեկտների (օրինակ՝ աղյուսակների) որոշ բնութագրիչները փոփոխելու համար։ Հիմնականում օգտագործվում են հետևյալ երեք հրամանները

* insert,
* update,
* delete։

3) Հարցումների լեզուն ընդգրկում է ընդամենը մեկ հրաման՝

* select

Այդ հրամանը իր բազմաթիվ օպցիաներով և նախադասություններով (այսինքն՝ բառբանալիներով և օպերատորներով, որոնց միջոցով կառուցվում են ընտրության որոշակի պայմաններ) օգտագործվում է տվյալների ռելյացիոն բազային հարցումներ ձևավորելու համար, ընդ որում՝ հարցումները կարող են լինել ինչպես պարզ և անորոշ, այնպես էլ բարդ և լիովին որոշակի (կոնկրետ)։ Հարցումը՝ դիմում է տվյալների բազային՝ համապատասխան ինֆորմացիա ստանալու նպատակով։

4) Տվյալների ղեկավարման լեզվի հրամաններն են՝

* alter password,
* grant,
* revoke,
* create synonym։

Որպես կանոն, այդ հրամանները օգտագործվում են տվյալների հասանելիության հետ կապված օբյեկտների ստեղծման համար և, բացի այդ, թույլ են տալիս վերահսկողություն սահմանել օգտվողների միջև արտոնությունների բաշխման նկատմամբ։

5) Տվյալների ադմինիստրացման հրամաններ են՝

* start audit,
* stop audit։

Դրանք թույլ են տալիս հսկել կատարվող գործողությունները և վերլուծել տվյալների բազայի գործառնությունները։ Պետք չէ շփոթել տվյալների ադմինիստրացում և բազայի ադմինիստրացում գաղափարները։ Բազայի ադմինիստրացումը տվյալների բազայի ընդհանուր ղեկավարումն է և ի նկատի ունի բոլոր մակարդակների հրամանների օգտագործում։

6) Տրանզակցիաների կառավարման հրամաններն են՝

* commit,
* rollback,
* savepoint,
* set transaction։

SQL լեզվի իրացումների մեծամասնությունը թույլ է տալիս գրանցել հրամանների օպերատորները ինչպես մեծատառերով, օրինակ՝ CREATE TABLE, այնպես էլ փոքրատառերով՝ create table։

* SQL SELECT հարցումներ

SQL լեզվի օպերատորներից ամենաբարդ կառուցվածքը ունի SELECT օպերատորը, որը ապահովում է տվյալների բազաից տվյալի դուրս բերումը։ Բազաից ամբողջ տվյալները տեսնելու համար SELECT օպերատորի ներսում դնում ենք \* նշանը,իսկ ցանկալի սյունը տեսնելու համար պետքե նշել սյան անունը։ Առանձնահատուկ (օրինակ,եթե մենք ցանկանում ենք ստանալ տարաբնույթ կոմբինացիաներ,այլ ոչ նույն ) տող ստնալու համար օգտագործում ենք DISTINCT բանալի բառը։ Ամբողջն ընդգրկելու համար օգտագործում ենք ALL բանալի բառը: Select-ում գրված ցանկացած դաշտ կարող ենք դասավորել մեզ ցանկալի ձևով։ Դրա համար օգտագործում ենք ORDER BY + դաշտի անուն, որը գրվում է SELECT-ի վերջում։ Նվազման կարգի դասավորելու համար DESC բանալի բառն ենք գրում, աճման համար՝ ASC :

* Համեմատման օպերատորներ

Համեմատման օպերատորներ` =; >; <; >=; <=; <> !=; փոքր չէ !<; մեծ չէ !>: Համեմատման գործողությունների արդյունքում վերադարձվում են 3 նշանակություններ` TRUE, FALSE, UNKNOWN: UNKNOWN նշանակություն վերադարձվում է այն դեպքում, երբ օպերատորը ունի 1 կամ 2 NULL նշանակությամբ արժեքներ և SET ANSI NULL աշխատել է ON բանալիով: Եթե SET ANSI NULL աշխատում է OFF բանալիով և 2 արտահայտություններն էլ NULL են, ապա օպերատորը վերադարձնում է TRUE արժեք:

* Տրամաբանական օպերատորներ

Այս խումբը սահմանվում է տվյալների բազմության կամ ընտրույթի նկատմամբ, վերադարձնում է TRUE կամ FALSE նշանակություններ. ALL- վերադարձնում է TRUE նշանակություն, եթե ամբողջ բազմության նկատմամբ համեմատումը վերադարձնում է TRUE AND- վերադարձնում է TRUE, եթե Boolean արտահայտությունները բոլորը TRUE են: ANY- վերադարձնում է TRUE, եթե բազմության նկատմամբ կատարված համեմատման արդյունքներից որևէ մեկը TRUE է: BETWEEN - վերադարձնում է TRUE, եթե ըստ ներկայացված գրելաձևի օպերանդը գտնվում է անհրաժեշտ միջակայքում: EXISTS- վերադարձնում է TRUE, եթե ընտրույթը ձևավորել է գոնե մեկ գրառում: IN- վերադարձնում է TRUE, եթե օպերանդը հավասար է ներկայացված բազմության կամ ընտրույթի տարրերից գոնե մեկին: LIKE -վերադարձնում է TRUE, եթե օպերանդը բավարարում է ներկայացված գրելաձևին: NOT -վերադարձնում է Boolean օպերատորի ինվերս նշանակությունը: OR - վերադարձնում է TRUE, եթե Boolean արտահայտություններից որևէ մեկը TRUE է: SOME - Ճիշտ նույննէ ինչ որ ANY -ն:

* SQL միավորման հարցումներ (JOIN)

From-ից հետո կարող է օգտագործել միավորման բաց օպերացիա, որը միավորում է մեկից ավելի աղյուսակներ։ Միավորումները լինում են ներքին (INNER) և արտաքին (OUTER), այս սպասարկվող բառերը կարող ենք չգրել, քանզի արտաքինը լինում է LEFT, RIGHT և FULL, իսկ ներքինը` JOIN ։ Պրեդիկատը որոշում է միավորման տողի պայմանը տարբեր աղյուսակների մոտ։ INNER JOIN - նշանակում է որ արդյունքի հավաքածուի մեջ կհայտնվեն միայն այն տողերը , որի համար պրեդիկատի արժեքը TRUE է ։

Ներքին միավորման LEFT JOIN-ն նշանակում է ,որ բացի տողից ,որի համար կատարվում է պրեդիկատի հարցումը,ընդհանուր ցանկ անցնում է բոլոր առաջի (ձախ մասի) աղյուսակը։

* Աղյուսակների միավորման հարցումներ (UNION)

UNION - Միավորում է երկու կամ ավելի հարցումների արդյունքները մի հավաքածուի մեջ, որը ներառում է միավորման բոլոր հարցումներին պատկանող բոլոր տողերը: UNION գործողությունը տարբերվում է երկու կամ ավելի աղյուսակների միավորման (սյունակների) JOIN գործողությունից: UNION , EXCEPT, INTERSECT գործողությունների հիմնական կանոններն են ՝  Բոլոր հարցումներում սյունակների քանակը և հերթականությունը պետք է լինի նույնը:  Տվյալների տիպերը պետք է լինեն համատեղելի: EXCEPT - Հարցման առաջին բազմությունից հեռացնում է երկրորդ հարցման բազմությունը։ Այս դեպքում հաշվի է առնվում SELECT հրամանում նշված բազմությունների հերթականությունը։ INTERSECT - Վերադարձնում է բոլոր հարցումների բազմություններում առկա նույն բազմությունը։ Այս դեպքում հաշվի չի առնվում SELECT հրամանում նշված բազմությունների հերթականությունը։

* Ենթահարցումներ

Այն կառուցվածքը, որտեղ մեկ կամ մի քանի SELECT օպերատորներ ներդրված են մեկ այլ SELECT օպերատորի WHERE օպերատորի մեջ կոչվում է ենթահարցում: Ենթահարցման ամենաառաջին SELECT օպերատորը կոչվում է արտաքին հարցում, իսկ WHERE օպերատորից հետո օգտագործվող հարցումը կոչվում է ներքին հարցում: Նախ կատարվում է ներքին հարցումը, ապա նրա արդյունքը փոխանցվում է արտաքին հարցմանը: Յուրաքանչյուր ներքին հարցումը կարող է պարունակել մեկ այլ ներդրված հարցում: Այդ տիպի հարցումները կոչվում են բազմամակարդակ ներդրված ենթահարցումներ: Ներքին ներդրված հարցումների մաքսիմալ քանակը որոշվում է ՏԲ-ի SELECT հրամանի համար նախատեսված հիշողության ծավալով: Գոյություն ունեն 2 տիպի ենթահարցումներ՝

* Սկալյար ենթահարցում, որը վերադարձնում է միայն մեկ արդյունք:
* Աղյուսակային ենթահարցում, որը վերադարձնում է բազմաթիվ արժեքներ:
* Ներկայացումներ (VIEWS)

Ներկայացումը(VIEW)- ձևավորում է տվյալների բազայի տրամաբանական կառուցվածքը։ Ներկայացումը օգտագործողի համար հանդես է գալիս որպես աղյուսակ: Ներկայացումը իրականացվում է որպես հիշողության մեջ պահվող հարցում, որի հիման վրա էլ իրականացվում է տվյալների ընտրումը տարբեր տվյալների բազաների աղյուսակներից։ Ներկայացումների առավելությունները՝

* Ապահովում են տեղեկատվության գաղտնիությունը, քանի որ թույլ են տալիս ցուցադրել միայն անհրաժեշտ տեղեկատվությունը, թաքցնելով որոշ դաշտեր:
* Պարզեցնում են տվյալների ներկայացումը, քանի որ օգտագործողը աշխատում է ներկայացման հետ որպես մեկ աղյուսակ, որը ստեղծված է մի քանի աղյուսակից տվյալների ընտրության հիման վրա։
* Վերահսկում են տվյալներին դիմելու մուտքի իրավունքները, օրինակ, աղյուսակի որոշակի դաշտերին հարցումների րականացման թույլատրություն տալու փոխարեն ավելի հեշտ է թույլ տալ յարցումների իրականացումը ներկայացումների միջոցով:

Ներկայացման օգնությամբ կարելի է սահմանափակել աղյուսակի որոշ դաշտերի մուտքը, այս դեպքում ասում են, որ աղյուսակի վրա կիրառված է ուղղահայաց ֆիլտր, այսինքն սահմանափակված է աղյուսակի որոշ դաշտերի հասանելիությունը: Եթե ընտրման հարցման կոդի մեջ սահմանված է գրառումների ընտրման պայման, ապա ասում են, որ աղյուսակի վրա կիրառված է հորիզոնական ֆիլտր։

* Պահպանվող պրոցեդուրաներ

Stored procedures դա Ttansact-SQL-ի օպերատորների խ-մբ է, որը պահվ-մ է սերվերի վրա: Այն իրականացն-մ է կլիենտի և սերվերի միջև գործող-թյ-նների կազմակերպ-մը: Stored procedures –ի օգն-թյամբ ենթադր-մ է հետևյալը` կլիենտը իրականացն-մ է սերվերի վրա պահվող հրամանների խմբի կանչը, սերվերը կատար-մ է այդ հրամանները և վերադարձն-մ է կլիենտին արդյ-նքը։ Stored procedures –ը ՏԲ-ի ինքն-ր-յն օբյեկտներ են, նրանք նման են տարբեր ծրագրավորման լեզ-ների պրոցեդ-րաներին և թ-յլ են տալիս՝ Ներառել տարբեր օպերատորներ և կանչ կազմակերպել այլ Stored procedures-ից; Ընդ-նել մ-տքային պարամետրեր և վերադարձնել արժեքներ ելքային պարամետրերի տեսքով;

* Տրիգերներ

Տրիգերները հատուկ տեսակի պահվող ընթացակարգեր են (Stored procedurs),որոնք ավտոմատ գործարկվում է SQL Server-ի կողմից որևէ աղյուսակի փոփոխման ժամանակ հետևյալ երեք օպերատորներից որևէ մեկով՝ Update, Insert, կամ Delete. Տրիգերները, ինչպես և մյուս Stored procedures-ները կարող են պարունակել T-SQL-ի պարզ կամ բարդ օպերատորները։ Ի տարբերություն այլ տեսակների Stored procedures-ների, տրիգերները գործարկվում են ավտոմատ կերպով, տվյալների նշված փոփոխությունների ժամանակ: Երբ իրականացվում է տրիգերի գործարկումը, ասում են, որ այն ակտիվաել է (fire): Տրիգերը ստեղծվում է ՏԲ-ի մեկ աղյուսակի համար։ Տրիգերները չի կարելի ստեղծել ժամանակավոր կամ համակարգային աղյուսակների համար, այլ միայն օգտագործողի կողմից նշված աղյուսակների կամ ներկայացումների համար։ Աղյուսակը, որի համար սահմանված է տրիգերը, կոչվում է տրիգերի աղյուսակ։ Տրիգերի ակտիվացման ժամանակ արդյունքները, եթե նրանք կան, վերադարձվում են կանչվող ծրագրին, ինչպես և stored procedure –ի դեպքում։ Տրիգերների տեսակները՝

* Update
* Insert
* Delete

Ըստ տրիգերների պահվածքի դրանք լինում են 2 տիպի՝

* AFTER` տրիգերը կատարվում է իրեն կանչող հրամանների հաջող կատարումից հետո։ Եթե հրամանները որևէ պատճառով չեն կարող հաջողությամբ ավարտվել, տրիգերը չի կատարվում։ FOR բառի կիրառումը համարժեք է AFTER վարքագծին։ AFTER վարքագծով տրիգերները կարող են ստեղծվել միայն աղյուսակների համար։
* INSTEAD OF՝ կանչվում է հրամանների կատարման փոխարեն։ Այս վարքագծով տրիգերները կարող են ստեղծվել ինչպես աղյուսակների այնպես էլ ներկայացումների համար։

Տրիգերները գործարկվում են միայն իրեն ակտիվացնող օպերատորի ավարտից հետո։ Եթե որևէ օպերատոր փորձում է իրականացնել որևէ գործողություն, որը խախտում է աղյուսակի որևէ սահմանափակում կամ հանդիսանում է որևէ ուրիշ սխալի պատճառ, ապա նրա հետ կապված տրիգերը չի ակտիվանում։ Տրիգերը ակտիվանում է միայն մեկ անգամ մի օպերատորի համար, նույնիսկ երբ այդ օպերատորը ազդում է տվյալների մի քանի գրառումների վրա։

* Java ծրագրավորման լեզու

Java -ն պլատֆորմից անկախ, ընդհանուր նշանակության ծրագրավորման լեզու է, որը զուգահեռ, դասերի վրա հիմնված, օբյեկտ – կողմնորոշված և հատուկ մշակված է, որպեսզի ունենա ինչքան հնարավոր է շատ իրագործումներ։ Այն նախատեսված է, որպեսզի «Գրվի մեկ անգամ, աշխատի ամենուր» ("Write once, run anywhere"), որը նշանակում է, կոմպիլացված Java կոդը կարող է կատարվել բոլոր Java աջակցող պլատֆորմների վրա առանց նորից կոմպիլացվելու։ Java ծրագրերը կոմպիլացվում են բայթկոդ-ի, որոնք կարող են իրագործվել ցանկացած Java վիրտուալ մեքենայի (JVM) վրա, անկախ համակարգչային ճարտարապետությունից։ Java-ն ստեղծվել է Ջեյմս Գոսլինգի կողմից Sun Microsystems – ում (որը հետագայում Oracle կազմակերպությունը գնեց) և ներկայացվեց 1995 թ. որպես Java պլատֆորմի հիմնական մաս։ Այն ծրագրավորվել է C++-ի միջոցով։ Լեզվի սինտաքսիսը մեծ մասամբ համընկնում է C և C++ լեզուների հետ, բայց այն ունի ավելի քիչ ցածր – մակարդակի միջոցներ, քան նրանցից յուրաքանչյուրը։ Ի տարբերություն C++-ի Java-ն ունի նաև ինտեգրված լայն կլասսների գրադարան (օրինակ բազմապրոցեսային, համացանց, ֆայլային և այլն), որոնք ի շնորհիվ պլատֆորմային անկախության աշխատում են տարատեսակ օպերացիոն համակարգերում։ Որպեսզի Java լեզվով ստեղծված ծրագիրը կարողանա աշխատել ձեր համակարգչում կամ շարժական էլեկտրոնային սարքում, դուք պետք է ունենաք համապատասխան միջավայր՝ JRE (Java Runtime Environment)։

* Տեխնիկական նկարագրություն

Գոյություն ուներ հինգ հիմնական նպատակ Java լեզվի ստեղծման ժամանակ։

* Այն պետք է լինի «պարզ, օբյեկտ – կողմնորոշված, ծանոթ»
* Այն պետք է լինի «հուսալի և անվտանգ»
* Այն պետք է լինի «ճարտարապետությունից անկախ և տեղափոխելի»
* Այն պետք է կատարվի «բարձր արագությամբ»
* Այն պետք է լինի «ինտերպրետացվող և դինամիկ»

Java-ի տեղափոխելությունը պայմանավորված է նրանով, որ կոմպիլացվելուց հետո կոդն անմիջապես չի դառնում միջավայրին համապատասխան մեքենայական կոդ, այլ վերածվում է միջանկյալ կոդի, որը կոչվում է Java բայթկոդ։ Վերջինս աշխատեցվում է Java վիրտուալ մեքենայի կողմից, որն արդեն հատուկ է տվյալ միջավայրին։ Սովորաբար բոլոր օգտագործողների սարքերում էլ տեղադրված է Java Runtime Environment-ը (JRE), որն էլ ապահովում է Java վիրտուալ մեքենայի առկայությունը։ Վերջինիս աշխատանքը ներկայացված է նկարում։ Ինչպես երևում է նկարից՝ Java վիրտուալ մեքենան նախատեսված է ոչ միայն Java-ի, այլ նաև այլ ծրագրավորման լեզուների համար (նկարում ներկայացված է Python-ը)։ Սովորաբար Java-ով գրված ծրագրերը C++-ով գրվածների հեմեմատ ունեն ավելի դանդաղ աշխատելու և շատ հիշողություն օգտագործելու համբավ։ Բայց Java-ի Just-in-time (JIT) կոմպիլացայի ներդրումից հետո ծրագրերի արագագործությունը զգալիորեն մեծացել է։ JIT-ի դերը կայանում է նրանում, որ այն դինամիկ ձևով Java բայթկոդը թարգմանում է մեքենայական կոդի, որն էլ միանգամից կատարվում է։ Ծրագրերի կատարման այս մեթոդի առավելությունը կայանում է նրանում, որ բայթկոդը անկախ է օպերացիոն համակարգից և սարքավորումից, հետևաբար Java ծրագրերը կարելի է կատարել ցանկացած սարքի վրա, որի համար գոյություն ունի համապատասխան վիրտուալ մեքենա։ Մյուս կարևոր առանձնահատկությունը կայանում է ճկուն անվտանգության համակարգում, որի շրջանակներում ծրագրի կատարումը ամբողջությամբ կառավարում է վիրտուալ մեքենան։ Ցանկացած օպերացիա, որը գերազանցում է ծրագրի իրավունքները (օրինակ չթույլատրված մուտք դեպի տվյալներ կամ ուրիշ համակարգչի հետ կապ), առաջացնում են ծրագրի անահապաղ դադարեցում։ Վիրտուալ մեքենայի գլխավոր թերությունը ցածր արագագործությունն է համարվում։ Որոշ կատարելագործումներ բարձրացրել Java ծրագրերի կատարման արագությունը։

* Բայթկոդի հեռարձակումը մեքենայական կոդի դինամիկ ձևով ծրագրի կատարման ժամանակ (JIT տեխնոլոգիա, Just-in-time (JIT))։
* Պլատֆորմ – կողմնորոշված կոդի լայն օգտագործում (native կոդ) ստանդարտ գրադարաններում։
* Սարքերի միջոցներ, որոնք ապահովում են բայթկոդի արագացված մշակում (օրինակ Jazelle տեխնոլոգիան, որը կիրառվում է որոշ ARM պրոցեսսորների համար)։

Java-ն օգտագործում է աղբահավաք՝ (անգլ.՝ Garbage collector - GC) հիշողության ավտոմատ կազմակերպման համար։ Երբ ծրագրի աշխատանքի ժամանակ որևէ օբյեկտ չի օգտագործվում, այսինքն չկա նրա վրա ցուցող որևէ հղում, ապա աղբահավաքը ջնջում է այն հիշողությունից։ Չնայած այս ավտոմատացված կազմակերպմանը՝ հիշողության արտահոսքի (անգլ.՝ Memory leak) վտանգը դեռ մնում է։ Օրինակ հիշողության արտահոսք կարող է լինել, եթե ոչ պետքական օբյեկտի վրա ցուցող հղում (անգլ.՝ Refernce) մնա։ Ի տարբերություն C++-ի Java-ն հնարավորություն չի տալիս ցուցիչների (անգլ.՝ Pointer) հետ աշխատելու, ձեռքով հիշողությունն ազատելու, խելացի ցուցիչներ (անգլ.՝ Smart pointer) օգտագործելու և այլն։ Աղբահավաքը կարող է աշխատել ցանկացած պահի, իդեալական դեպքում այն աշխատում է համեմատաբար պարապուրդի մեջ եղած ժամանակ։ Երաշխավորված է, որ այն կաշխատի, երբ բավարար հիշողություն չլինի նոր օբյեկտ ստեղծելիս։ Սա թույլ է տալիս աղբահավաքին տեղափոխել օբյեկտների հասցեները և տալիս է տիպերի ապահովություն։ Ինչպես C++-ում և որոշ օբյեկտ-կողմնորոշված ծրագրավորման լեզևներում, Java - ում պրիմիտիվ տիպերը(int, double, boolean, char) պահվում են դաշտերում (օբյեկտների համար), ստեկում (անգլ.՝ Stack) (ֆունկցիաների համար), այդ պատճառով դրանց համար աղբահավաք չի աշխատում։ Սա Java - ի ստեղծողների գիտակցված որոշումն է, որի նպատակն է բարձրացնել արագագործությունը։ Java-ն ունի մի քանի տեսակի աղբահավաքներ։ Java - ով գրված ծրագրերի 90% -ի համար Concurrent Mark-Sweep (անգլ.՝ Concurrent mark sweep collector) աղբահավաքը բավարար է։ Oracle -ը մտադրված է փոխարինել այն Garbage-first collector (G1) աղբահավաքով։

* Spring Boot Framework

Spring Boot-ը Spring-ի convention-over-configuration լուծումն է: Spring-ի հիմքով ավտոնոմ հավելված ստեղծելու համար, որը կարելի է ուղակի աշխատեցնել։

Առավելություններն են՝

* Ավտոնոմ (ինքնավար) Spring հավելվածների ստեղծում,
* ՈՒղղակիորեն ներդրված են Tomcat կամ Jetty՝ կարիք չկա տեղակայելու WAR ֆայլերը,
* Ապահովել Project Object Models (POMs), Maven կոնֆիգուրացիաները հեշտացնելու համար,
* Հնարավորության դեպքում ավտոմատ կարգաբերել Spring-ը,
* Ապահովել արտադրության պատրաստի ֆունկցիաներ, ինչպիսիք են չափիչները, առողջապահական ստուգումները և այլն,
* Բացարձակապես ոչ մի կոդի ստեղծում և XML ձևակերպման պահանջ։

Spring Boot- ը բաց կոդով Java- ի վրա հիմնված framework է, որն օգտագործվում է միկրո ծառայության ստեղծման համար: Այն մշակվում է առանցքային թիմի կողմից: Spring Boot-ը պարունակում է համալիր ենթակառուցվածքային աջակցություն միկրո ծառայության զարգացման համար եւ հնարավորություն է տալիս զարգացնել ձեռնարկության պատրաստած ծրագրեր, որոնք կարող եք «պարզապես գործարկել»:

Միկրո ծառայությունը ճարտարապետություն է, որը թույլ է տալիս մշակողին մշակել եւ տեղակայել ծառայությունները ինքնուրույն: Յուրաքանչյուր ծառայություն վարում է իր սեփական գործընթացը, եւ սա հասնում է բիզնեսի ծրագրերին աջակցելու թեթեւ մոդելին:

Spring Boot- ն լավ հարթակ է ապահովում Java- ի մշակողների համար, որոնք կկարողանան զարգացնել ինքնուրույն եւ արտադրական ծրագրերը:

Spring Boot- ի նպատակն է ապահովել արագ եւ ուղղակիորեն Java- ի հավելվածներ ներկառուցված սերվերի միջոցով լռելյայն, այն օգտագործում է Tomcat- ի ներդրված տարբերակը` դրանով իսկ վերացնելով Java EE կոնտեյների կարիքը: Գարուն Boot- ի օգնությամբ մենք կարող ենք տրամադրել REST- ի ծառայությունները, անկախ միմյանցից, ճիշտ այնպես, ինչպես առաջարկվում է միկրոալիքային ճարտարապետության մեջ, այնպես որ բաղադրիչների ցանկացած պահպանումով մենք այլեւս չենք վերաբերի բոլոր իր հաճախորդներին:

Այն ապահովում է Java Beans- ի, XML կոնֆիգուրացիաների եւ տվյալների բազայի գործարքների ճկուն ձեւ, հզոր խմբաքանակի վերամշակում եւ ղեկավարում է REST վերջնական կետերը:

Spring Boot –ով գործարկվող ծրագրի մուտքի կետը դասը պարունակում է @SpringBootApplication անոտացիա եւ հիմնական մեթոդը: Այն ավտոմատ կերպով վերլուծում է ծրագրի մեջ ներառված բոլոր բաղադրիչները, օգտագործելով @ComponentScan անոտացիան:

* Model–view–controller

Model-view-controller- ը օգտագործող ինտերֆեյսների ձևավորման համար սովորաբար օգտագործվող կառուցվածքային նմուշ է, որը հավելվածը բաժանում է երեք փոխկապակցված մասերի: Դա կատարվում է ներքին ինֆորմացիայի և օգտագործողի կողմից ներկայացված և ընդունված տեղեկատվության առանձնացման համար: MVC-ի դիզայնի ձևը առանձնացնում է այս բաղադրիչները, և թույլ տալիս ծրագրի արդյունավետ վերօգտագործում և զուգահեռ մշակում:

Ավանդաբար օգտագործվող desktop գրաֆիկական ինտերֆեյսների համար (GUIs), այս կառուցվածքները հայտնի են դարձել վեբ ծրագրերի և նույնիսկ բջջային, desktop և այլ հավելվածների նախագծման համար: Հայտնի ծրագրավորման լեզուները, ինչպիսիք են Java, C #, Ruby, PHP- ն վեբ հավելվածի մշակման մեջ կարող են օգտագործել MVC framework: Model-view-controller-ի բաղադրիչներն են՝

* Model-ը նմուշների կենտրոնական բաղադրիչն է: Այն հավելվածի դինամիկ տվյալների կառուցվածքն է, անկախ օգտագործողի ինտերֆեյսից: Այն ուղղակիորեն կառավարում է տվյալների, տրամաբանության և կիրառման կանոնները:
* View-ն կարող է լինել տեղեկատվության ցանկացած ելքային ներկայացում, օրինակ, աղյուսակ կամ դիագրամ: Միևնույն տեղեկատվության տարբեր տեսքերը հնարավոր են, օրինակ, կառավարման գծապատկերային աղյուսակը և հաշվապահների համար աղյուսակային տեսք:
* Երրորդ մասը կամ բաժինը, controller-ը, ընդունում է մուտքագրումը և այն ձևափոխում հրամանի model-ի կամ view-ի համար:
* Փոխազդեցություններ

Հավելվածի երեք բաղադրիչի բաժանումից բացի model–view–controller դիզայնը սահմանում է նրանց միջև փոխազդեցությունը:

* Model-ը պատասխանատու է հավելվածի տվյալների կառավարման համար: Այն controller-ից ստանում է օգտագործողի մուտքագրումը:
* View-ն մոդելի ներկայացումն է որոշակի ձևաչափով:
* Controller-ը արձագանքում է օգտագործողի մուտքագրմանը, ընդունում, վավերացնում է այն, հետո մուտքագրում է Model-ում:
* Առավելությունները
* Միաժամանակյա զարգացում - Մի քանի ծրագրավորողներ կարող են միաժամանակ աշխատել մոդելի, վերահսկիչի և դիտումների վրա:
* Բարձր համակցություն - MVC- ն հնարավորություն է տալիս վերահսկիչիn կապված գործողությունների տրամաբանական խմբավորում:
* Փոփոխության հեշտություն - Պատասխանատվությունների բաժանման պատճառով հետագա զարգացումը կամ փոփոխումն ավելի հեշտ է։
* Մոդելները կարող են ունենալ բազմաթիվ դիտարկումներ։
* Թերություններ
* Ծրագրի նավիգացիան – Նավիգացիան framework-ում կարող է բարդ լինել, քանի որ այն կատարում է բաժանումներ և պահանջում է օգտագործողներին հարմարվել MVC- ի տարանջատման չափանիշներին:
* Բազմակողմանի հետևողականություն - Մի հավելվածը երեք մասի բաժանումը առաջացնում է սփռում: Այսպիսով, մշակողները պետք է պահպանեն միանգամից բազմաթիվ հատվածների հետևողականությունը:
* Գիտելիքների շեղում - Մի քանի տեխնոլոգիաների իմացությունը դառնում է անհրաժեշտ: MVC մշակողները պետք է ունենան մի քանի տեխնոլոգիաների հմտություններ:
* Java Database Connectivity (JDBC)

JDBC-ն ծրագրավորման ինտերֆեյսն է (API) Java ծրագրավորման լեզվի համար, որը սահմանում է, թե ինչպես կարող է օգտագործողը մուտք գործել տվյալների բազա: Այն Java- ի վրա հիմնված տվյալների հասանելիության տեխնոլոգիա է, որն օգտագործվում է Java տվյալների բազայի կապի համար: Այն հանդիսանում է Java Standard Edition պլատֆորմի մի մասը, Oracle Corporation- ից: Այն տրամադրում է մեթոդներ տվյալների բազայում տվյալների հարցման և թարմացման համար, և աշխատում է ռելացիոն տվյալների բազաների հետ։

* Ֆունկցիոնալությունը

JDBC- ն թույլ է տալիս բազմաթիվ հավելվածների օգտագործում միևնույն հավելվածում: API- ն ապահովում է ճշգրիտ Java փաթեթների դինամիկ բեռնում և գրանցում դրանք JDBC դրայվեր մենեջերի միջոցով: Դրայվեր մենեջերը օգտագործվում է որպես JDBC կապի միացման գործարան:

JDBC- ի հղումներն աջակցում են հայտարարությունների ստեղծմանն ու իրականացմանը: Սրանք կարող են լինել նոր հայտարարություններ, ինչպիսիք են SQL- ի ստեղծումը, INSERT- ը, UPDATE- ը և DELETE- ը կամ հարցման հայտարարություններ, ինչպիսին է SELECT-ը: Բացի այդ, պահվող ընթացակարգերը կարող են օգտագործվել JDBC կապի միջոցով: JDBC- ը ներկայացնում է հայտարարություններ, օգտագործելով հետևյալ դասերից մեկը՝

* Statement - հայտարարությունը ուղարկվում է բազային սերվերին ամեն անգամ:
* Made status - Պատրաստված կարգավիճակը, հայտարարությունը պահվում է, և ապա կատարման ուղին որոշվում է տվյալների բազայի սերվերի վրա, որը թույլ է տալիս այն արդյունավետ կերպով իրականացնել բազմակի անգամ:
* CallableStatement - օգտագործվում է տվյալների բազայում ընթացակարգեր իրականացնելու համար:

Թարմացրեք այնպիսի հայտարարություններ, ինչպիսիք են INSERT, UPDATE և DELETE վերադարձնում է թարմացման հաշվարկ, որը ցույց է տալիս, թե քանի տող է փոփոխվել տվյալների բազայի վրա: Այս հայտարարությունները որևէ այլ տեղեկատվություն չեն վերադարձնում:

Հարցման հայտարարությունները վերադարձնում են JDBC տողի արդյունքների շարք: Փոխակերպման արդյունքների հավաքածուն ունի մետատվյալներ, որոնք նկարագրում են սյուների անունները և դրանց տեսակները:

* JDBC դրայվերներ

JDBC դրայվերները կողմնորոշիչները են (սերվերի վրա տեղակայված, այլ ոչ թե սերվերի վրա), որոնք փոխակերպում են Java- ի հավելվածի պահանջները այն արձանագրության, որը DBMS- ը կարող է հասկանալ: Դրայվերները լինում են՝

* 1-ին տեսակ, որը կանչում է տեղական մատչելի ODBC դրայվերի ներքին կոդը:
* 2-րդ տեսակ , որը հաճախորդի կողմից տվյալների բազայի մատակարարի հարազատ գրադարանն է: Այս կոդը այնուհետև կապվում է տվյալների բազայի հետ ցանցով:
* 3-րդ տեսակ , մաքուր java դրայվերը, որը կապվում է միջանկյալ սերվերի հետ, որը հետագայում կապվում է տվյալների բազայի հետ:
* 4-րդ տեսակ, մաքուր java դրայվերը, որը օգտագործում է տվյալների բազայի հարազատ արձանագրությունը:

Կա նաև ներքին JDBC դրայվերի տիպ, որը Java- ի կողմից հաստատված SQL տվյալների բազաների մեջ JRE- ով տեղադրված դրայվեր է: Այն օգտագործվում է Java- ի պահպանված ընթացակարգերի համար: Դրա օրինակ է KPRB- ն (Kernel Program Bundled) դրայվերը , որը աշխատում է Oracle RDBMS- ի հետ: "jdbc: default: connection" - ն առաջարկում է համեմատաբար ստանդարտ կերպով նման կապի ստեղծում ։ Այնուամենայնիվ, JDBC- ի ներքին վարորդի դեպքում JDBC- ի հաճախորդը իրականում աշխատում է տվյալների բազայի հասանելի մասով և այդպիսով կարող է մուտք գործել ուղղակիորեն, այլ ոչ թե ցանցային արձանագրությունների միջոցով:

**1.3 Խնդրի դրվածք**

Վեբ ծրագրերի ճնշող մեծամասնությունը ի վերջո բախվում է որոշ կատարողական խնդիրների հետ՝ կապված տվյալ ռելացիոն տվյալների բազայի նախնական կառավարման համակարգի լուծումներից։ Հաճախ անհրաժեշտ է լինում կատարել տվյալների միգրացիա մի համակարգից մյուսը: Խնդիր է դրվել մշակել ինքնասպասարկող վեբ հավելված, որի նպատակն է ապահովել տվյալների բազաների միգրացիաներ և լուծել այս հիմնական խնդիրները ՝

* տվյալների շտեմարանների կառավարում,
* տվյալների բազաների սխեմաների տեղափոխում մի RDBMS համակարգից մյուսը։
* տվյալների միգրացիան հնացած համակարգերի համար։

Ծրագիրը պարունակելու է օգտագործողի սեփական ներքին կառավարում , թույլ կտա օգտագործողներին միանալ տվյալների բազայի համակարգերին, տեղափոխել տվյալները մեկից մյուսը՝ օգտագործելով batch SQL մշակում։ Նախնական օգտագործվող RDBMS համակարգերը կլինեն MySQL-ը, PostgreSQL-ը և Oracle-ը։

## Գլուխ 2 Նախագծի աշխատանքի նկարագրություն

## 2․1 Սերվերային աշխատանքի կազմակերպում

* WEB սերվերներ

Քոմփյութերային ցանցերի լայնորեն օգտագործվող կազմակերպական ձևերից մեկը կլիենտ-սերվերն է (Client-Server), ըստ որի ցանցի քոմփյութերները բաժանվում են ինֆորմացիոն ռեսուրսներ պարունակող սերվերների (Server – սպասարկում բառից) և այդ ռեսուրսներից օգտվող քոմփյութեր-կլիենտների:

Սերվերների գլխավոր խնդիրը սեփական ռեսուրսներին ուղղված հայտերի սպասարկումն է: Սերվերները մշտապես գտնվում են քոմփյութեր-կլիենտներից ստացվող հայտերի սպասման վիճակում, հայտերը ստանալուն պես սերվերները իրագործում են դրանք և կլիենտներին ուղարկում են պահանջվող ինֆորմացիան: Որպես սերվեր օգտագործվում են բավականաչափ հզոր քոմփյութերները, որոնք, բացի կլիենտների կողմից օգտագործվող ինֆորմացիոն ռեսուրսներից, ունենում են նաև անհրաժեշտ գործառույթներն ապահովող ծրագրային միջոցներ:

Կլիենտ-սերվեր տեխնոլոգիան օգտագործվում է ցանցային տարբեր ծառայությունների, այդ թվում և Web-ի իրագործման համար: Web-ծառայությունը բաղկացած է Web սերվերներից և կապուղիներով դրանց միացված Web կլիենտներից: Web սերվերները կարող են ունենալ հիշողություններում գրանցված հավելվածներ՝ հնարավոր է՝ մի քանիսը մեկ սերվերում:

Ցանցի սերվերում գրանցված կայքերը ունենում են հասցեներ, որոնք բնորոշում են տվյալ կայքերի տեղաբաշխումը սերվերի հիշողության մեջ և սերվերինը՝ ցանցում:

Web ծառայությունը իրագործվում է ինչպես Internet գլոբալ ցանցում, այնպես էլ կազմակերպությունների կորպորատիվ ցանցերում: Internet ցանցի դեպքում հավելվածների հասցեները կոչվում են URL և բնորոշում են Internet տարածքի Web-սերվերը և այդ սերվերի հավելվածը:

# Հաղորդակցություն Web տարածքում

Internet-ի հիմքը կազմում է հաղորդակցական ինֆրակառուցվածքը, իսկ ավելի ճիշտ գլոբալ ցանցը՝ կազմված հաղորդակցական հանգույցներից և կապի հաղորդակցական կապուղիներից (հիմնականում ձևավորվող արբանյակային և օպտիկամանրաթելային կապուղիների հիման վրա): Հաղորդակցական յուրաքանչյուր հանգույցը միացվում է այլ հանգույցների, ինչն ապահովում է տարբեր ուղղություններով տվյալների հաղորդման հնարավորությունը: Յուրաքանչյուր քոմփյութեր-կլիենտ միացած է Web սերվերին, որը, իր հերթին միացած է Internet-ի հաղորդակցական հանգույցին` ձևավորելով արդյունքում Web հավելվածների և Web կլիենտների գլոբալ ցանցը:

Հաղորդակցական հանգույցը հզոր քոմփյութեր է, որը պարունակում է ինֆորմացիա իրեն միացրած Web սերվերների, հավելվածների, ինչպես նաև այն հերթական հանգույցի մասին, որին հարկավոր է դիմել որոնվող հավելվածի հաղորդակցական հանգույցին հասնելու համար:

Քոմփյութեր-կլիենտի օգնությամբ ձևավորված հայտը պարունակում է որոնվող Web սերվերի, սերվերի հավելվածի, հավելվածի էջերից մեկի, ինչպես նաև տվյալ քոմփյութեր-կլիենտի հասցեները: Կլիենտից հայտն ուղարկվում է սերվեր, որից հաղորդակցական հանգույց: Այնուհետև հայտը ցանցի կապուղիների միջոցով փոխանցվում է հաղորդակցական մեկ հանգույցից մյուսին՝ մինչև որոնվող սերվերի հանգույցին հասնելը: Հանգույցից հայտը ուղարկվում է սերվեր, ուր ենթարկվում է մշակման: Սերվերը գտնում է պահանջվող կայքն ու կայքի էջը և այն քոմփյութեր-կլիենտի հասցեի հետ միասին առաքում է հետ՝ հաղորդակցական հանգույց: Այս պահից սկսվում է էջի «ճամփորդությունը» ցանցում կլիենտի ուղղությամբ: Էջը, ցանցի կապուղիներով՝ մեկ հաղորդակցական հանգույցից մյուսին փոխանցվելով, վերադառնում է հետ սկզբնական սերվեր, բեռնավորվում է սկզբնական քոմփյութեր-կլիենտի մեջ և արտապատկերվում է էկրանին:

Նշենք, որ նկարագրված հաղորդումների մեծ մասը կատարվում է արբանյակային և օպտիկամանրաթելային կապուղիների միջոցով, որոնք ապահովում են տվյալների փոխանցման մեծ արագություն և բավականաչափ զարգացած հաղորդակցական միջավայրում օգտագործողն աշխատում է իրական ժամանակի ռեժիմով՝ ստանալով ինֆորմացիա առանց ուշացման:

# Ծրագրային սերվեր

Ծրագրային սերվերն այն ծրագրաշարի շրջանակն է, որն ապահովում է երկու հնարավորությունները `վեբ ծրագրեր եւ սերվերի միջավայր ստեղծելու համար:

Application Server Frameworks- ը պարունակում է ծառայությունների ծածկույթի ամբողջական մոդել: Ծրագրային սերվերը հանդես է գալիս որպես ծրագրակազմի մշակող հասանելի բաղադրիչների շարք, պլատֆորմի համար սահմանված ստանդարտ API-ի միջոցով: Վեբ հավելվածների համար այս բաղադրիչները սովորաբար կատարվում են նույն վազող միջավայրում, քանի որ դրանց վեբ սերվերը (ները) եւ նրանց հիմնական աշխատանքը աջակցում է դինամիկ էջերի կառուցմանը: Այնուամենայնիվ, շատ դիմումատուներ թիրախ են ավելի շատ, քան պարզապես վեբ էջի սերունդը: Նրանք իրականացնում են ծառայություններ, ինչպիսիք են կլաստերացումը, չկատարելը եւ բեռը հավասարակշռելը, այնպես որ մշակողները կարող են կենտրոնանալ գործարար տրամաբանության իրականացման վրա:

Java- ի կիրառման սերվերների դեպքում, սերվերը իրեն նմանեցնում է ընդլայնված վիրտուալ մեքենայի վարող ծրագրեր իրականացնելու համար, թափանցիկ կերպով տվյալների վրա տվյալների բազայի կապակցումը, եւ, հաճախ, մյուս կողմից Web client- ի հետ կապեր հաստատումը:

Դիմորդի սերվերները համակարգչային ծրագրեր են, որոնց վրա վեբ ծրագրերը կամ աշխատասեղանի հավելվածները գործարկվում են: Դիմումի սերվերները բաղկացած են վեբ սերվերի միացումներից, համակարգչային ծրագրավորման լեզուներից, ռեսուրսների գրադարաններից, տվյալների բազայի միակցիչներից եւ կառավարման կոդից, որոնք անհրաժեշտ են տեղակայել, կարգավորել, կառավարել եւ միացնել այս բաղադրիչները վեբ հոսթին: Ծրագրային սերվերը վարում է վեբ սերվերի հետևում եւ SQL- ի նախադրյալների տվյալների բազայի առաջ (օրինակ, PostgreSQL, MySQL կամ Oracle): Վեբ ծրագրերը համակարգչային կոդ են, որոնք աշխատում են վերոնշյալ ծրագրային սերվերների վրա:

Java EE- ի կիրառման սերվերները ապահովում են համակարգային ծառայություններ` լավ սահմանված, բաց, արդյունաբերական ստանդարտում: Ծրագրի մշակողները զարգացնում են ծրագրեր, համաձայն Java EE- ի հատկորոշման, եւ ոչ թե կիրառման սերվերի համաձայն: Java EE ստանդարտի համաձայն մշակված Java EE ծրագիրը կարող է տեղակայվել ցանկացած Java EE դիմումի սերվերի մեջ:

* Apache Tomcat

Apache Tomcat- ը, որը հաճախ կոչվում է Tomcat Server, հանդիսանում է Apache Software Foundation- ի (ASF) կողմից մշակված բաց կոդով Java Servlet Container- ը: Tomcat- ը իրականացնում է մի քանի Java EE- ի առանձնահատկություններ, Java Servlet, JavaServer Pages (JSP), Java EL եւ WebSocket, եւ ապահովում է «մաքուր Java» HTTP վեբ սերվերի միջավայր, որտեղ Java կոդը կարող է գործարկել:

Tomcat- ը մշակվել եւ պահպանվել է Apache Software 2.0 լիցենզիայի տակ թողարկված Apache Software Foundation- ի հովանու ներքո մշակողների բաց հասարակության կողմից եւ բաց կոդով ծրագրային ապահովում է:

Tomcat 4.x- ը թողարկվեց Catalina- ի (servlet կոնտեյներ), Coyote- ի (HTTP միակցիչ) եւ Jasper- ի (JSP շարժիչ) միջոցով:

Catalina- ն Tomcat- ի servlet կոնտեյներ է: Catalina իրականացնում է Sun Microsystems- ի առանձնահատկությունները servlet եւ JavaServer Pages (JSP): Tomcat- ում, Realm տարրը ներկայացնում է օգտագործողների անունները, գաղտնաբառերը եւ դերերը (նման Unix խմբերի): Տարածքի տարբեր տարբերակները թույլ են տալիս Catalina- ին ինտեգրվել միջավայրում, երբ այդպիսի նույնականացման տեղեկատվությունը արդեն ստեղծվում եւ պահպանվում է, եւ այնուհետեւ օգտագործում է այդ տեղեկատվությունը, օգտագործելով Container Managed Security- ը, ինչպես նկարագրված է Servlet Specification- ում:

Coyote- ն Tomcat- ի միակցիչ բաղադրիչ է, որը պաշտպանում է HTTP 1.1 արձանագրությունը որպես վեբ սերվեր: Սա թույլ է տալիս Catalina- ին, անվանապես Java Servlet- ի կամ JSP- ի կոնտեյներով, ինչպես նաեւ հանդես գալ որպես պարզ վեբ սերվիս, որը ծառայում է տեղական ֆայլերը որպես HTTP փաստաթղթեր: Coyote- ը լսում է սերվերի մուտքային կապի համար հատուկ TCP -ի վրա եւ պահանջում է Tomcat Engine- ին խնդրանքը մշակել եւ ուղարկել պատասխանող հաճախորդին: Մեկ այլ Coyote- ի միակցիչ, Coyote JK- ը, նույնպես նույնն է լսում, այլ փոխում է իր պահանջները մեկ այլ վեբ սերվերի, օրինակ Apache- ի միջոցով, օգտագործելով JK արձանագրությունը:

Jasper- ն Tomcat- ի JSP- ի շարժիչն է: Jasper- ը վերլուծում է JSP ֆայլերը `դրանք Java կոդը, որպես servlets (որը կարող է վարվել Catalina - ի կողմից): Ժամանակի ընթացքում Jasper- ը հայտնաբերում է փոփոխություններ JSP ֆայլերի եւ վերակոմպիլացնում նրանց:

Tomcat- ը օգտագործում է Jasper 2, որն իրականացնում է Sun Microsystems- ի JSP 2.0 հստակեցումը: Jasper- ից մինչեւ Jasper 2, կարեւոր առանձնահատկություններ են ավելացվել՝ JSP- ի թղթապանակի գրադարանների միավորում - JSP ֆայլի յուրաքանչյուր տառատեսակի շղթա վարվում է տեքստային մշակողի կարգավիճակով, Tag աշխատող դասի օբյեկտները կարող են միացվել եւ վերագործվել ամբողջ JSP servlet- ում:

Tomcat 7-ի թողարկմամբ ավելացվել է երեք նոր բաղադրիչ`

Կլաստեր բաղադրիչն ավելացվել է մեծ ծրագրեր կառավարելու համար: Այն օգտագործվում է բեռի հավասարակշռման համար, որը կարելի է ձեռք բերել բազմաթիվ մեթոդների միջոցով: Կլաստերային աջակցությունը ներկայումս պահանջում է JDK տարբերակը 1.5 կամ ավելի բարձր:

Բարձր բարելավման հնարավորությունը ավելացվել է համակարգի վերազինման պլանավորման համար (օրինակ, նոր թողարկումներ, փոփոխության հարցումներ) առանց կենդանի միջավայրի վրա ազդելու: Դա կատարվում է ուղիղ երթեւեկության հարցումներ ուղարկելով ժամանակավոր սերվերի վրա, իսկ հիմնական սերվերը վերազինվում է հիմնական միջավայրում: Դա շատ օգտակար է բարձր արտադրողականության համար՝ վեբ դիմումների վերաբերյալ օգտվողների խնդրանքների հետ կապված հարցերում :

Վեբ կիրառումը նաեւ ավելացրել է օգտագործողը, ինչպես նաեւ համակարգային վեբ հավելվածների ընդլայնումը `աջակցելու համար միջավայրի տարբեր տեղակայման համար: Այն նաեւ փորձում է կառավարել նիստերը, ինչպես նաեւ ցանցի դիմումները:

**2․2 Տվյալների բազայի կառավարման համակարգեր**

Ռելացիոն տվյալների բազայի կառավարման համակարգը (RDBMS) հիմնված է տվյալների փոխկապակցման մոդելի հիման վրա: Այսօր տարածված տվյալների բազաների մեծ մասը հիմնված է այս մոդելի վրա:

RDBMS- ը 1980-ական թվականներից ի վեր ֆինանսական հաշվետվությունների, արտադրության եւ լոգիստիկ տեղեկատվության, կադրային տվյալների եւ այլ ծրագրերի համար օգտագործվող տվյալների բազաների տեղեկատվության պահպանման ընդհանուր տարբերակն է: Հարաբերական տվյալների բազաները հաճախ փոխարինել են ժառանգական հիերարխիկ տվյալների բազաների եւ ցանցային տվյալների բազաների համար, քանի որ դրանք ավելի հեշտ էին իրականացնել եւ կառավարել: Այնուամենայնիվ, հարակից տվյալների բազաները ստացել են շարունակական, անհաջող մարտահրավերներ 1980-ական եւ 1990-ական թվականների ընթացքում օբյեկտների տվյալների կառավարման համակարգերի միջոցով (որոնք ներկայացվել են այսպես կոչված «օբյեկտային հարաբերական իմպաստային անհամապատասխանության» հարաբերական տվյալների բազաների եւ օբյեկտի վրա հիմնված կիրառական ծրագրերի միջեւ): ինչպես նաեւ XML- ի տվյալների բազայի կառավարման համակարգերը 1990-ականներին: [citation needed] Սակայն, տեխնոլոգիաների տարածման շնորհիվ, ինչպիսիք են համակարգչային կլաստերների հորիզոնական մասշտաբը, NoSQL- ի տվյալների բազաները վերջերս հայտնի դարձան որպես RDBMS տվյալների բազայի այլընտրանք:

**2․3 Օգտագործված տվյալների բազաների կառուցվածքը**

MySQL -ը [բաց կոդով](https://hy.wikipedia.org/w/index.php?title=%D4%B2%D5%A1%D6%81_%D5%AF%D5%B8%D5%A4&action=edit&redlink=1) [տվյալների բազաների կառավարման համակարգ](https://hy.wikipedia.org/wiki/%D5%8F%D5%BE%D5%B5%D5%A1%D5%AC%D5%B6%D5%A5%D6%80%D5%AB_%D5%A2%D5%A1%D5%A6%D5%A1%D5%B6%D5%A5%D6%80%D5%AB_%D5%AF%D5%A1%D5%BC%D5%A1%D5%BE%D5%A1%D6%80%D5%B4%D5%A1%D5%B6_%D5%B0%D5%A1%D5%B4%D5%A1%D5%AF%D5%A1%D6%80%D5%A3) է։ MySQL-ի մշակումն ու աջակցումն իրականացվում է Oracle կորպորացիայի կողմից։ Վերջինս MySQL-ի նկատմամաբ իրավունքները ստացել է [Sun Microsystems](https://hy.wikipedia.org/w/index.php?title=Sun_Microsystems&action=edit&redlink=1) ընկերության գնումից հետո, որն էլ, նախքան այդ, իր հերթին գնել էր [MySQL AB](https://hy.wikipedia.org/w/index.php?title=MySQL_AB&action=edit&redlink=1) ընկերությունը։ MySQL-ն տարածվում է [GNU GPL](https://hy.wikipedia.org/w/index.php?title=GNU_GPL&action=edit&redlink=1), ինչպես նաև սեփական կոմերցիոն արտոնագրի ներքո, չնայած մշակողներն նոր ֆունկցիաներ են ավելացնում արտոնագրված օգտագործողների պատվերներով։

MySQL-ը հիմնականում օգտագործվում է փոքր և միջին ծանրաբեռնվածության ծրագրերի համար։ Այն ներառված է մի շարք վեբ-ծրագրավորման համար նախատեսված ծրագրերի փաթեթների մեջ, ինչպիսիք են [WAMP](https://hy.wikipedia.org/wiki/WAMP)-ը, [AppServ](https://hy.wikipedia.org/w/index.php?title=AppServ&action=edit&redlink=1)-ը, [LAMP](https://hy.wikipedia.org/w/index.php?title=LAMP&action=edit&redlink=1)-ը, [XAMPP](https://hy.wikipedia.org/wiki/XAMPP)-ը և [Open Server](https://hy.wikipedia.org/w/index.php?title=Open_Server&action=edit&redlink=1)-ը։ MySQL բավականին ճկուն է։ Այն աջակցում է աղյուսակների այնպիսի տեսակներ, ինչպիսիք են [MyISAM](https://hy.wikipedia.org/w/index.php?title=MyISAM&action=edit&redlink=1)-ը կամ [InnoDB](https://hy.wikipedia.org/w/index.php?title=InnoDB&action=edit&redlink=1)-ն։ Բացի այդ, այն տեղադրվում է հատուկ EXAMPLE անվանմամբ աղյուսակի տեսակով, որն ցուցադրում է նոր աղյուսակի տիպեր ստեղծելու սկզբունքները։ GPL արտանգրի շնորհիվ, MySQL-ն անընդհատ հարստանում է նոր աղյուսակի տեսակներով։

PostgreSQL- ը, հաճախ Postgres- ը, օբյեկտների հետ շփման տվյալների բազայի կառավարման համակարգ է (ORDBMS), ընդգծելով ընդարձակման եւ ստանդարտների համապատասխանությունը: Այն կարող է կարգավորել աշխատանքային բեռնվածքը `սկսած փոքր միանվագ մեքենայական ծրագրերից մինչեւ ինտերնետի առջեւ ծառացած մեծ ծրագրեր (կամ տվյալների պահեստավորում), շատ զուգորդող օգտվողների հետ, MacOS Server- ում PostgreSQL- ը լռելյայն տվյալների բազա է և այն նաեւ հասանելի է Microsoft Windows եւ Linux- ի համար:

PostgreSQL- ն ներառում է ներկառուցված երկուական վերարտադրություն, որը հիմնված է փոխարինելու փոփոխությունները (գրելու համար առաջացած տեղեկամատյանները (WAL)), չհամագործակցող հանգույցների համար, երբեմն սինխրոն կերպով, կարդալու միայն հարցումները այս վերարտադրվող հանգույցների դեմ: Սա թույլ է տալիս կարդալ երթեւեկությունը բազմակի հանգույցների միջեւ արդյունավետ կերպով: Նախկինում replication- ի ծրագրային ապահովումը, որը թույլ էր տալիս ընթերցել նման սանդղակը, սովորաբար հիմնված էր վարպետի վերարտադրման տատանումների վրա, լրացուցիչ ծանրաբեռնվածության վրա:

PostgreSQL- ը ներառում է ներկառուցված համաժամանակյա վերարտադրումը , որը ապահովում է, որ յուրաքանչյուր գրելու գործարքի համար վարպետը սպասում է մինչեւ առնվազն մեկ կրկնօրինակ հանգույց գրել տվյալները գործարքի գրանցամատյանում: Ի տարբերություն այլ տվյալների բազայի համակարգերի, գործարքի երկարակեցությունը (անկախ այն, որ այն համաժամանակյա է կամ համաժամանակյա) կարող է սահմանվել մեկ բազայի, մեկ օգտագործողի, մեկ նստաշրջանի կամ նույնիսկ մեկ գործարքի համար: Սա կարող է օգտակար լինել աշխատանքային բեռների համար, որոնք չեն պահանջում նման երաշխիքներ, եւ չեն կարող պահանջվել բոլոր տվյալների համար, քանի որ դա որոշակի բացասական ազդեցություն կունենա կատարողականի հավաստիացման պահանջի պատճառով: Կա սինխրոն եւ ասինխրխային սպասողական սերվերների խառնուրդը: Սինխրոն սպասման սերվերների ցանկը կարող է սահմանվել այն կազմաձեւում, որը որոշում է, թե որ սերվերներն են համաժամանակյա վերարտադրման թեկնածուները: Առաջինը, որը ներկայումս միացված է եւ ակտիվորեն հոսք է, այն օգտագործվելու է որպես ներկա համաժամանակյա սերվեր: Երբ դա չկատարվի, այն ընկնում է հաջորդին:

Oracle տվյալների բազան (սովորաբար կոչվում է Oracle RDBMS կամ պարզապես որպես Oracle) հանդիսանում է Oracle կորպորացիայի կողմից արտադրվող եւ շուկա շուկայի բազմատեսակ տվյալների բազա :

Այն տվյալների բազան, որը սովորաբար օգտագործվում է առցանց գործարքների վերամշակման (OLTP), տվյալների պահեստավորման (DW) եւ խառը (OLTP & DW) տվյալների բազայի աշխատանքների համար: Վերջին սերնդի Oracle Database 18c- ը հասանելի է on-prem, on-Cloud կամ հիբրիդ-Cloud միջավայրում: 18C- ն կարող է նաեւ տեղադրվել Oracle Engineered Systems- ի վրա (օրինակ, Exadata), Oracle- ի (հանրային) Cloud- ի կամ (մասնավոր) Cloud- ի վրա Հաճախորդի կողմից: [5] Սան-Ֆրանցիսկոյում Openworld 2017-ում, Կոմիտեի եւ CTO- ի գործադիր նախագահ Լարի Էլլիսոնը հայտարարեց հերթական տվյալների շտեմարանի ստեղծման մասին, Oracle ինքնավար տվյալների բազան:

**2․4 Իրական ժամանակի ռեժիմում հաղորդագրություններ փոխանակելու համակարգ**

WebSocket, լրիվ դուպլեքս միացման արձանագրություն (կարող է փոխանցել և ստանալ միաժամանակ) TCP-միացման հիման վրա՝ նախատեսված բրաուզերի և վեբ-սերվերի միջև իրական ժամանակի ռեժիմում հաղորդագրություններ փոխանակեու համար։ Ներկայումս [W3C](https://hy.wikipedia.org/wiki/W3C)-ում իրականացվում է API Web Sockets ստանդարտացում։ Այդ տարբերակի սևագրային տարբերակը հաստատված է [IETF](https://hy.wikipedia.org/w/index.php?title=IETF&action=edit&redlink=1)-ի կողմից։

WebSocket-ը մշակված է վեբ-բրաուզերներում և վեբ-սերվերներում օգտագործման համար, բայց այն կարող է նաև օգտագործվել ցանկացած սերվերային կամ կլիենտային հավելվածի համար։ WebSocket-ը անկախ արձանագրություն է՝ հիմնված TCP արձանագրության վրա։ Այն հնարավոր է դարձնում առավել սերտ համագործակցությունը բրաուզերի և վեբ-կայքիի միջև՝ նպաստելով ինտերակտիվ բովանդակության տարածմանը և իրական ժամանակում աշխատող հավելվածների ստեղծմանը։

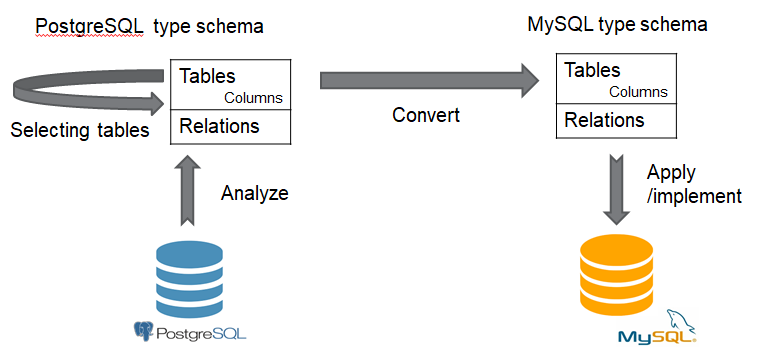
WebSocket- ը HTTP- ից այլ արձանագրություն է: Երկու արձանագրությունները տեղակայված են OSI մոդելի 7-րդ շերտում եւ որպես այդպիսին կախված են TCP- ի 4-րդ շերտում: Չնայած դրանք տարբեր են, RFC 6455- ը նշում է, որ WebSocket- ը <<նախագծված է աշխատել HTTP 80 եւ 443-ի վրա, ինչպես նաեւ աջակցել HTTP վստահված անձինք եւ միջնորդները >>, դրանով իսկ այն համատեղելի է HTTP արձանագրության հետ:

WebSocket պրոտոկոլը թույլ է տալիս փոխկապակցել վեբ բրաուզերի (կամ այլ հաճախորդի հավելվածի) եւ ավելի ցածր գներով աշխատող վեբ սերվերի միջեւ, դյուրացնելով իրական ժամանակի տվյալների փոխանցումը եւ սերվերին: Սա հնարավոր է դարձնում ստանդարտացված եղանակը `սերվերի բովանդակությունը պատվիրատուին ուղարկելու, առանց հաճախորդի կողմից առաջին անգամ խնդրանքի, եւ կապի բացումը պահելու թույլտվություն ստանալու համար, Այսպիսով, հաճախորդը եւ սերվերի միջեւ կարող է տեղի ունենալ երկկողմանի զրույց: Կապերը կատարվում են TCP պորտի 80-ի (կամ 443-ը, TLS-encrypted connections- ի դեպքում), որն օգուտ է այն միջավայրերի համար, որոնք արգելափակում են ոչ վեբ ինտերնետ կապեր `օգտագործելով firewall- ը: Նման երկկողմանի զննարկիչ-սերվերի հաղորդակցությունը ստացել է ոչ ստանդարտացված եղանակներով, օգտագործելով stopgap տեխնոլոգիաները, ինչպիսիք են Comet- ը:

Բրաուզերների մեծ մասը աջակցում է արձանագրությունը, ներառյալ Google Chrome, Microsoft Edge, Internet Explorer, Firefox, Safari եւ Opera:

**2․5 Ալգորիթմի նկարագրություն**

Օգտագործողի կողմից տվյալների բազաների մասին ներմուծված տվյալներից որոշվում են սկզբնական աղբյուր հանդիսացող և նպատակային տվյալների բազաների տիպերը։ Սկզբնական տվյալների բազայի հիման վրա ծրագրային մակարդակում ստեղծվում է տիպին համապատասխան սխեմա, որն իր մեջ ներառում է այդ տվյալների բազայում գտնվող բոլոր աղյուսակները, դրանց մասին տեղեկությունը, բոլոր սյուների տիպերը, դրանց վրա դրված սահմանապակումները եվ աղյուսակների միջև կապերը։ Օգտագործողին հնարավորություն է տրվում ընտրել այն աղյուսակները, որոնք պետք է միգրացիայի ենթարկվեն, կատարվում է սխեմայի փոխակերպում նպատակային RDBMS տիպի սխեմայի։ Դրա հիման վրա նպատակային RDBMS-ի կանոններին համապատասխան գեներացվում են աղյուսակների ստեղծման, primary և foreign key-ների տեղադրման և գրանցումների ավելացման SQL հարցումները։ Ապա այդ հարցումները կատարվում են նպատակային տվյալների բազայի կառավարման համակարգի համար։



## 

## Գլուխ 3 Ծրագրային համակարգի նկարագրություն

**3․1 Կառուցվածքային մասի նկարագրություն**

### 

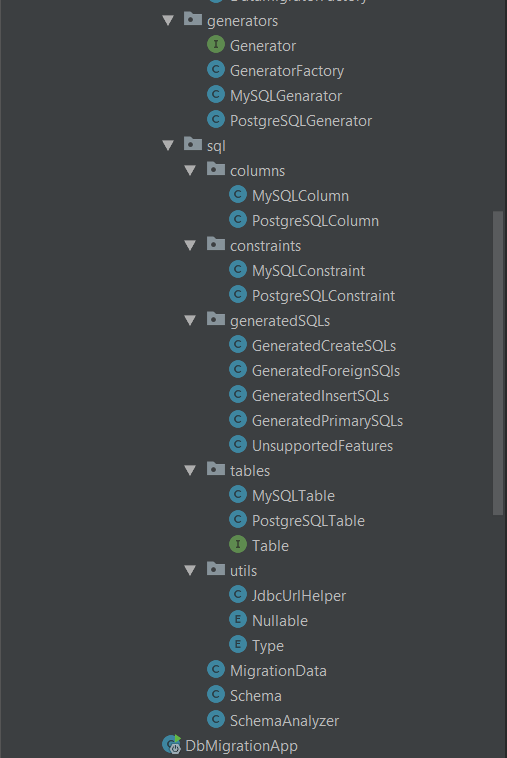
Տվյալների բազայի կառուցվածքի վերլուծության դասեր

Կոնֆիգուրացիոն դասեր

Կոնտրոլերներ

Տվյալների բազաի սխեմաների փոխակերպման դասեր

Գրանցումների տեղափոխման դասեր



SQL հարցումների գեներացման դասեր

Տարբեր տիպերի սյունակները ներկայացնող դասեր

Տարբեր տիպերի սահմանափակումները ներկայացնող դասեր

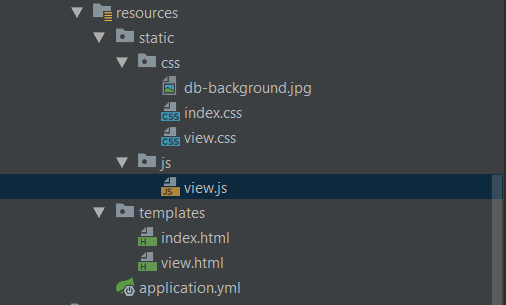
Տարբեր SQL հարցումների պահպանման դասեր

Տարբեր տիպերի աղյուսակները ներկայացնող դասեր

Տվյալների տիպերը պահող , լրացուցիչ դասեր

Հիմնական կառուցվածքային դասեր

Ծրագրի գործարկման դաս



Կայքին ձևավորում տվող ֆայլեր

[HTML](https://hy.wikipedia.org/wiki/HTML) էջերին դինամիկություն հաղորդող ֆայլ

[WEB-էջերի](https://hy.wikipedia.org/wiki/%D5%8E%D5%A5%D5%A2-%D5%A7%D5%BB) նշագրման ֆայլեր

Կառուցվածքային ֆայլ

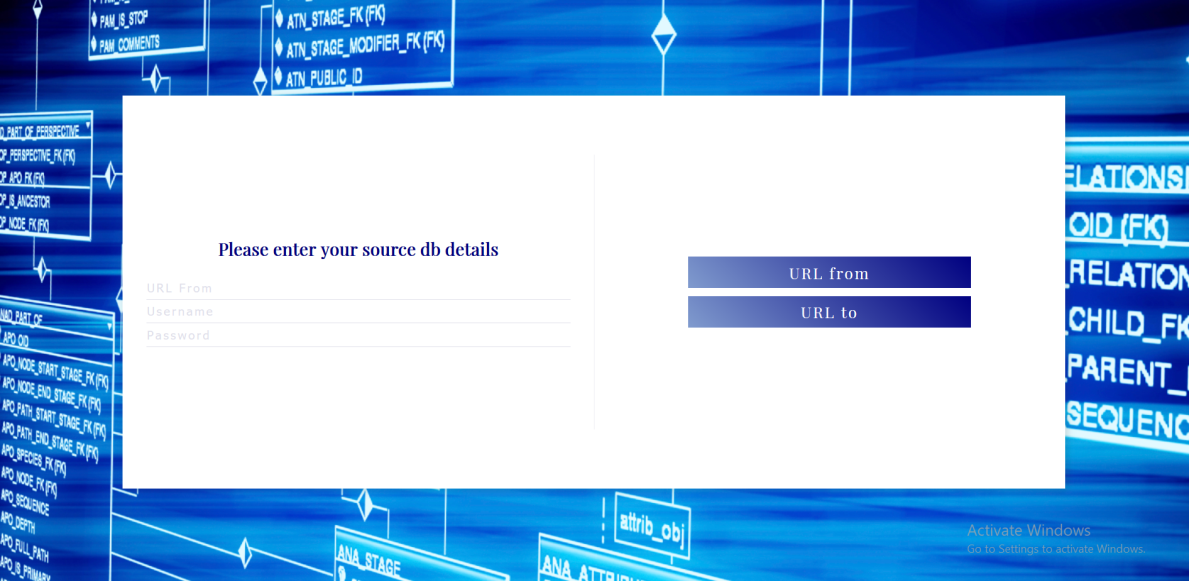
**3․2 Կատարման ընթացքի նկարագրություն**

Այսպիսով, անհրաժեշտ է ստեղծել վեբ հավելված, որի նպատակն է ապահովել տվյալների բազաների միգրացիաներ և լուծել այս հիմնական խնդիրները ՝

* տվյալների շտեմարանների կառավարում,
* տվյալների միգրացիան հնացած համակարգերի համար։

Ծրագիրը պարունակելու է օգտագործողի սեփական ներքին կառավարում, թույլ կտա օգտագործողներին միանալ տվյալների բազայի համակարգերին, տեղափոխել տվյալները մեկից մյուսը՝ օգտագործելով batch SQL մշակում։ Նախնական օգտագործվող RDBMS համակարգերը կլինեն MySQL-ը, PostgreSQL-ը և Oracle-ը։

Ծրագրի գործարկվելուն պես էկրանին հայտնվում է պատուհան, որտեղ անհրաժեշտ է մուտքագրել այն տվյալների բազաների տվյալները, որոնց միջև պետք է կատարել միգրացիան։

Յուրաքանչյուր տվյալների բազայի համար անհրաժեշտ է մուտքագրել համապատասխան JDBC url-ը, տվյալների բազային հասանելիություն ստանալու համար Username-ը և գաղտնաբառը։ Այս տվյալներով կապ է հաստատվում տվյալների բազաների հետ։

Առաջին հերթին կապը հաստատվում է այն տվյալների բազայի հետ, որտեղից պետք ե տեղափոխվեն տվյալները։ DDLAnalyzerFactory դասի getAnalyzer(urlFrom, usernameFrom, passwordFrom) ֆունկցիայով որոշվում է տվյալների բազայի տիպը և ընտրվում է համապատասխան Analyzer-ը (MySQLDDLAnalyzer, PostgreSQLDDLAnalyzer, OracleDDLAnalizer)։ SQL հարցումներով վերցվում է այդ տվյալների բազայի սխեման information\_schem-ից , կազմվում է Schema օբյեկտ, որը պարունակում է բոլոր աղյուսակները (համախատասխան տիպի Table օբյեկտներ), աղյուսակների միջև կապերը(համախատասխան տիպի Constraint օբյեկտներ), աղյեւսակների սյուները(համախատասխան տիպի Column օբյեկտներ), տիպերը, դրանց համապատասխան բնութագրող հատկորոշիչները։ Սխեման արտապատկերվում է էկրանին և օգտագործողը կարող է ընտրել այն աղյուսակները, որոնք ցանկանում է տեղափոխել։

Մուտքագրումից հետո ծրագիրը ընտրված աղյուսակները փոխակերպում է նպատակային տվյալների բազային համապատասխան աղյուսակների ConverterFactory դասի getConverter(sourceUrl, destinationUrl) ֆունկցիայով ընտրելով համապատասխան փոխակերպիչը։ Փոխակերպման ընթացքում ընտրվում են տվյալների տիպերը, սահմանվում են դրանց առավելագույն երկարությունը , լռելյայն արժեքները, ավտոմատ գեներացման հատկությունները։ Այնուհետև ուսումնասիրելով փոխակերպված սխեման, աղյուսակների կառուցվածքը սյուների տիպերը, դրանց բնութագրիչները, աղյուսակների միջև կապերը գեներացվում է նպատակային տվյալների բազայի տիպին համապատասխան SQL հարցումները։ Նաև վերցվում է սկզբնական տվյալների բազայի ընտրված աղյուսակների պարունակությունը և կազմվում դրանց INSERT հարցումները ըստ նպատակային տվյալների բազայի syntax-ի։ Օգտագործողը տեղեկացվում է , թե որ կապերը չեն կարող իրականացվել կախված չընտրված աղյուսակներից։ Նաև տեղեկացվում է այն գործառույթների մասին, որոնք ընդգրկված չեն ծրագրային ապահովման մեջ։

Օգտագործողին տեղեկացումից հետո տրվում է համաձայնվելու հայտ , որի ընդունման դեպքում իրականացվում է միգրացիան։ Կապ է հաստատվում նպատակային տվյալների բազայի հետ ։ Batch SQL Processing-ի միջոցով խումբ-խումբ կատարվում են նախ սխեմայի կառուցման հարցումները, ապա սյուների սահմանափակումները հաստատող հարցումները, աղյուսակների միջև կապեր հաստատող հառցումներ և վերջապես տվյալների աղյուսակների մեջ գրանցման հարցումները։ Այսպիսով միգրացիան համարվում է իրականացված և օգտագործողը վերադառնում է տվյալների բազաների հղումերի և գաղտնաբառերի մուտքագրման էջ, որտեղ կա հաղորդագրություն հաջող միգրացիայի կատարման մասին։