ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
«АЛМАТЫ ЭНЕРГЕТИКА ЖӘНЕ БАЙЛАНЫС УНИВЕРСИТЕТІ»
коммерциялық емес акционерлік қоғамы
IT-инжиниринг кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
Кафедра меңгерушісі
PhD, доцент
_____ Т.С. Картбаев
«____» _____ 2019 ж.

МАГИСТРЛІК ДИССЕРТАЦИЯ

Тақырыбы: «Simple binary encoding» хаттамасын қолданып көлемі үлкен деректерді тасымалдау мен өңдеу процестерін зерттеу
Мамандығы: 6М070300 – «Ақпараттық жүйелер»
Орындаған: Болысбекова Алина Бақытқызы Тобы: МИСн-17-1 _____
Ғылыми жетекші: к.ф.-м.н., доцент Умурзаков Б.Н. _____

Кеңесшілер:

Бағдарламалық қамту кеңесшісі: аға оқытушы Айткулов Ж. С.
_____ «13» 05 2019 ж.
(қолы)

Норма бақылау кеңесшісі: аға оқытушы Алимсеитова Ж.К.
_____ «22» 05 2019 ж.
(қолы)

Сын-пікір беруші: к.ф.-м.н., доцент, Әл-Фараби атындағы ҚазҰУ Ақпараттық технологиялар факультетінің информатика кафедрасының меңгерушісі Урмашев Б. А.
_____ «____» _____2019ж.
(қолы)

Алматы 2019

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

«АЛМАТЫ ЭНЕРГЕТИКА ЖӘНЕ БАЙЛАНЫС УНИВЕРСИТЕТІ»
коммерциялық емес акционерлік қоғамы

Басқару жүйелері және ақпараттық технологиялар институты

IT-инжиниринг кафедрасы

Мамандығы 6М070300– «Ақпараттық жүйелер»

Магистрлік диссерацияны орындауға берілген
ТАПСЫРМА

Білім алушы: Болысбекова Алина Бақытқызы

Диссертация тақырыбы: «Simple binary encoding» хаттамасын қолданып көлемі үлкен деректерді тасымалдау мен өңдеу процестерін зерттеу

2017 жылғы «23» 10 № 161 университет бұйрығымен бекітілген.

Аяқталған диссертацияны тапсыру мерзімі: «20» 05 2019 ж.

Магистрлік диссертацияның бастапқы мәліметтері (зерттеу (жоба) нәтижелерінің талап етілген параметрлері мен объектінің бастапқы мәліметтері): Ұсынылып отырған магистрлік диссертацияда қарапайым бинарлық кодтау (SBE) хаттамасын қолдана отырып, көп қаржы ақпаратын тезірек аударуға арналған әдістер мен қосымшалар. SBE хаттамасын JSON хаттамасымен салыстыру ұсынылады, қарастырылып отырған хаттаманың артықшылығы - деректерді жылдам өңдеу және жіберу.

Магистрлік диссертацияда қарастырылған мәселелер тізімі немесе жобаның қысқаша мазмұны:

а) Сериализация форматтары немесе деректерді сериализацияландыру пішімдерін салыстыру;

б) Simple Binary Encoding хаттамасы;

в) SBE архитектурасы және хабар сұлбаларын іздеу;

г) Ғылыми жұмыстың дамуы және деректерді жинау;

д) Бағдарламаны құру.

Графикалық материалдар тізімі (міндетті сызбалар дәл көрсетілуі тиіс): 3.5.5, 3.5.6, 3.5.6 (а), 3.6.1 және 3.6.2-суреттер; диаграмма 3.7.

Ұсынылатын негізгі әдебиеттер:

1. М. Томсон (2014, 6 қаңтар) Real-logic/simple-binary-encoding [Онлайн]. https://github.com/real-logic/simple-binary-encoding/wiki/FIX-SBE-XML-Primer.

2. М. Томсон (2014, 5 мамыр) Mechanical Sympathy [Онлайн]. http://mechanical-sympathy.blogspot.com/2014/05/simple-binary-encoding.html.

3. BestWebSoft (2013, 10 қаңтар) Forex Historical Data [Онлайн]. http://www.histdata.com/download-free-forex-historical-data/?/metatrader/1-minute-bar-quotes/EURUSDo.

4. JavaPortal.ru (2013, 12 мамыр) Что такое JDBC ? [Онлайн]. http://www.javaportal.ru/java/tutorial/tutorialJDBC/intro.html.

Магистрлік диссертация бөлімдеріне қатысты белгіленген кеңес берушілер

Бөлімдер	Кеңесшілер	Мерзімі	Қолы
Норма бақылау	Алимсеитова Ж.К.	22.05.2019	
Бағдарламалық қамту	Айткулов Ж. С.	13.05.2019	

Магистрлік диссертацияны дайындау
КЕСТЕСІ

Бөлімдер атауы, қарастырылатын мәселелер тізімі	Ғылыми жетекшіге ұсыну мерзімдері	Ескерту
Талдау бөлімі	04.09.2017-31.12.2018	
Жобалау бөлімі	01.01.2018-31.04.2018	
Жүзеге асыру және тестілеу бөлімі	01.05.2018-15.05.2019	

Тапсырманың берілген күні «4» 09 2017ж.

Кафедра меңгерушісі _____ Картбаев Т.С.

Диссертацияның ғылыми жетекшісі _____ Умурзаков Б.Н.

Тапсырманы орындауға алған
білім алушы _____ Болысбекова А.Б.

Аңдатпа

Зерттеу жұмысының тақырыптары: қарапайым бинарлық кодтау (SBE) хаттамасын қолдана отырып, көп қаржы ақпаратын тезірек аударуға арналған әдістер мен қосымшалар. Деректер сериализациясы мен десериализациясының бірнеше форматтары қарастырылады. Сондай-ақ, SBE көмегімен деректерді жіберу және өңдеудің бағдарламалық қамтамасыз етілуін әзірлеу және схеманы құру архитектурасы сипатталады. Қаржы биржасының бағамы және биткоиндардың криптовалютасының бағамының графикалық көріністері бағдарламалық қамтудың нәтижелері ретінде қарастырылады. SBE хаттамасын JSON хаттамасымен салыстыру ұсынылады, қарастырылып отырған хаттаманың артықшылығы - деректерді жылдам өңдеу және жіберу.

Аннотация

Предметы исследовательской деятельности: методы и приложения для ускоренной передачи больших объемов финансовой информации с применением протокола simple binary encoding (SBE). Рассматриваются несколько форматов сериализаций и десериализаций данных. Также описывается разработка программного обеспечения передачи и обработки данных с помощью SBE и архитектура построения схемы. Графические представления курса финансовой биржи и курса криптовалюты биткоинов рассматриваются как результат программного обеспечения. Приводится сравнение протокола SBE с протоколом JSON, преимуществами интересуемого протокола являются более быстрая обработка и передача данных.

Annotation

Subjects of research: methods and applications for accelerated transmission of large amounts of financial information using the Protocol simple binary encoding (SBE). There are several formats for serializing and deserializing data. It also describes the development of software for data transmission and processing using SBE and the architecture of the scheme. A graphical representation of the course of financial exchanges and of course the cryptocurrency bitcoin are viewed as the result of the software. A comparison of the SBE Protocol with the JSON Protocol is given, the advantages of the Protocol in question are faster data processing and transmission.

Мазмұны

Кіріспе.....6
1 Жіберу технологиялары мен әдістерін зерттеу, сондай-ақ көптеген деректерді өңдеу.....7
1.1 Үлкен деректерді зерттеу және талдау.....7
1.1.1 Үлкен деректер.....7
1.1.2 Үлкен деректер қалай сақталады және өңделеді.....9
1.2 Қаржы биржасында деректерді жіберу технологиялары мен әдістері.....11
1.2.1 Сериализация форматтары немесе деректерді сериализацияландыру пішімдерін салыстыру.....11
1.2.2 Бағдарламалау тілін қолдау.....12
1.3 SBE архитектурасы және хабар сұлбаларын іздеу.....19
1.3.1 Деректерді сериализациялау және десериализациялау.....19
1.3.2 FIX SBE қаржылық ақпарат алмасуы.....20
1.3.3 Қаржылық ақпаратты пайдаланушылар.....23
1.3.4 Криптовалюта деректері және оның маңыздылығы.....25
1.3.5 Биржалық графиктердің жұмыс жасау принциптері.....29
1.4 Зерттеудің өзектілігі мен жаңалығы.....32
2 Simple Binary Encoding хаттамасы.....33
2.1 CPU & Memory революциясының тарихы.....33
2.1.1 Кэшті алдын ала таңдау.....34
2.1.2 Ағынды қатынау.....35
2.1.3 Қауіпсіз API-интерфейсі.....35
2.2 Әдістерді және қолданыстағы деректерді жіберу хаттамаларын сараптау және демонстрациялық талдау.....36
2.2.1 Жобалау реті.....38
2.3 Ғылыми жұмыстың дамуы және деректерді жинау.....40
2.4 Simple binary encoding (SBE) деректерді жіберу хаттамасының соңғы қолдануларына талдау.....40
3 Бағдарламаны құру.....44
3.1 Жүйенің архитектурасы.....44
3.2 Сервер жұмысы.....46
3.3 SBE деректер құрылымы.....47
3.4 SBE схемасының сипаттамасы.....48
3.5 Деректерді кодтау және декодтау.....50
3.6 Биржа мен ресми сайтқа шолу.....53
3.7 SBE және JSON салыстырмалы талдауы.....57
Қорытынды.....61
Пайдаланылған әдебиеттер тізімі.....62
А қосымшасы.....64
Б қосымшасы.....65
В қосымшасы.....66
Г қосымшасы.....67

Кіріспе

Осы диссертациялық жұмыстың басты ерекшелігі - SBE хаттамасын қолдану арқылы қаржылық ақпараттардың үлкен көлемін жылдамдатылған түрде жіберуге арналған қосымша жасау. Қазіргі уақытта инновациялық кезеңде ақпараттық технологиялар барлық қызмет салаларында маңызды рөл атқарады. Жіберілген және қабылданған мәліметтердің саны жылдам артуы байқалады және деректерді жіберу жылдамдығын арттыру қажеттілігі туындайды. Осыған сәйкес, зерттеудің практикалық маңыздылығын атап кетсек, бұл құрастырылған бағдарлама ақпарат алудың кідірулерінің төмендеуі мен деректердің үлкен көлемін жедел түрде жіберу мүмкіндігін қамтамасыз етеді. Біздің еліміздің көптеген аймақтарында ақпараттық технологиялардың таралуы өте жоғары. Қор биржасы осы салалардың бірі болып табылады. Өздеріңіз білетіндей, қор нарығы бүгінгі таңда жоғары технологиялық салаларының бірі болып табылады. Ал осы прогрестің негізінде жоғары жылдамдықтағы байланыс арналарын жоғары жүктемемен күресетін бірнеше сауда алаңдары, брокерлік жүйелер құрылады. Бұдан, үлкен көлемдегі ақпаратты жіберу үшін деректердің үлкен компрессиясын қамтамасыз ететін қосымшаларға деген сұраныс тек қаржылық жүйеде ғана емес, сонымен қатар маркетинг, бизнес, телекоммуникация, сауда, логистика және үкіметте өзектілігі мол. Статистикаға сүйенсек, сақталған және алынған деректердің көлемі жыл сайын өсіп келеді, сонымен бірге жылдам деректерді жіберу мәселесі өте маңызды және ол XXI ғасырда үлкен рөл атқарады. Жұмыстың ғылыми жаңалығы - үлкен көлемдегі қаржы деректерін жедел түрде кешіктіріп жіберуін азайту әдісін әзірлеу, бұл декодталған және десериализацияланған деректерді дерекқорда сақтауға және жапондық шамдар түрінде ұсынуға мүмкіндік береді. SBE хаттамасының арасындағы сапалық айырмашылықтарды ажырату үшін артықшылықтар мен кемшіліктерді көрсететін JSON хаттамасымен салыстырмалы талдау жүргізілді.

1 Жіберу технологиялары мен әдістерін зерттеу, сондай-ақ көптеген деректерді өңдеу

1.1 Үлкен деректерді зерттеу және талдау

1.1.1 Үлкен деректер

Деректер - бұл электр сигналдары түрінде сақталатын, берілетін және магниттік, оптикалық немесе механикалық жазба тасығыштарына жазылатын компьютер көмегімен жасалатын операциялар орындалатын сандар, белгілер немесе символдар [1].

Үлкен деректер - бұл адам немесе машина сияқты ақпарат көздерінен алынған нақты айқындалғаннан бастап әлсіз анықталған деректерге дейінгі әр түрлі көздерден алынған деректер жиынтығы [2].

Үлкен деректер - бұл құрылымдалған, жартылай құрылымдалған және құрылымсыз деректердің үлкен көлемін сипаттайтын дамушы термин, олар ақпарат үшін өндіріліп, машиналық оқыту жобаларында және басқа да алдыңғы қатарлы аналитикалық қосымшаларда қолданылуы мүмкін [3].

Үлкен деректер жиі «3V» астында сипатталады: «volume» - деректердің экстремалды көлемі, «variety» - деректердің көп түрлілігі және «velocity» - деректер өңделетін жылдамдығы. Бұл сипаттамаларды алғаш рет әлемдегі жетекші Gartner зерттеу және кеңес беру компаниясының талдаушысы Дуг Лейни 2001 жылы жарияланған есепте анықтады. Жақында үлкен деректердің қатарына «veracity» шынайылық, «value» - құндылық, және «variability» - өзгергіштік сияқты тағы бірнеше «V» сипаттамалары қосылды. Кем дегенде үлкен деректер деректердің қандай да бір нақты көлеміне теңестірілмейді, бұл термин уақыт өте келе басып алынған терабайттарды, петабайттарды және тіпті эксабайттарды сипаттау үшін жиі қолданылады [4].

Мұндай көлемді деректер бизнес-транзакциялар жүйесі, клиенттердің деректер қоры, медициналық жазбалар, интернет-трафик журналдары, мобильді қосымшалар, әлеуметтік желілер, ғылыми эксперименттердің жиналған нәтижелері, машинамен жасалған деректер және IoT (интернет заттар) ортасында нақты уақытта қолданылатын деректер датчиктері сияқты көптеген түрлі көздерден түсуі мүмкін. Деректер өңделмеген күйінде қалдырылуы немесе деректерді талдаудан бұрын оларды дайындау үшін деректерді зияткерлік талдау құралдарын немесе бағдарламалық жасақтаманы пайдалану арқылы алдын ала өңделуі мүмкін [4].

Үлкен деректер, сондай-ақ Деректер базалары мен SQL деректер қоймаларында құрылымдалған деректерді, Hadoop кластерлерінде немесе NoSQL жүйесінде сақталатын мәтіндік файлдар мен құжаттар файлдары сияқты құрылымдалмаған деректерді және веб-сервер журналдары немесе датчиктерден деректерді ағынмен жіберу сияқты жартылай құрылымдалған деректерді қамтиды. Сонымен қатар, үлкен деректер бір мезгілде бірнеше деректер көздерін қамтиды, олар басқаша біріктірілуі мүмкін емес.

Мысалы, үлкен деректерді талдау жобасы өткен сату туралы деректерді, қайтару туралы деректерді және осы өнім үшін сатып алушының

онлайн-талдау деректерін салыстыру арқылы өнімнің табысы мен болашақ сатуды өлшеуге тырысуы мүмкін.

Жеделділік үлкен деректер генерацияланатын және өңделуі және талдануы тиіс жылдамдыққа жатады. Көптеген жағдайларда үлкен деректер жиынтықтары көптеген дәстүрлі деректер қоймаларында күнделікті, апта сайынғы немесе ай сайынғы жаңартулармен салыстырғанда нақты немесе нақты уақыт режимінде жаңартылады. Үлкен деректерді талдау жөніндегі жобалар келіп түскен деректерді қабылдайды, салыстырады және талдайды, содан кейін бәрін қамтитын сұрау негізінде жауап немесе нәтиже береді [4].

Үлкен деректер қалай сақталады және өңделеді

Үлкен көлемді деректерді өңдеу қажеттілігі базалық есептеу инфрақұрылымына бірегей талаптар қояды. Үлкен көлемді және деректер түрлерін жылдам өңдеу үшін қажетті есептеу қуаты бір серверді немесе серверлер кластерін бұзуы мүмкін. Ұйымдар қажетті жылдамдыққа қол жеткізу үшін үлкен деректер бар міндеттер үшін барабар есептеу қуатын қолдануы тиіс. Бұл кластерлік сәулетте өңдеуді және бірлесіп жұмыс істей алатын жүздеген немесе мыңдаған серверлерді талап етуі мүмкін [4].

Мұндай жылдамдыққа қол жеткізу экономикалық тиімді тәсіл де проблема болып табылады. Кәсіпорындардың көптеген басшылары үлкен деректердің, әсіресе тәулік бойы жұмыс істемейтіндердің жұмыс жүктемелерін қолдау үшін серверлер мен қоймалардың кең инфрақұрылымына қаражат салуға тырысады. Жалпы қол жетімді бұлтты есептеулер нәтижесінде қазіргі уақытта үлкен деректер жүйесін орналастырудың негізгі құралы болып табылады. Жалпы бұлт провайдері деректер петабайларын сақтай алады және үлкен деректерді талдау жобасын орындау үшін серверлердің қажетті санын көбейте алады. Бизнес іс жүзінде қолданылған сақтау және есептеу уақыты үшін ғана төлейді және бұлттың даналарын қайтадан қажет болғанша өшіруге болады [5].

Қызмет көрсету деңгейін одан әрі арттыру үшін, жалпы қол жетімді бұлт қызметінің жеткізушілері Apache Hadoop жоғары таралған есептеу даналарын, Apache Spark өңдеу механизмін және тиісті үлкен деректер технологияларын қамтитын басқару қызметтері арқылы үлкен деректермен жұмыс істеуге мүмкіндік береді. Amazon Web Services (AWS) Amazon Elastic MapReduce (EMR) - көпшілік бұлтта жұмыс істейтін үлкен деректер қызметінің мысалдарының бірі; басқалары Microsoft Azure HDInsight және Google Cloud Dataproc қамтиды. Бұлт ортасында үлкен деректер таратылған Hadoop (HDFS) файлдық жүйесінде немесе Amazon Simple Storage Service (S3) сияқты арзан бұлт қоймасында сақталуы мүмкін. NoSQL деректер қоры - бұлтта тағы бір нұсқасы [6].

Жергілікті үлкен деректер жүйесін өрістетуді қалайтын ұйымдар үшін, әдетте, Hadoop және Spark қосымша Ашық бастапқы Apache технологиясы, оның ішінде ресурстарды келісудің тағы бір құралы (YARN), Hadoop-ге енгізілген ресурстар менеджері және тапсырмаларды жоспарлаушы; MapReduce бағдарламалау шеңбері; Kafka, қосымшалар мен деректерді

ағындық беру арасында хабар алмасу платформасы; HBase деректер қоры; Drill, HiveИмпала және осылар сияқты SQL-on-Hadoop сұраныстарының механизмдері мен тетіктері пайдаланушылар ашық бастапқы коды бар технологиялар нұсқаларын өздері орната алады немесе Cloudera, Hortonworks және MapR Technologies ұсынатын үлкен деректердің коммерциялық платформаларына жүгіне алады, олар да бұлтта қолдау көрсетеді. Алайда, Cloudera және Hortonworks 2018 жылдың қазан айында қосылуға келісті, бұл қол жетімді жергілікті платформалар санын екіге дейін қысқартуы мүмкін [6].

Үлкен деректерді шағын деректермен, тағы бір дамып келе жатқан терминмен салыстыруға болады, ол мәліметтерді сипаттау үшін жиі пайдаланылады, оның көлемі мен пішімін өзіне-өзі қызмет көрсету талдаушылары үшін оңай пайдалануға болады. Аксиома жиі айтылатын «үлкен деректер машиналарға арналған; шағын деректер адамдарға арналған» [6].

Қаржы саласында, әсіресе қаржы қызметтері саласында үлкен деректер, мысалы, көптеген қосымшаларда пайдаланылады:

Қызметкерлерді бақылау және Мониторинг жүргізу;

Кредит беру туралы шешім қабылдау үшін сыйлықақыларды анықтау үшін сақтандыру андеррайтерлері мен кредиттік мамандар пайдалана алатындай болжаушы модельдер;

Қаржы нарықтарының бағытын болжау алгоритмдерін әзірлеу;

Жылжымайтын мүлік сияқты өтімді емес активтерді бағалау.

Қаржы мекемелері үлкен деректерді кеңінен пайдаланады: киберқауіпсіздікті арттырудан бастап клиенттердің кетуін төмендетуге дейін, заманауи банктік қызметтерді жоғары дараландырылған ететін инновациялық және дербестендірілген ұсыныстар есебінен клиенттердің және көптеген басқалардың ниеттестігін дамыту [6].

1.1.2 Үлкен деректер қалай сақталады және өңделеді

Үлкен көлемді деректерді өңдеу қажеттілігі базалық есептеу инфрақұрылымына бірегей талаптар қояды. Үлкен көлемді және деректер түрлерін жылдам өңдеу үшін қажетті есептеу қуаты бір серверді немесе серверлер кластерін бұзуы мүмкін. Ұйымдар қажетті жылдамдыққа қол жеткізуде үлкен деректер бар тапсырмаларды шешу үшін барабар есептеу қуатын қолдануы тиіс. Бұл кластерлік архитетектурада өңдеуді және бірлесіп жұмыс істей алатын жүздеген немесе мыңдаған серверлерді талап етуі мүмкін [6].

Экономикалық тиімді тәсілмен мұндай жылдамдыққа қол жеткізу де проблема болып табылады. Кәсіпорындардың, әсіресе тәулік бойы жұмыс істемейтіндердің көптеген басшылары үлкен деректердің жұмыс жүктемелерін қолдау үшін серверлер мен қоймалардың кең инфрақұрылымына қаражат салуға тырысады. Жалпы қол жетімді бұлтты есептеулер нәтижесінде қазіргі уақытта үлкен деректер жүйесін орналастырудың негізгі құралы болып табылады. Жалпы бұлт провайдері

деректер петабайларын сақтай алады және үлкен деректерді талдау жобасын орындау үшін серверлердің қажетті санын көбейте алады. Бизнес іс жүзінде қолданылған сақтау және есептеу уақыты үшін ғана төлейді және бұлттың даналарын қайтадан қажет болғанша өшіруге болады [6].

Қызмет көрсету деңгейін одан әрі арттыру үшін, жалпы қол жетімді бұлт қызметінің жеткізушілері Apache Hadoop жоғары таралған есептеу даналарын, Apache Spark өңдеу механизмін және тиісті үлкен деректер технологияларын қамтитын басқару қызметтері арқылы үлкен деректермен жұмыс істеуге мүмкіндік береді.

Amazon Web Services (AWS) компаниясынан Amazon Elastic MapReduce (EMR) - көпшілік бұлтта жұмыс істейтін үлкен деректер қызметінің мысалдарының бірі; басқалары Microsoft Azure HDInsight және Google Cloud Dataproc қамтиды. Бұлт ортасында үлкен деректер таратылған Hadoop (HDFS) файлдық жүйесінде немесе Amazon Simple Storage Service (S3) сияқты арзан бұлт қоймасында сақталуы мүмкін.); NoSQL деректер қоры қолданбаларға арналған бұлтта жақсы нұсқалардың тағы бі түрі болып табылады [6].

Жергілікті үлкен деректер жүйесін өрістетуді қалайтын ұйымдар үшін, әдетте, Hadoop және Spark үшін қосымша болатын ашық бастапқы коды бар Apache технологиясы, оның ішінде ресурстарды келісудің тағы бір құралы (YARN), Hadoop-қа енгізілген ресурстар менеджері және тапсырмаларды жоспарлаушы; MapReduce бағдарламалау шеңбері; Kafka, қосымшалар мен деректерді ағындық жіберу арасында хабар алмасу платформасы; HBase деректер қоры; Drill, HiveИмпала және сол сияқты SQL-on-Hadoop сұраныстарының механизмдері мен тетіктері пайдаланушылар ашық бастапқы коды бар технологиялар нұсқаларын өздері орната алады немесе Cloudera, Hortonworks және MapR Technologies ұсынатын үлкен деректердің коммерциялық платформаларына жүгіне алады, олар да бұлтта қолдауын көрсетеді. Алайда, Cloudera және Hortonworks қазан айында қосылу туралы келісті 2018, бұл қол жетімді жергілікті платформалар санын екіге дейін қысқартуы мүмкін [6].

Үлкен деректерді шағын деректермен, тағы бір дамып келе жатқан терминмен салыстыруға болады, ол мәліметтерді сипаттау үшін жиі пайдаланылады, оның көлемі мен пішімін өзіне-өзі қызмет көрсету талдаушылары үшін оңай пайдалануға болады. Аксиома жиі айтылатын «үлкен деректер машиналарға арналған; шағын деректер адамдарға арналған» [6].

Қаржы саласында, әсіресе қаржы қызметтері саласында үлкен деректер, көптеген қосымшаларда пайдаланылады, мысалы:

– Қызметкерлердің мониторингі және бақылауы;

– Сақтандыру андеррайтерлері сыйақы анықтау үшін және кредит беру туралы шешім қабылдау үшін кредиттік мамандар пайдалана алатындай болжаушы модельдер;

– Қаржы нарықтарының бағытын болжау алгоритмдерін әзірлеу;

– Жылжымайтын мүлік сияқты өтімді емес активтерді бағалау.

Қаржы мекемелері үлкен деректерді кеңінен пайдаланады: киберқауіпсіздікті арттырудан бастап клиенттердің кетуін төмендетуге дейін, заманауи банктік қызметтерді жоғары персонализацияланған түрге келтіретін инновациялық және дербестендірілген ұсыныстар есебінен клиенттердің және тағы да басқалардың ниеттестігін дамыту [7].

1.2 Қаржы биржасында деректерді жіберу технологиялары мен әдістері

Деректер сериализациясы - бұл құрылымдалған деректерді пішімге түрлендіру процесі, ол деректерді бірге пайдалануға немесе олардың бастапқы құрылымын қалпына келтіруге мүмкіндік беретін пішімде сақтауға мүмкіндік береді [8, 10].

Бұл деректер құрылымын немесе нысанның күйін сақтауға болатын форматқа (мысалы, файл немесе жад буферінде немесе желілік қосылым арнасында беріледі) және кейін сол немесе басқа компьютерлік ортада қалпына келтірілуі мүмкін түрлендіру процесі.

Кейбір жағдайларда деректерді сериализациялаудың екінші мақсаты деректер мөлшерін азайту болып табылады, содан кейін дискілік кеңістікке немесе өткізу қабілетіне қойылатын талаптарды төмендетеді. 2019 жылы деректерді сериализациялау форматтарының 60-тан астам түрі бар. Ең танымал және кең таралған түрлері ол: JSON, XML, ASN.1, Protocol Buffers (protobuf), және басқа да көптеген [8].

Сериализация жеке іске асыру бөлшектерін әлеуетті аша отырып, абстрактілі деректер түрінің мөлдірлігін бұзады. Барлық деректерді сериализациялайтын тривиальды іске асыру инкапсуляцияны бұзуы мүмкін.

Бірлескен өнімдерді жасамауға бәсекелестерді көндіру үшін, проприетарлық бағдарламалық қамтамасыз ету баспагерлері өз бағдарламаларын сериализациялау форматтарының бөлшектерін жиі құпияда ұстайды. Кейбіреулер әдейі шатастырады немесе сериалданған деректерді шифрлайды. Дегенмен, үйлесімділік қосымшалар бір-бірінің сериализация пішімдерін түсінуді талап етеді. Сондықтан CORBA сияқты әдістерді қашықтан шақыру архитектурасы өзінің сериализация пішімдерін егжей-тегжейлі анықтайды.

Мұрағаттар мен кітапханалар сияқты көптеген мекемелер болашақта өздерінің резервтік мұрағаттарын, атап айтқанда, оларды біршама оқуға ыңғайлы сериализацияланған форматта сақтай отырып деректер қорының қоқыстарын тексеруге тырысады [8, 9].

1.2.1 Сериализация форматтары немесе деректерді сериализацияландыру пішімдерін салыстыру

1980 жылдардың басында Xerox Network Systems Courier технологиясы бірінші кең қабылданған стандартқа әсер етті. Sun Microsystems 1987 жылы «сыртқы деректерді ұсыну» (XDR) жариялады [8].

1990-шы жылдардың соңында стандартты сериализация хаттамаларына

балама беруге ұмтылу басталды: XML ыңғайлы мәтіндік кодтауды құру үшін пайдаланылды. Мұндай кодтау адам оқуы және түсінуі немесе бағдарламалау тіліне қарамастан басқа жүйелерге берілуі мүмкін тұрақты объектілер үшін пайдалы болуы мүмкін. Ол байт ағынының негізінде жинақы кодтауды жоғалтудан тұрады, бірақ осы сәтте сақтау мен жіберудің үлкен сыйымдылықтары есептеудің алғашқы күндеріне қарағанда файлдың мөлшерін аса маңызды етіп жасады.

Екілік XML бұл мәтіндік редакторлар оқымаған, бірақ әдеттегі XML-ге қарағанда ықшамдау болды. 2000 жылдары XML Ajax веб-қосымшаларында клиент пен сервер арасында құрылымдалған деректерді асинхронды тасымалдау үшін жиі қолданылды [8].

JSON - бұл веб-қосымшалардағы клиент сервер байланысы үшін кеңінен қолданылатын XML мәтіндік баламасы. JSON JavaScript синтаксисінде негізделген, бірақ басқа да бағдарламалау тілдерінде қолдау көрсетіледі [8].

YAML протоколы JSON сияқты сериализация үшін күшті, әадамға ыңғайлы» және әлеуетті жинақы функцияларды қамтиды. Бұл функциялар деректер типтерін таңбалау ұғымын, деректердің иерархиялық емес құрылымдарын қолдауды, деректерді шегініспен құрылымдау мүмкіндігін және деректерді скалярлық дәйексөздердің көптеген нысандарын қамтиды [8].

Тағы бір адамға түсінікті сериализация форматы ол - NeXTSTEP, GNUstep және macOS Cocoa-да қолданылатын қасиеттер тізімі [8].

Жерсеріктік деректер және климаттың, ауа райының немесе мұхиттың сандық үлгілерінің шығу деректері сияқты ғылыми деректер жиынтығының үлкен көлемі үшін, мысалы, HDF, netCDF және GRIB алғашқы нұсқасы сериализацияның нақты екілік стандарттары әзірленді [9].

1.2.2 Бағдарламалау тілін қолдау

Кейбір объектілі-бағытталған бағдарламалау тілдері объектілердің сериализациясын (немесе объектілерді мұрағаттауды) немесе синтаксистік элементтердің көмегімен немесе бұл үшін стандартты интерфейсті қамтамасыз ету үшін тікелей қолдайды. Осы бағдарламалау тілдерінің кейбіреулері Ruby, Smalltalk, Python, PHP, Objective-C , Delphi, Java және тілдер отбасысы болып табылатын [10] .NET. Сондай-ақ, кірістірілген қолдау жоқ тілдер үшін сериализацияға қолдау көрсететін кітапханалар да бар.

CFML деректер құрылымын <cfwddx> тегпен және JSON-ның SerializeJSON() функциясы арқылы WDDX деректер құрылымын сериализациялауға мүмкіндік береді.

OCaml стандартты кітапханасы Marshal модулі (оның құжаттары) және Pervasive output_value және input_value функциялары арқылы маршалингті қамтамасыз етеді. OCaml бағдарламалау тілі типке статикалық түрде тексерілген кезде, Marshal модулі пайдалану тип кепілдігін бұзуы мүмкін, себебі тексеру тәсілі жоқ болғандықтан, күтілген түрдегі объектілер маршалингтелмеген ағын ұсынады ма? OCaml функцияны немесе функцияны

қамтитын деректер құрылымын (мысалы, әдіс бар нысан) бағыттау қиын, себебі функциялардағы орындалатын код әр түрлі бағдарламалар арқылы берілуі мүмкін емес. (Маршалинг коды позициясы функциясының туы бар, бірақ ол тек сол бір бағдарламада ғана жойылуы мүмкін). Стандартты маршаллинг функциялары ортақ қатынауды сақтап, тумен теңшеуге болатын циклдік деректерді өңдей алады [10].

Кейбір Perl модульдер CPAN механизмдердің сериализациясын қамтамасыз етеді, оның ішінде Storable, JSON: XS және FreezeThaw. Storable Perl деректер құрылымын файлдар мен файлдар немесе Perl скалярлары үшін сериализация және десериализациялау функцияларын қамтиды. Файлдарды тікелей сериализациялаудан басқа, Storable скаляр оралған деректердің іске асырылмаған көшірмесін қайтару freeze функциясын және мұндай скалярдың thaw десериализациясын қамтиды.

Бұл желілік сокет арқылы күрделі деректер құрылымын жіберу немесе оны деректер базасында сақтау үшін пайдалы. Storable арқылы құрылымдарды сериализациялау кезінде желілік қауіпсіз функциялар бар, олар әрқашан өз деректерін аз жылдамдықпен кез келген компьютерде оқылатын пішімде сақтайды. Бұл мүмкіндіктер N-store, N-freeze және т. б. аталды. Бұл құрылымдарды десериализациялау үшін "n" функциялары жоқ - "N" функцияларымен және олардың машинаға тәуелді баламаларымен сериализацияланған, десериализацияланған тұрақты thaw және retrieve құрылымдар.

C және C++ қандай да бір жоғары деңгейлі құрылым ретінде сериализацияны қамтамасыз етпейді, бірақ екі тіл де кірістірілген деректердің кез келген түрін, сондай-ақ екілік деректер түрінде қарапайым ескі деректер құрылымын жазуды қолдайды. Осылайша, сериализацияның таңдамалы функцияларын жазу әдетте тривиальды.

Сонымен қатар, C ++ үшін ODM, ORM жүйесі сияқты шешімдер компиляторында негізделген, шағын өзгерістермен немесе сынып хабарландыруларында өзгеріссіз сериализация кодын автоматты түрде жасауға қабілетті. Басқа танымал сериализация фреймворктары - Boost.Boost Framework бастап Serialization, s11n фреймворк, және Cereal. Microsoft көп функциялы орталықтың құрылымы да өзінің Document-View архитектурасының бір бөлігі ретінде сериализация әдіснамасын ұсынады [10].

Delphi IDE-мен толық интеграцияланған компоненттерді (сондай-ақ, тұрақты объектілер деп аталатын) сериализациялау үшін кіріктірілген механизмді ұсынады . Құрамдастың мазмұны DFM файлына сақталады және ұшуға жүктеледі.

Java Автоматты сериализацияны қамтамасыз етеді, ол объектінің java жүзеге асыру арқылы java.io.Serializable интерфейс белгіленуін талап етеді. Интерфейсті іске асыру класты "сериалдауға болады" деп белгілейді, содан кейін Java ішінде сериализацияны өңдейді. Serializable интерфейсінде сериализация әдістері анықталмаған, бірақ сериализациялық класс белгілі бір

арнайы аттары мен сигналдары бар әдістерді анықтау мүмкін, олар анықталса, сериализация және десериализация процесінің бір бөлігі ретінде шақыртылады. Сондай-ақ, бағдарламалық тіл әзірлеушіге басқа интерфейсті іске асыру арқылы сериализация процесін, Externalizable интерфейсін мұқият айқындауға мүмкіндік береді, ол объектінің күйін сақтау және қалпына келтіру үшін пайдаланылатын екі арнайы әдісті қамтиды. Нысандар әдепкі бойынша серияланбаған және Java сериализациялау механизміне қатынасу үшін Serializable интерфейсін іске асыру керек үш негізгі себебі бар. Біріншіден, барлық объектілер іске асырылмаған жағдайда пайдалы семантиканы бекітпейді.

Мысалы, Thread объектісі ағымдағы JVM жағдайына байланысты. Thread объект десериализацияланған контекст жоқ пайдалы семантиканы қолдайды. Екіншіден, объектінің сериализацияланған жағдайы оның класстарының үйлесімділігі туралы келісімнің бір бөлігі болып табылады. Сериализациялық класстардың нұсқаларының үйлесімділігін қолдау қосымша күш-жігер мен назар аударуды талап етеді. Сондықтан, сериализациялық класты құру әдепкі шарт емес, әдейі дизайнерлік шешім болуы керек. Ақырында, сериализация стационарлық емес жеке сынып мүшелеріне кіруге мүмкіндік береді, олар басқаша қол жетімді емес. Құпия ақпаратты қамтитын класстар (мысалы, пароль) сериализацияланатын немесе экстернализацияланатын болмауы тиіс. Кодтаудың стандартты әдісі объектінің класының дескрипторы және байттар ағынына сериалданған өрістердің негізінде рекурсивті аударманы қолданады. Примитивтер, сондай-ақ ауыспалы емес, статикалық емес объектілер ағынға кодталады. Transient ретінде белгіленбеген өріс арқылы сериалданған объект сілтейтін әрбір объект та сериализациялануы тиіс;сонымен қатар, егер қандай да бір объект сериализацияланбайтын объектілерге толық бағанда қандай да бір объект болса, сериализация қате аяқталады. Әзірлеуші бұл мінез-құлыққа әсер ете алады, өтпелі немесе объект үшін сериализацияны қайта анықтайды, сондықтан бақылау бағанасының кейбір бөлігі қиылысады және сериализаяланбайды [10].

Java объектілерді сериализациялау үшін конструкторды пайдаланбайды. Java объектілерін JDBC арқылы сериализациялауға және оларды деректер базасына сақтауға болады. Свинг кезінде компоненттер Serializable интерфейсін іске асырады,олар Java виртуалды машинасының әр түрлі нұсқаларының арасында төзімділікке кепілдік бермейді. Осылайша, Swing компоненті немесе оның мұратын кез келген компонент байттар ағыны сериализацяланған болуы мүмкін, бірақ ол басқа компьютерде түрлендірілетіні кепілдік емес.

JavaFX-мазмұнды графикалық интерфейсі бар қосымшаларды жасау үшін Java негізіндегі платформа. Операциялық жүйелерден тікелей іске қосылатын үстелдік қосымшаларды құру үшін де, браузерлерде жұмыс істейтін интернет-қосымшалар (RIA) үшін де және мобильді құрылғылардағы қосымшалар үшін де пайдаланылуы мүмкін. JavaFX бұрын қолданылған

Swing кітапханасын ауыстыруға арналған. JavaFX платформасы Microsoft Silverlight, Adobe Flash және ұқсас жүйелермен бәсекелеседі.

Java 11 нұсқасынан бастап Java SE-ге енбейді және Oracle компаниясы әзірлемейді(Gluon компаниясы жеке модуль ретінде қолдау көрсетеді) [1]. Бірақ Oracle Java SE 8 бөлігі ретінде 2022 жылға дейін қажетті өзгерістерді енгізеді.

ECMAScript 5.1 бастап, JavaScript кіріктірілген JSON объектісі мен оның JSON.parse () және JSON.stringify() әдістерін қосты. JSON бастапқыда JavaScript ішкі жиынында негізделген болса да, JSON JavaScript рұқсат етілген емес шекаралық жағдайлары бар. Атап айтқанда, JSON Unicode U + 2028 LINE SEPARATOR және U + 2029 PARAGRAPHSEPARATOR жолдарын тырнақшадағы экрандалған жолдарда көрсетеді, ал ECMAScript 2018 және одан алдынғы нұсқалары оны жасамайды.

.NET Framework Microsoft әзірлеген бірнеше сериализаторы бар. Сондай-ақ, үшінші тұлғалардан көптеген сериализаторлар бар. Тізім үнемі өсуде.

PHP бастапқыда сериализация енгізілген serialize() және unserialize() функциясы арқылы жүзеге асырылады. PHP ресурстардан басқа (файлдық көрсеткіштер, сокеттер және т. б.) деректердің кез келген түрін сериализация алады. Кірістірілген unserialize() функциясы мүлдем сенімсіз деректермен пайдаланғанда жиі қауіпті. Объектілер үшін екі «сиқырлы әдіс» бар, олар класс ішінде іске асырылуы мүмкін - __sleep() және _ _ wakeup () - ішінен serialize() және unserialize() тиісінше, бұл қалпына келтіру және объектіні қалпына келтіру мүмкін. Мысалы, сериализация кезінде деректер базасымен байланысты жабу және десериализация кезінде байланысты қалпына келтіру қажет болуы мүмкін; бұл функционалдылық осы екі сиқырлы әдіспен өңделеді. Олар, сондай-ақ, объектіге сериализацияның қандай қасиеттерін таңдауға мүмкіндік береді. PHP 5.1 бастап объектілер үшін Serializable интерфейс объектілі-бағытталған сериализация механизмі бар [10].

Жалпы, курсивті емес және бөлінбейтін объектілер storeOn протоколын пайдалана отырып, адам үшін ыңғайлы объектіде сақталуы және алынуы мүмкін:/ readFrom:. storeOn: әдістер пайдаланып, бағаланған кезде Smalltalk өрнек мәтінін жасайды; Readfrom бастапқы нысанды қалпына келтіреді. Бұл схема деректердің өздері емес, объектінің рәсімдік сипаттамасын пайдаланатын мағынада ерекше болып табылады. Сондықтан ол өте икемді, бұл кластарға ықшам көріністерді анықтауға мүмкіндік береді. Алайда өзінің бастапқы түрінде ол деректердің циклдік құрылымын өңдемейді және жалпы сілтемелердің сәйкестігін сақтамайды (яғни бір объектіге екі сілтеме екі тең, бірақ бірдей емес көшірмелерге сілтеме ретінде қалпына келтіріледі). Бұл үшін әртүрлі портативті және төзімсіз балама бар. Олардың кейбіреулері Smalltalk немесе класс кітапханаларын нақты іске асыру үшін ерекше болып табылады. Squeak Smalltalk-да объектілерді стерильдеу мен сақтаудың бірнеше жолы бар. Ең қарапайым және ең көп қолданылатын storeOn:/readFrom: және екілік сақтау пішімдері, негізінде smartrefstream

сериализаторлар. Сонымен қатар, байланысты объектілер Image Segments көмегімен сақталуы және алынуы мүмкін, екеуі де "екілік объектілерді сақтау құрылымын" ұсынады, ол сериализацияны және ықшам екілік формадан шығаруды қолдайды [10].

Екеуі де циклдық, рекурсивті және бөлінетін құрылымдарды өңдейді, класс пен метакласса туралы ақпаратты сақтайды, алады және объектінің "ұшуға" көшуі үшін (яғни, объектінің басқа макеті бар класстың ескі нұсқасымен жазылған даналарды түрлендіру үшін) тетіктерді қамтиды. API интерфейстері ұқсас (storeBinary/readBinary), бірақ кодтау бөлшектері әр түрлі, бұл екі форматты үйлеспейтін етеді. Дегенмен, Smalltalk/X коды ашық бастапқы код болып табылады және басқа Smalltalks жүктелген болуы мүмкін. Сериализация объектілері ANSI Smalltalk ерекшелігінің бөлігі емес. Нәтижесінде, объектіні сериализациялау коды Smalltalk жүзеге асыруға байланысты.

Алынған екілік деректер де әр түрлі. Мысалы, Squeak Smalltalk-да құрылған сериализацияланған объект Ambrai Smalltalk-да қалпына келтірілуі мүмкін емес. Сондықтан, объектілерді сериализациялауға сүйенетін Smalltalk бірнеше реализацияларында жұмыс істейтін түрлі қосымшалар осы түрлі реализациялар арасында деректерді бірлесіп пайдалана алмайды. Бұл қосымшалар MinneStore Объектілік деректер базасын және кейбір RPC пакеттерін қамтиды. Бұл мәселені шешу - сериализация үшін XML негізінде пішімді пайдаланатын бірнеше Smalltalks пакеті [10].

Әдетте, Lisp деректер құрылымы «read» және «print» функцияларымен сериалдануы мүмкін. Мысалы, массивтер тізімі бар foo айнымалысы басып шығарылады (print foo). Сол сияқты объект s (read s) атымен ағыннан оқылуы мүмкін. Lisp іске асырудың осы екі бөлігі принтер және оқырман деп аталады. «Print» шыққанын адам оқиды; ол тізімдерді жақшада қолданады, мысалы: (4 2.9 "x" y).

Common Lisp қоса алғанда, Lisp көптеген түрлерінде принтер барлық деректер типін анықтай алмайды, өйткені деректердің барлық түрлерін ұсына алмайды. Мысалы, Common Lisp принтері CLOS объектілерін басып шығара алмайды. Оның орнына бағдарламашы print-object әмбебап функциясына арналған әдісті жаза алады, бұл объектіні басып шығарғанда пайда болады. Бұл Ruby-да қолданылатын әдіс сияқты. Lisp коды оқу синтаксисі деп аталатын оқырманның синтаксисінде жазылған. Көптеген тілдер кодпен және деректермен жұмыс істеу үшін жеке және әртүрлі парсерлерді пайдаланады, Lisp тек біреуін ғана пайдаланады. Lisp коды бар файл деректер құрылымы ретінде саналуы мүмкін, басқа бағдарламамен түрлендірілуі мүмкін, содан кейін, мысалы, read-eval-print циклінде орындалуы немесе жазылуы мүмкін. Барлық оқырмандар және жазушылар циклдік, рекурсивті немесе бөлінетін құрылымдарды қолдамайды [10].

Haskell сериализация Read және Show класс мүшелері болып табылатын түрлерге арналған. Read типі класының мүшесі болып табылатын әрбір типі түсірілетін деректердің жол көрінісінен деректерді алатын

функцияны анықтайды. Show типтері класы, өз кезегінде, объектінің жол көрінісі түзілуі мүмкін show функциясы бар. Қарапайым жарнама типін программистке функцияны айқын анықтаудың қажеті жоқ, ол Read туындысы немесе show туындысы болып табылады, немесе екеуі де компиляторды көптеген жағдайларға сәйкес функцияларды генерациялау мүмкін (бірақ, барлығы үшін емес: мысалы, функциялар түрлері автоматты түрде Show немесе Read типтерін қабылдай алмайды) [10].

Show үшін автоматты түрде жасалған экземпляры жарамды бастапқы кодты жасайды, сондықтан Haskell мәнін show жасаған кодты, мысалы, Haskell интерпретаторында іске қосып, пішімдеуге болады. Тиімді стерильдеу үшін Haskell кітапханалары бар, олар екілік форматта жоғары жылдамдықты сериализацияны орындауға мүмкіндік береді.

Windows PowerShell сериализацияны Export-CliXML кірістірілген командлетінің көмегімен іске асырады. Export-CliXML .NET объектілерін сериализациялайды және алынған XML файлын файлда сақтайды. Объектілерді қайта жасау үшін Import-CliXML командлетін пайдаланылады, ол XML-ден экспортталған файлға десериализацияланған объектіні жасайды. "Меншік қаптары" ретінде жиі белгілі десериализацияланған объектілер тірі объектілер емес; бұл қасиеттері бар, бірақ әдістері жоқ суреттер. Екі өлшемді деректер құрылымы Import-CSV және Export-CSV командлеттерін қолдану арқылы CSV форматында сериалдауға болады [10].

Julia сериализацияны serialize()/ модульдері арқылы жүзеге асырады, сол Julia нұсқасында және/немесе бір жүйе түрінде жұмыс істеуге арналған. Пакет құжатталған форматты және түрлі тілдер үшін орама салынған жалпы кітапхананы пайдалана отырып, неғұрлым тұрақты балама ұсынады, ал сериализация форматы әдепкі бойынша әзірленген, ал есте желі бойынша байланыс үшін ең жоғары өнімділігімен.

Python - жалпы мақсаттағы бағдарламалау тілі. Python ядросының синтаксисі ең аз. Сонымен қатар стандартты кітапхана пайдалы функциялардың үлкен көлемін қамтиды.

Python құрылымдық, объектілі-бағытталған, функционалдық, императивті және аспектілі-бағытталған бағдарламалауды қолдайды. Негізгі архитектуралық ерекшеліктер - динамикалық типизация, жадыны автоматты басқару, толық интроспекция, ерекшеліктерді өңдеу механизмі, көп ағынды есептеулерді қолдау, деректердің жоғары деңгейлі құрылымы. Өз кезегінде, пакеттерге бірігуі мүмкін модульдерге бағдарламаларды бөлу қолдайды.

Python динамикалық типтеуді қолдайды, яғни айнымалы түрі тек орындау кезінде анықталады. Сондықтан "айнымалы мәнін тағайындау "орнына" кейбір атауымен мәнді байланыстыру " туралы айту жақсы. Python-да енгізілген түрлері бар: түйреуіш, жол, Unicode-жол, еркін дәлдіктің бүтін саны, өзгермелі үтірмен Сан, кешенді сан және т.б. Python коллекциясынан енгізілген: тізім, кортеж (өзгермейтін тізім), сөздік, көптеген және басқалар. Барлық мәндер объектілер болып табылады, оның ішінде функциялар, әдістер, Модульдер, сыныптар.

Жаңа түрді қосу класты (class) жазу немесе кеңейту модулінде жаңа түрді (мысалы, C тілінде жазылған) анықтау арқылы болады. Кластар жүйесі мұраға (жалғыз және көпше) және метапрограммалауға қолдау көрсетеді. Көптеген кірістірілген түрлер мен кеңейтулердің типтерінен мұрагерлік болуы мүмкін.

Барлық объектілер сілтемелік және атомдық болып бөлінеді. Атомдық int, long (3 нұсқасында int кез келген саны, себебі 3 нұсқасында өлшемге шектеу жоқ), complex және басқа да. Атомдық объектілерді беру кезінде олардың мәні көшіріледі, ал сілтемелік үшін тек объектіге көрсеткіш көшіріледі,осылайша, берілгеннен кейін екі айнымалылар бірдей мәнді пайдаланады. Сілтеме нысандары өзгертілетін және өзгермейтін болады. Мысалы, жолдар мен кортеждер өзгермейтін, ал тізімдер, сөздіктер және басқа да көптеген нысандар өзгеретін болады. Python-да Кортеж шын мәнінде өзгермейтін тізім. Көптеген жағдайларда кортеждер тізімдерден жылдамырақ жұмыс істейді, сондықтан сіз тізбекті Өзгертуді жоспарламасаңыз,онда оларды пайдалану жақсы.Python-белсенді дамып келе жатқан бағдарламалау тілі, тілдік қасиеттерді қосу/өзгерту жаңа нұсқалары шамамен екі жарым жылда бір рет шығады. Тіл ресми стандарттауға ұшырамады, де-факто стандартының рөлін Тіл авторының бақылауымен жасалған CPython орындайды [10].

Python қарапайым екілік кодтау декодеріне негізделген (SBE).

Sbedecoder - SBE кодталған деректерді талдау үшін қарапайым python пакеті. sbedecoder XML пішіміндегі SBE парсерін динамикалық жасайды. Бұл sbeschema() данасын жасау және файл атымен оның parse() әдісін шақыру арқылы қол жеткізіледі:

```
from sbedecoder import SBESchema
schema = SBESchema()
schema.parse('path/to/schema.xml').
```

Бұл объектілер мен деректердің сериализациясын тікелей қолдайтын объектілі-бағытталған бағдарламалау тілдерінің бөлігі болды.

Python әзірлеу үшін бірнеше арнайы IDE бар. JetBrains PyCharm-Python үшін толық функциялы IDE, Windows, Mac OS X және Linux платформаларында қол жетімді, тегін (Community) және ақылы (Professional) нұсқаларда бар.PyCharm-Python бағдарламалау тілі үшін интеграцияланған даму ортасы. Кодты талдау құралы, графикалық реттеуші, юнит-тестерді іске қосу құралы және Django веб-әзірлемесін қолдайды. PyCharm IntelliJ IDEA негізінде JetBrains компаниясымен жасалған [10].

PyCharm Windows, Mac OS X және Linux операциялық жүйелерімен жұмыс істейді.

Мүмкіндіктері:

– Кодты статикалық талдау, синтаксис және қателерді жарықтандыру.

– Жоба және бастапқы код бойынша Навигация: жобаның файлдық құрылымын көрсету, файлдар, сыныптар, әдістер мен әдістерді пайдалану арасында жылдам өту.

– Рефакторинг: атауын өзгерту, әдісті шығару, айнымалыны енгізу, константаны енгізу, әдісті көтеру және түсіру және т. б.

– Django фреймворк пайдалану веб-әзірлеу құралдары

– Python үшін кірістірілген реттеуші

– Юнит-тестілеу үшін кірістірілген құралдар

– Google App Engine пайдаланып әзірлеу

– Нұсқаларды бақылау жүйесін қолдау: Mercurial, Git, Subversion, Perforce және CVS үшін жалпы пайдаланушы интерфейсі [10].

1.3 SBE архитектурасы және хабар сұлбаларын іздеу

1.3.1 Деректерді сериализациялау және десериализациялау

Компьютерлік ғылымдар саласында деректерді сақтау контекстінде сериализация - бұл деректер құрылымын немесе объектінің күйін сақтау (мысалы, файл немесе жад буферінде) немесе жіберу (мысалы, Желіге қосылу арқылы) және кейінірек қалпына келтіру (мүмкін, басқа компьютерлік ортада) мүмкін форматқа түрлендіру процесі. Нәтижесінде биттердің қатары сериализация форматына сәйкес қайта есептелгенде, оны бастапқы объектінің семантикалық бірдей клонын жасау үшін пайдалануға болады. Процесс сілтемелерін кеңінен қолданатын көптеген күрделі объектілер үшін бұл оңай емес. Объектілі-бағытталған объектілердің сериализациясы бұрын байланысты болған әдістердің ешқайсысын қамтымайды. Нысан сериализациясының бұл процесі нысанның маршалингі немесе «marshaling» деп аталады. Байттар сериясынан деректер құрылымын алатын қарама-қарсы операция десериализация («unmarshalling») болып табылады [11]. Пайдаланылады:

— Сымдар бойынша деректерді жіберу тәсілі (хабар алмасу);

— Деректерді сақтау тәсілі (деректер базасында, қатқыл дискілерде);

— Мысалы, SOAP сияқты қашықтағы процедураларды шақыру әдісі;

— Объектілерді бөлу әдісі, әсіресе COM, CORBA және т. б. сияқты компоненттер негізінде бағдарламалық қамтамасыз етуді әзірлеу;

— Уақыт өзгеретін деректердің өзгерістерін анықтау әдісі.

Осы функциялардың кейбірі пайдалы болуы үшін архитектурадан тәуелсіз болу керек. Мысалы, басқа аппараттық архитектурада жұмыс істейтін компьютердің таралуын барынша пайдалану үшін байттар тәртібіне қарамастан сериалданған деректер ағынын сенімді қалпына келтіру мүмкіндігі болуы тиіс. Бұл деректер құрылымының жад құрылымын тікелей көшірудің қарапайым және жылдам процедурасы барлық архитектура үшін сенімді жұмыс істей алмайтынын білдіреді. Деректер құрылымын архитектурадан тәуелсіз форматта сериалдау байттарды реттеу, жад белгілеуді немесе әр түрлі бағдарламалау тілдерінде деректер құрылымдарын

ұсынудың әртүрлі тәсілдерін болдырмауды білдіреді [11].

Кез-келген сериализация схемасына сипаттама бойынша деректерді кодтау тізбекті болып табылатындықтан, сериалданған деректер құрылымының бір бөлігін алу үшін барлық нысанның басынан аяғына дейін оқылуы және қайта жаңартылуы талап етіледі. Көптеген қосымшаларда бұл желілік артықшылық болып табылады, өйткені ол объектінің күйін сақтау және жіберу үшін қарапайым жалпы енгізу-шығару интерфейстерін пайдалануға мүмкіндік береді. Жоғары өнімділік маңызды қосымшаларда күрделі желілік емес сақтау ұйымын шешу үшін көп күш жұмсаудың мағынасы болуы мүмкін.

Тіпті бір компьютерде қарапайым нысандарды сақтау үшін ол тым нәзік көрсеткіштер, себебі олар көрсететін объектілер жадындағы басқа орынға қайта жүктелуі мүмкін. Мұны орындау үшін, сериализация процесі көрсеткіштің ыдырауы немесе ыдырауы деп аталатын кезеңді қамтиды, онда көрсеткіштің тікелей сілтемелері аттарға немесе позицияларға негізделген сілтемелерге айналады. Десериализация процесі «swizzling» көрсеткіші деп аталатын кері қадамды қамтиды [11].

Сериализация және десериализация жалпы кодқа негізделгендіктен (мысалы, Microsoft Foundation Classes Serialize функциясы), жалпы код бір мезгілде басқа да қызметтерді атқара алады және осылайша, біріншіден, сериализацияланған нысандар мен олардың алдыңғы көшірмелері болып табылатын объектілер арасындағы айырмашылықтарды анықтайды және, екіншіден, осындай келесі анықтау үшін енгізуді қамтамасыз етеді.Алдыңғы көшірмені жасаудың қажеті жоқ, себебі айырмашылықтар тасымал кезінде көрінуі мүмкін. Бұл техника *дифференциалды орындау* деп аталады. Бұл уақыт ішіндегі мазмұны өзгеретін пайдаланушы интерфейстерін бағдарламалау кезінде пайдалы - осы тапсырмаларды орындау үшін жеке код жазусыз кіріс оқиғаларын өңдеу үшін графикалық нысандар жасауға, жоюға, өңдеуге немесе жасауға болады [11].

1.3.2 FIX SBE қаржылық ақпарат алмасуы

«eXchange (FIX) қаржылық ақпарат» хаттамасы ең жиі қолданылатын электрондық байланыс хаттамаларының бірі [12].

FIX хаттамасы қаржы нарықтарын сатып алуда да, сатуда да кеңінен қолданылады. Оның пайдаланушылары арасында - өзара қорлар, инвестициялық банктер, брокерлер, қор биржалары және ECN бар [12].

Электрондық сауда хаттамаларының көп бөлігін FIX хаттамаларының жанұялары құрайды. Негізінен олар Америка мен Еуропада, Азия елдерінің аздаған бөлігінде қаржы биржаларында электронды алмасушылармен пайдаланылады.

FIX хаттамаларының соңғы әзірлемелеріне өңдеу протоколы және Simple Binary Encoding үлкен көлемді деректерді жылдам жіберу хаттамалары кіреді [12].

Қарапайым екілік кодтау немесе Simple Binary Encoding (SBE) - нарықтық деректерді тарату саласында қолданылатын жоғары өнімді екілік кодтау болып табылады [13].

SBE FIX сауда қауымдастығында жоғары деңгейдегі жұмыс тобы құрған хаттамалар тобының бөлігі болып табылады. SBE-төмен кідірісі бар қолданбаларды қолдау үшін екілік пішімдегі хабарларды кодтау/декодтау үшін OSI 6 ұсыну деңгейінің хаттамасы [13].

FIX Trading Community стандарттар корпусындағы жоғары өнімді транзакциялар мен деректер арналарын қолдайтын FIX электрондық коммерция хаттамасының нұсқасы компанияларға жоғары жылдамдықты жабдықтауды, бағдарламалық жасақтаманы және желілік қосылымдарды қолдауға көмектесу үшін FIX Trading Community стандарттар органын ресми түрде іске қосты [13].

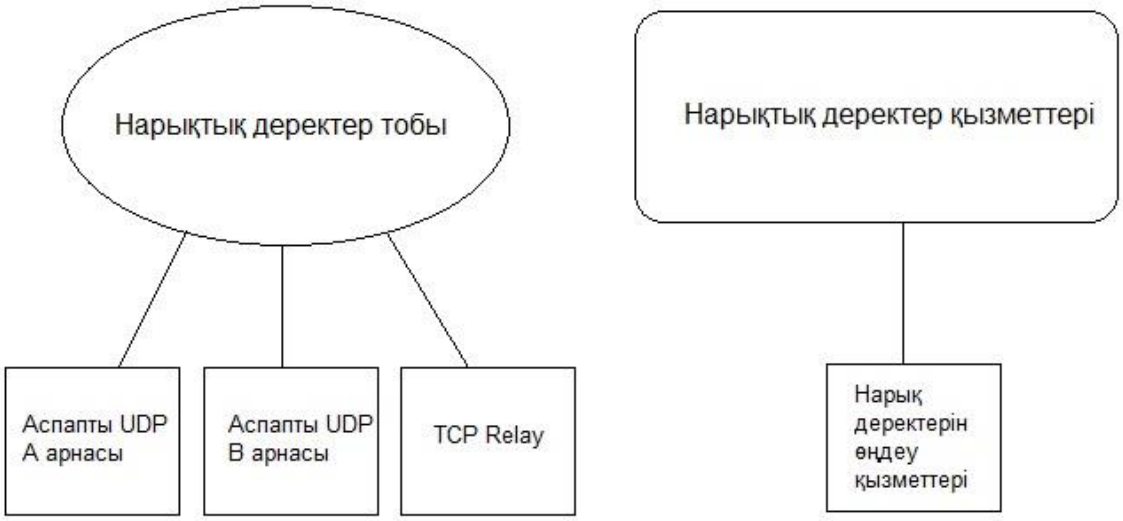
Жаңа нұсқа ондаған наносекундқа хабарламаларды кодтау мен декодтау қоса алғанда, аз кідіріспен жүзеге асыру үшін оңтайландырылды дейді ресми көздер. SBE протоколы CME Group, Moscow Exchange, Thomson Reuters және т.б. сияқты компанияларда табысты пайдаланылуда. Жаңа FIX нұсқасы хабар кітапханалары үшін Real Logic әзірленген ашық бастапқы коды бар нұсқаларға ие.

SBE хаттамасы басқа екілік кодтауға қарағанда әр түрлі сипаттамаларды қамтамасыз етеді. Ол төмен латенттілік үшін оңтайландырылған.

1.0 нұсқасы стандарт жобасы 2016 жылғы 9 ақпандағы жаһандық техникалық комитетінен SBE 1.0 техникалық сипаттамасының нұсқасына дейін көтерілді. Бұл 1.0 нұсқасының соңғы ерекшелігі. SBE стандарттары Git Hub және FIX Trading Community сайтында қол жетімді. SBE 1.0 site нұсқасы ерекшелігі 2018 жылдың 27 шілдесінде жарияланды [13].

2.0 RC1 нұсқасы жаһандық техникалық комитеті 2018 жылы 16 тамызда 90 күндік қоғамдық шолу үшін мақұлданды. Мүмкіндіктер танымал сұраныстың арқасында қосылды. Ол 1.0 нұсқасымен үйлесімді емес болғандықтан, сым пішімінің елеусіз өзгеруі себебімен, ол негізгі нұсқа ретінде әзірленген [13].

Қысқа хабарламалар өлшемдері пайдаланылады, бірақ деректерлі ықшамдаудың өңдеуі құны жоқ. Барлық семантиканы FIX қолдайды. Кодтау стандарты сеанс хаттамасы мен қосымша деңгейі үшін басқа FIX стандарттарын толықтырады.



Сурет 1.1 - SBE нарықтық деректердің оңтайландырылған архитектурасы

Нарықтық деректер тобы. UDP A және B. UDP-A арнасы және UDP-B арнасы SBE шифрланған FIX хабарламаларын пайдалана отырып, инкрементттік нарықтық деректерді тарату үшін пайдаланылады. FIX хабарламаларының барлық түрлері UDP Feed A және UDP Feed B қолданылатын нарықтық деректер топтары арқылы жіберіледі. Бұл екілік UDP байланысты хабарламаларды жоғалту ықтималдығын азайтады. Әрбір SBE хабарламасы екі арна арқылы жіберіледі (Сурет 1.1) [13].

TCP Replay қалпына келтіру. Клиенттік жүйелер TCP болжамының реттік нөмірі мен компонентін пайдалана отырып жіберілген белгілі бір хабарламаларды қалпына келтіруі мүмкін. TCP тарихи ойнату компоненті жүйелерге UDP инкрементттік нарығының деректер арнасында жарияланған хабарларды қайталауды сұрауға мүмкіндік береді. Сұрауларды жүзеге асыру үшін хабарларды көрсетеді. SBE Market Data Request (тег 35-MsgType = V) сұрауы пайдаланады [13].

Бұл сұраудың түрі клиенттік жүйелермен орнатылған жаңа TCP байланысы арқылы жіберіледі. Жауаптар сол байланыс арқылы CME Group жіберіледі және қайта жіберу аяқталғаннан кейін CME Group қосылымы жабылады. Барлық жауаптар SBE хаттамасымен кодталады (ауытқу жауабын қоса алғанда).

Хабарламаларды TCP Historical Replay арқылы сұрағанда келесі шектеулер қолданылады:

Нарықтық деректердің сұрау салуында (35=V) 2000-нан артық емес хабарлама сұратылуы мүмкін.

Ағымдағы күннің сұраулары ғана сұралуы және қайтадан жіберілуі мүмкін [13].

1.3.3 Қаржылық ақпаратты пайдаланушылар

EXchange (FIX SBE) қаржылық ақпараты жеткізушілерге қатысты электрондық коммуникациялық хаттама жеткізушілерге қатысты нақты уақытта бағалы қағаздармен алмасу операцияларымен халықаралық алмасу үшін бейтарап болып табылады, бұл қорлар, инвестициялық менеджерлер мен фирмалар үшін пайдалы.

FIX SBE жүйелері валюта айырбастау қорлары арқылы және олардың арасында бағалы қағаздармен жасалған мәмілелер туралы нақты және уақтылы қаржылық ақпаратты береді. Оны пайдалану қолданушыларға уақытылы және дәл шешімдер қабылдауға мүмкіндік береді.

FIX SBE саудаға дейінгі, саудадағы және саудадан кейінгі алмасулар үшін, сондай-ақ АҚШ-тың нормативтік есептемесі үшін де-факто хабарлама алмасу стандартына айналды. Ол барлық кең таралған желілік технологиялармен үйлесімді [14].

FIX Protocol, Ltd. Корпорациясы FIX жүйесін иеленеді және қолдайды. Компания осы мақсатқа жету және жүйенің қоғамдық игілікте қалуын қамтамасыз ету үшін толығымен құрылды [14].

FIX байланысы мәтіндік хабарлар мен электрондық поштаны, бағалы қағаздарды бөлуді, жаңалықтар, өтінімдер мен өзгерістерді, жарнама мен есептерді орындауды қамтиды. Негізінен кәсіпорындар арасындағы өзара іс-қимыл үшін пайдаланылады, ол іскерлік хабарламалар мен транзакция ағынын жақсартуға арналған. FIX бұл мақсатқа жетуде, артық функцияларын азайту және телефон байланысына, жазбаша хабарламаларға, транзакцияларға және құжаттарға жұмсалатын уақытты қысқарту жұмыстарын жүзеге асыруда.

1992 жылы Salomon Brothers және Fidelity Investments арасында акцияларды сату үшін енгізілген FIX хаттамасы опциондар мен фьючерстермен алмасуда стандартқа айналады. Ол неғұрлым тиімді және есеп беретін транзакцияларды қамтамасыз ету және жазбаларды жүргізу үшін жүзеге асырылды, ол негізінен телефон арқылы өңделген жүйені алмастырды. Ескі жүйеде қызығушылық белгілері жиі «күту» режимінде жоғалды немесе дұрыс емес трейдерге жіберілді. FIX сауда-саттық және сауданы жүзеге асыру үшін стандартты электрондық хаттамаға айналды [13].

Дүниежүзілік банктер арасында қаржылық телекоммуникация (SWIFT) бэк-кеңседе хабар алмасу стандарты болып табылатынын назарға ала отырып, FIX фронт-кеңсенің хабар алмасу стандарты болып табылады [14].

FIX қаржы нарықтарында сатып алуды жақтаушыларда да, сату жағында да (брокерлер/дилерлер) танымал. Пайдаланушылар өзара қорларды, инвестициялық банктерді, брокерлерді, қор биржаларын және басқа да электрондық байланыс желілерін (ECNs) қамтиды.

Негізінен ол акциялармен операциялар үшін қолданылады, бірақ ол облигациялармен, шетел валютасымен және деривативтермен операцияларды өңдей алады [15].

FIX Trading Community™ мүше-фирмалардың бірлескен күш-жігері FIX хабар алмасу стандартын қолдайды және осы хаттаманың ары қарай дамытуын жалғастырады.

FIX қауымдастығының мүшелері бүкіл әлем бойынша бірнеше жетекші қаржы мекемелерін қамтиды. Осы мүше-фирмалардың жұмысы стандарт жаңа және пайда болатын сауда талаптарын қанағаттандыру үшін дамып келе жатқанына кепілдік береді. Олардың әрекеттері бүкіл әлемде FIX енгізуге ықпал етеді. FIX протоколының өзі патенттелмеген, тегін және ашық стандарт болып табылады [15].

FIX-тұрақты, өзгермелі мән, сонымен қатар, саладағы және технологиялардағы өзгерістер туралы хабардар болуға ұмтылады. 2018 жылдың басындағы жағдай бойынша қатысушылар киберқауіпсіздікті, сандық валютаны және бұғаттауды, орындаудың ашықтығын және өнімділікті арттыруды қамтитын ағымдағы проблемалар мен мәселелерді талқылайды.

FIX (SBE) қарапайым екілік кодтау жоғары өнімді сауда жүйелеріне арналған, бірақ басқа жоғары өнімді қосымшалар үшін де қолайлы. Ол төмен кідіріспен кодтау және декодтау үшін оңтайландырылған [15].

Бұл кодтау ерекшелігі хабарларға арналған сымды хаттаманы (көрініс деңгейі) сипаттайды. Осылайша, ол өзара іс-қимыл жасайтын тараптар арасындағы өзара іс-қимыл үшін стандартты қамтамасыз етеді. Пайдаланушылар бұл стандартты өздерінің қажеттіліктеріне ең жақсы сәйкес келетін етіп қолдана алады. Кодтау стандарты сеанс протоколы мен бағдарлама деңгейінің мінез-құлқы үшін басқа FIX стандарттарына қосымша болып табылады. SBE FIX бүкіл семантикасын ұсынуға арналған.

Типтердің бинарлық жүйесі. FIX дәстүрлі семантикасын қолдау үшін барлық құжатталған өрістер түрлерін қолдайды. Алайда, кодтаудың басылатын символдық көріністерінің орнына *fix tag = value* типтердің жүйесі деректердің екілік типтерімен байланысады және қажет болған жағдайда туынды түрлерін анықтайды [15].

Екілік түрлер жүйесі келесі жолдармен жетілдірілді:

Ондық сандардың және уақыт белгілерінің дәлдігін, сондай-ақ рұқсат етілген сандардың диапазондарын көрсету үшін құралдарды ұсынады. Бұл әдіс белгіленген ұзындықтағы символды массивтерді айнымалы ұзындықтағы жолдардан ажыратады.

Көптеген таңдау бар тізбектер мен өрістердің келісілген жүйесін қамтамасыз етеді.

Дизайн принциптері:

SBE жобасы күрделі түрлендірулерсіз немесе шартты логикасыз деректерге тікелей қатынауға ұмтылады. Бұл арқылы қол жеткізіледі:

Бағасы және уақыт белгілері сияқты меншікті екілік файлдардан алынған деректердің екілік типтерін және қарапайым түрлерін пайдалану.

Ретімен өңделуі керек тікелей деректерге қол жетімділігін қолдап және көптеген элементтерді айнымалылар ұзындығында басқарудың қажеттілігін

болдырмайтын, тіркелген позиция және өрістері бекітілген ұзындықтың артықшылығын қолдайды [15].

Хабарлама схемасы. Бұл стандарт өрістердің қалай кодталатынын және жалпы хабар құрылымын сипаттайды. Хабар түрінің мазмұны хабарлама сұлбасымен анықталады. Хабарлама сұлбасы хабарлардың қай өрістері тиесілі екенін және олардың хабарламада орналасуын хабарлайды. Сонымен қатар, метадеректер рұқсат етілген мәндер ауқымын және тұрақты мәндер сияқты сымдарға жіберілмейтін ақпаратты сипаттайды.

Хабарлардың схемалары FIX хабарларының стандартты ерекшеліктеріне негізделуі мүмкін немесе контрагенттер арасындағы келісімге сәйкес бапталуы мүмкін.

Барлық хабарлар баспагерлері мен тұтынушыларын бір уақытта жаңартудың қажеті жоқ. Белгілі бір шектеулер шеңберінде хабарламалар сұлбалары мен сым пішімдері бақыланатын түрде кеңейтілуі мүмкін. Егер бизнес өңдеу үшін қосылған өрістерді немесе хабарларды түсіндіру қажет болмаса, схеманың ескі нұсқасын пайдаланатын тұтынушылар үйлесімді болуы тиіс.

Қарапайым екілік кодтау стандартын әлемдік сауда орталығында орналасқан стандарттау жөніндегі коммерциялық емес салалық орган FIX Trading Community жоғары өнімді жұмыс тобы әзірледі [15].

1.3.4 Криптовалюта деректері және оның маңыздылығы

"Криптондық валюта" ұғымы валютаның сандық түрлеріне жатады. Оны криптографиялық әдістердің көмегімен жасайды және береді, негізінен блокчейн технологиясы негізінде. "Монеталар" бастапқыда электронды түрде математикалық есептеулерді жүргізу есебінен шығарылады. Қарапайым сөздер, крипто-валюта-бұл ресми курсы бар нақты ақшаға теңесетін жасанды төлем жүйесі [16].

Криптовалюта - бұл әлемді өзгертуге қабілетті тағы бір технология. Бұл ұғым бірнеше жыл бұрын пайда болды. Алайда, технология өте пайдалы және түрлі салаларда сұранысқа ие болды. Күнделікті төлемдерден бастап ұлттық мүдделерді қорғауға дейін.

Криптовалютаның құрылу тарихы Bitcoin құрылуынан басталды. Биткоин-крипто-валютаның ең бірінші кезеңі. Ол бүгінгі таңда ең танымал және ең қымбаты [16].

Сурет 1.2 - Биткоин

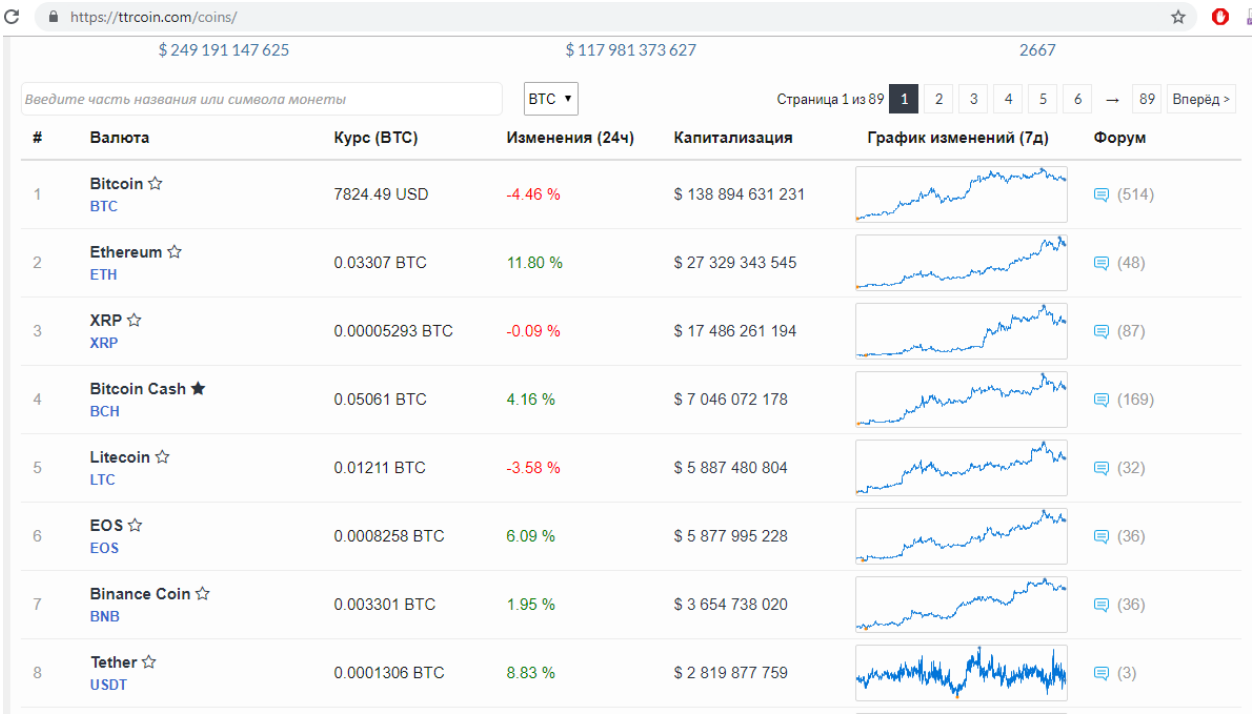
Барлығы 2008 жылғы қаржы дағдарысының ең қызған тұсында басталды. Сол жылы энтузиастар тобы саясатқа тәуелді болмайтын әмбебап төлем шешімін құруды шешті. 2008 жылдың 31 қазанында программистерге арналған сайттардың бірінде мақала пайда болды [17]. Ол мақала «Биткоин: пирингті жүйесіне электрондық қолма-қол ақша» - деп аталды. 9 қаңтар 2009 Bitcoin-әмиянның бірінші нұсқасы шығарылды және алғашқы Bitcoin пайда болды. Іс жүзінде - бұл алғашқы крипто-ақша.

Бірнеше ай бойы осы жүйенің белгілі бір пысықтаулары жүргізілді. Көптеген түрлі параметрлер мен танымал linux операциялық жүйесін қолдау пайда болды. Сол сәтте, биткоин оның әзірлеушілері мен тестерлерінің тобына ғана белгілі болды. Сондықтан, жүйені әзірлеушілер жұртшылықтың назарын аудартуды шешті. 2009 жылдың қараша айында биткоиндермен есеп айырысу туралы форум ашылды. Биткоиндердің танымалдығы аздап өсе бастады. Сонымен қатар, талқылауда бірінші криптовалютасын жақсарту бойынша жаңа идеялар пайда болды.

2010 жылы әзірлеушілер криптовалютасының бірнеше осалдығын тапты және жойды. Биткоиндарды жіберу жылдамдығы ұлғайтылды. Осыдан кейін, Bitcoin-әмияндар саны өсе бастады. 2010 жылдың мамыр айында биткоиндердің нақты тауарға алғашқы алмасуы болды [17].

Содан кейін, американдық Ласло Ханеч 10 000 биткойндарды екі пиццаға айырбастады. Шын мәнінде, бұл ақша ретінде битокиндерді алғашқы пайдалану болды. 2012 жылдан бастап, Bitcoin-дардың дамуын қамтамасыз етуші американдық Bitcoin Foundation компаниясы жетекшілік етеді. Бұл компанияның бас әзірлеуші - Гэвин Андресен. Дәл қазір оны жобаның жетекшісі деп санауға болады. Бүгінгі күні 250-ден астам әртүрлі крипто-валюта бар екені белгілі. https://ttrcoin.com/coins/ сайтынан алынған

2019 жылдың мамыр айындағы криптовалюта биржасының күйі (сурет 1.3) [18].



Сурет 1.3 – Криптовалютаның бағамдары

Жаңа криптовалюта сандық түрде шығарылады. Кез келген адам крипто-валюта ала алады. Жаңа крипто-валюта өндіру процесі майнинг деп аталады. Майнингпен айналысу үшін компьютердің есептеу қуаты пайдаланылады. "Майнинг" сөзі бізге ағылшын тілінен келді. "Пайдалы қазбаларды өндіру"деп аударылады.

"Криптондық валюта" термині алғаш рет Forbes журналының жариялауынан бастап 2011 жылы талқылана бастады. Содан бері атауы кіріске берік кірді және қағаз банкноттар немесе металдан жасалған монеталар түріндегі өрнектері жоқ коэндерге (сөзбе-сөз тиындар) қатысты қолданылады. Ақшаның мұндай түрі тек сандық кеңістікте бар.[17]

Басқа электрондық төлем жүйелеріне қарағанда, криптондық валюта бастапқыда нақты ақшасыз пайда болады. Коиндердің белгілі бір сомасының иесі болу үшін оларды құру сервисіне қосылу, бірыңғай майнинг желісінің қатысушысы болу және өзінің "табысын" күту жеткілікті. Бұл криптовалютасының нақты ақшадан негізгі айырмашылығы - соңғысы РФ Орталық Банкінің шешімі бойынша қатаң шығарылады. https://www.rbc.ru/ ресми сайтынан алынған соңғы 5 жылдағы bitcoin-ның сату бағамының диаграммасы (сурет 1.4) [18].

Сурет 1.4 – Соңғы 5 жылдағы биткоинның бағамының өсуі

Соңғы он жылда криптовалютаның ақпараттық технологиялар салаларында маңызды рөл атқарады. Осыған байланысты жіберілген және қабылданған мәліметтердің саны жылдам артуы байқалады және деректерді жіберу жылдамдығын арттыру қажеттілігі туындайды. Осыған сәйкес, зерттеудің практикалық маңыздылығын атап кетсек, бұл құрастырылған бағдарлама ақпарат алудың кідірулерінің төмендеуі мен деректердің үлкен көлемін жедел түрде жіберу мүмкіндігін қамтамасыз етеді. Біздің еліміздің көптеген аймақтарында ақпараттық технологиялардың таралуы өте жоғары. Криптовалютаның қор биржасы осы салалардың бірі болып табылады. Қор нарығы бүгінгі таңда жоғары технологиялық салаларының бірі болып табылады. Ал осы прогрестің негізінде жоғары жылдамдықтағы байланыс арналарын жоғары жүктемемен күресетін бірнеше сауда алаңдары, брокерлік жүйелер құрылады. Бұдан, үлкен көлемдегі ақпаратты жіберу үшін деректердің үлкен компрессиясын қамтамасыз ететін қосымшаларға деген сұраныс тек қаржылық жүйеде ғана емес, сонымен қатар маркетинг, бизнес, телекоммуникация, сауда, логистика және үкіметте өзектілігі мол. Статистикаға сүйенсек, сақталған және алынған деректердің көлемі жыл сайын өсіп келеді, сонымен бірге жылдам деректерді жіберу мәселесі өте маңызды және ол XXI ғасырда үлкен рөл атқарады. Жұмыстың ғылыми жаңалығы - үлкен көлемдегі қаржы криптовалюта деректерін жедел түрде кешіктіріп жіберуін азайту әдісін әзірлеу, бұл декодталған және десериализацияланған деректерді дерекқорда сақтауға және жапондық шамдар түрінде ұсынуға мүмкіндік береді [17].

1.3.5 Биржалық графиктердің жұмыс жасау принциптері

Биржалық графиктер нарықтық талдаудың маңызды құралы болып табылады, өйткені баға туралы ақпаратты графикалық, визуалды нұсқада ұсынады, ол мәтіндік немесе сандық ақпаратқа қарағанда нарықтағы жағдайды жалпы түсіну үшін неғұрлым қарапайым болып табылады. Биржалық графиктер тобырдың жаппай мінез-құлқын көруге, сондай-ақ сатушылар мен сатып алушылар арасындағы күштердің орналасуын бағалауға мүмкіндік береді, бұл ақыр соңында мәміле табыстылығының әлеуетін түсінуге мүмкіндік береді [19].

Биржалық графиктердің түрлері. Графикалық құнның үш негізгі түрі бар-сызық, бар және жапон шам. Осы құралдардың әрқайсысы тиісті кезеңнің ашылу бағасы туралы, жабу бағасы туралы, ең төмен және ең жоғары бағалар туралы (жабу бағасының негізінде ғана құрылатын желілік кестеден басқа) ақпарат береді. Трейдерлер пайдаланатын кестелер белгілі бір уақыт бөлігі үшін, мысалы, 5, 15, 60 минут үшін, бір күн, апта немесе ай үшін бағаның өзгергенін куәландырады.

Тарихи биржалық графика нарықта басым бәсекелестер арасындағы күрес тарихы ретінде басқа нәрсе емес. Мұндай күрес нәтижесінде мәмілелердің үлкен саны жасалады, олардың әрқайсысы кестеде міндетті түрде көрсетіледі – бір тик осындай мәмілеге сәйкес келеді. Баға өскен кезде, бұл біреу шығынға ұшырайды, ал біреу табыс табады дегенді білдіреді.

Мұндай модельдер кері болуы мүмкін, яғни олар пайда болғаннан кейін баға өзінің бағытын қарама-қарсы немесе жалғастыру фигураларына өзгертуі мүмкін, яғни олардың қалыптасуы нәтижесінде баға сол бағытта әрі қарай да қозғалатын болады (баға формацияларының мысалдары – бас пен иық, қос түбі, үш ұшы, Ту мен жалауы, үшбұрыш және басқалар).

Биржалық кестелер ағымдағы нарықтық жағдайды талдаудың және бағаның одан әрі мінез-құлқын болжаудың маңызды құралы болып табылады. Олар бұқа мен аюлар арасындағы күштер балансын анықтауға және біреулердің пайдасына таңдау жасауға мүмкіндік береді. Бұдан басқа, биржалық кесте мәмілеге кіру туралы шешім қабылдау үшін талданатын құрал бойынша өсу (немесе құлау) әлеуетін бағалауға көмектесе алады [19].

Техникалық талдауды зерттеу олардан басталады.модельдер кері болуы мүмкін, яғни. олар пайда болғаннан кейін баға өзінің бағытын қарама-қарсы немесе жалғастырушы фигураларға өзгерте алады, яғни олардың қалыптасуы нәтижесінде баға ары қарай да осы бағытта қозғалады (баға формацияларының мысалдары – басы мен иығы, қос түбі, үш ұшы, жалау мен жалау, үшбұрыш және т.б.). Биржалық саудада қазіргі уақытта кестелердің үш негізгі түрі қолданылады: сызықтық, барлар және жапон шамдары.

Сызықтық кесте. Биржалық баға белгілеулерді ұсыну үшін ең қарапайым және түсінікті. Дәл осындай графиктер барлығы мектепте, институтта сурет салды. Кестеде кезеңдердің жабылу бағасына тең нүктелер қойылады. Және бұл белгілер сызықтармен қосылады. Сызықтық кесте адам

қабылдауға ыңғайлы, өйткені онда қосымша ақпарат жоқ. Барлығы өте қарапайым — сызық жоғары — өсу тренд, құлдырау-төмендеу. Алайда, елеулі талдау үшін, онда көп қосымша ақпарат жетіспейді, сондықтан көптеген трейдерлер графиктердің басқа түрлерін пайдаланады.

Барлар. Барларды пайдалана отырып кестедегі бағаны графикалық ұсыну неғұрлым ақпараттық. Желілік барларға қарағанда, барлар қосымша ашу бағасын, берілген кезеңде бағаға жеткен максимум және минимумын көрсетеді (1.5-сурет).

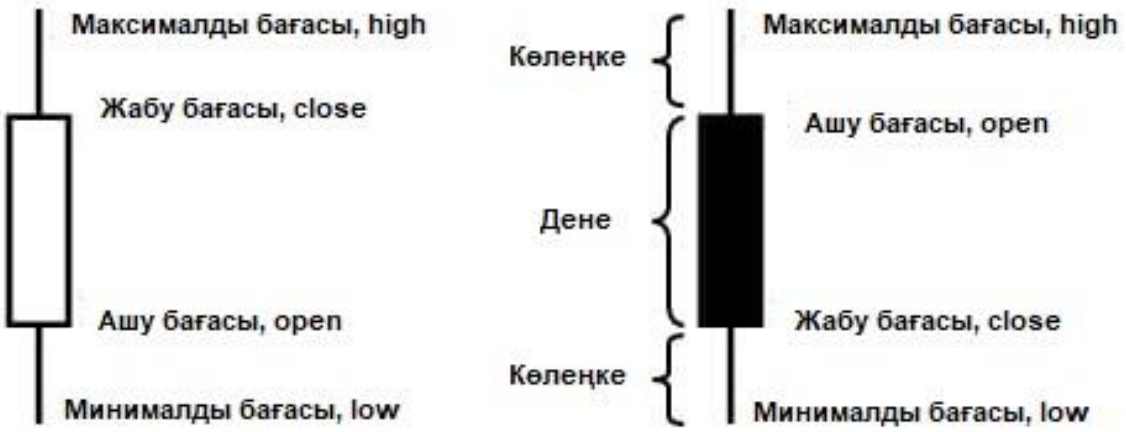
Бар оң және сол жақтағы көлденең кесінділері бар тік сызық болып табылады. Графиктерді көрсету интервалының қандай таңдалғанына байланысты, мұндай деректер бір барды көрсетеді. Егер күндізгі интервал таңдалса, онда бір бар — бір күн, апталық-1 апта және т. б. [19].

Сурет 1.5 – Барлар түріндегі кесте

Барлардың ұзындығы баға ауқымына сәйкес келеді. Жоғарғы және төменгі нүктелер - бұл котировкалардың ең жоғарғы және ең төменгі деңгейі. Оң жақ сызық - ашу бағасы, сол жақ-жабу. Егер сол сызық оңнан төмен болса, онда жабу бағасы ашу бағасынан жоғары және Біз өсуді байқаймыз. Керісінше, сол сызық оңнан жоғары болғанда, жабу бағасын төмен аламыз, ашылу бағасы және нарықта баға белгіленімнің құлдырауы.

Жапон шамдары. Жапон шамдары барлар сияқты, атап айтқанда, ашу және жабу бағасы, таңдалған кезеңдегі ең көп және ең аз баға белгілеулер сияқты мүлдем ұқсас ақпаратты көтереді. Барлар мен Шам арасындағы айырмашылық тек графикалық көрініс те (1.6).

Егер барлар көлденең сызық түрінде көрсетілсе, онда шамдардың ашылу және жабу бағасының арасындағы бөлігі қалыңдатылған болады. Оны шам денесі деп атайды. Ал шам денесінен максимумға дейін-шам көлеңкесі.



Сурет 1.6 - Жапон шамдары

Сурет 1.7 – Биткоин бағамының сызықтық және барлар біріккен кестесі

Бұдан басқа, өсуіне немесе құлауына байланысты жапон шамының денесі түрлі түстерге боялады. Классикалық нұсқада, ақ түс-өсу, қара-құлдырау. Басқа кең таралған нұсқа: қызыл дейді және Quotes төмендету жасыл және қызыл түстердің комбинациясы, өсуі туралы жасыл. Көптеген трейдерлер түстердің комбинацияларын пайдаланады, бірақ бұл әдеттер.

Түсті бояу графикті қабылдауды жақсартады. 1.7-суретте https://www.ino.com/ сайтынан алынған биткоинның 2019 жылдың 12 және 14 мамыр аралығындағы сызықтық және жапон шамдарының графиктері [19].

1.4 Зерттеудің өзектілігі мен жаңалығы

Бүгінгі күні, қазіргі заманғы инновациялар әлемінде Ақпараттық технологиялар қызметтің барлық салаларында маңызды рөл атқарады. Берілетін және алынатын ақпараттың деректер көлемі жылдам өсуді көрсетеді және деректерді беру жылдамдығын арттыру қажеттілігі проблемасы пайда болады. Осыған сәйкес, бұл жұмыста практикалық маңыздылығы, әзірленген қосымша ақпаратты алуда кідірістерді азайтып, деректердің үлкен көлемін жылдамдатуға мүмкіндік береді.

Еліміздің көптеген аудандарында ақпараттық технологиялардың таралуы өте жоғары. Қор биржасы осы салалардың бірі болып табылады. Қор нарығы қазіргі уақытта жоғары технологиялық сала болып табылатыны белгілі. Осы прогресстер негізінде бірнеше сауда платформалары, брокерлік жүйелер құрылады, олар жоғары жылдамдықты байланыс арналарының үлкен жүктемесін орындайды. Осыдан, ақпараттың едәуір көлемін беру үшін үлкен деректер компрессиясын қамтамасыз ететін қосымшалардың қажеттілігі қаржы жүйесі саласында ғана емес, сондай-ақ маркетинг, бизнес, телекоммуникация, сауда, логистика және мемлекеттік басқармаларда өте жоғары.

Статистикаға сәйкес, сақталатын және алынатын деректердің барлық көлемі жыл сайын өсіп келеді және деректердің жедел трансмиссиясы мәселесі өте маңызды және XXI ғасырда маңызды рөлге ие. Жұмыстың ғылыми жаңалығы декодталған және десериалданған деректерді деректер базасына сақтауға және оларды биржалық жапон шырақтары түрінде ұсынуға мүмкіндік беретін кем дегенде төмендетілген қаржы материалдарының үлкен көлемін жедел беру үшін әдісті әзірлеу болып табылады.

2 Simple Binary Encoding хаттамасы

2.1 CPU & Memory революциясының тарихы

Simple Binary Encoding (SBE) - бұл қаржы индустриясында қолданылатын өте жылдам сериализация кітапханасы [20].

Сериализацияның барлық мақсаты - хабарламаларды кодтау және декодтау, және XML хаттамасынан бастап қол жетімді опциялар көп, оларға JSON, Protobufer, Thrift, Avro және т. б. жатады.

XML, JSON – бұл мәтін негізіндегі кодтау және декодтау, бұл көп жағдайда жақсы, бірақ латенттілік маңызды болған кезде, онда мәтін негізіндегі кодтау және декодтау тар болады.

Protobuffer, Thrift, Avro екілік опциялар болып табылады және өте кең қолданылады [20].

Simple Binary Encoding екілік болып табылады және базалық жабдықты (процессордың кэші, алдын ала таңдау, қол жеткізу үлгісі, конвейер нұсқаулығы және т.б.) пайдалану үшін механикалық симпатия негізінде салынған [20].

Қазіргі заманғы технологиялар саласында 8-биттік, 16-биттік, 32-биттік, 64-биттік қуатты процессор пайда болды. Бүгінгі күні жады орындарының құны арзан болып жатқандықтан 512 ГБ мөлшерінде серверді сатып алу өте оңай [20].

Жүйенің көп бөлігі орындау кезінде оңтайландыруға сүйенеді, бірақ SBE толық шынымен тәсілді қолданады, және оңтайландырудың бірінші деңгейі компилятор орындайды.

Сурет 2.1 - Simple binary encoding сериализациясы

2.1 суретте көрсетілгендей бұл жердегі «схема» - xml-файл макеті мен хабар деректерінің түрін анықтауға арналған. Ал «компилятор» - кіріс деректер ретінде схеманы қабылдайды және аралық көрініс (Intermediate Representation немесе Аралық көрініс (IR) - бұл деректер құрылымы немесе бастапқы кодты ұсыну үшін компилятор немесе Виртуалды машина қолданатын код) жасайды. Бұл қабатта көптеген қызықты, мысалы, соңғы/тұрақты, оңтайландырылған кодты пайдаланулар бар. Келесі, хабарлама – ол буфер арқылы нақты хабар-қаптама болып табылады.Толық стек әдісі әр түрлі деңгейде оңтайландыруды жүргізуге мүмкіндік береді.

Қажетсіз кодтың болмауы төмен кідірісі бар жүйе үшін өте маңызды, егер бұл туралы қамқорлық болмаса, онда бағдарлама CPU кэштерін дұрыс пайдалана алмайды және GC-кідірісіне көшуі мүмкін.

SBE flyweight үлгісінің айналасында құрылады, барлық нәрсе JVM жады жүктемесін азайту үшін нысанды қайта пайдалану болып табылады.

Ол буфер ұғымы бар және қайта пайдалануға болады, кодер және декодер буферді кіріс ретінде қабылдай алады және онымен жұмыс істей алады. Кодер және декодер бөлінбейді немесе өте аз (яғни String жағдайында).

Ол GC суреттен толық шығару үшін direct/off буферін пайдалануды ұсынады, бұл буфер ағын деңгейінде бөлінуі мүмкін және хабарларды кодтау және кодтау үшін пайдаланылуы мүмкін.

Буферді пайдалану үшін код фрагменті:

```
final ByteBuffer byteBuffer = ByteBuffer.allocateDirect(4096);

final UnsafeBuffer directBuffer = new UnsafeBuffer(byteBuffer);


tradeEncoder        .tradeId(1)
        .customerId(999)
        .qty(100)
        .symbol("GOOG")
        .tradeType(TradeType.Buy);
```

2.1.1 Кэшті алдын ала таңдау

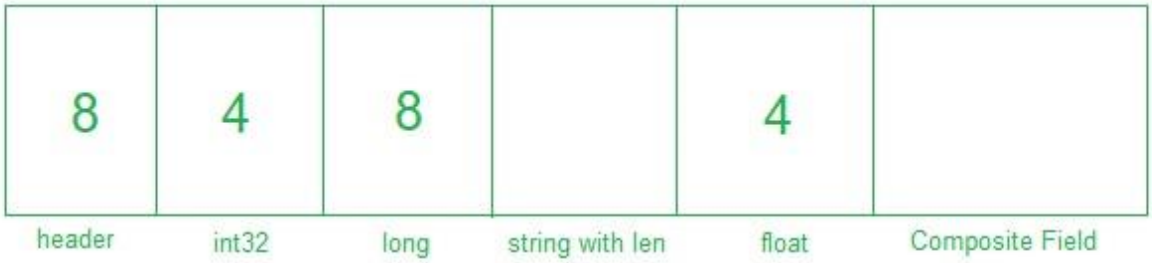
CPU орнатылған prefetcher аппараттық бар. Кэшті алдын ала таңдау - бұл нұсқаулық немесе деректерді бастапқы қоймадан баяу жадқа жылдам жергілікті жадқа шығару арқылы орындау өнімділігін арттыру үшін компьютерлік процессорлар қолданатын әдіс.

CPU жылдам кэшінен алынған деректерге негізгі жадқа қарағанда жылдамырақ ретпен кіру.

Егер алгоритм ағыны болса, алдын ала таңдау өте жақсы жұмыс істейді,ал пайдаланылатын деректер массив ретінде үздіксіз. Массивке кіру өте жылдам, себебі ол дәйекті және болжамды.

SBE массиві негізгі қойма ретінде пайдаланады және оған өрістер буып-түйілген.

Деректер кэш жолының шағын партияларымен жылжиды, ол әдетте 8 байтты құрайды, сондықтан қолданба 1 байтты сұраса, ол 8 байт деректер алады. Деректер массивке оралған болғандықтан, алдын ала таңдау массивінің бір беттік мазмұнына қол жеткізуді және бұл өңдеуді тездетеді [20] .



Сурет 2.2 - Байттардағы деректер түрлерінің өлшемдері

2.1.2 Ағынды қатынау

SBE барлық қарапайым типтерді қолдайды, сондай-ақ ауыспалы өлшемдегі пайдаланушы түрлерін анықтауға мүмкіндік береді, бұл ағындық және тізбекті кодтар мен декодер болуға мүмкіндік береді. Бұл кэш жолынан деректерді оқу жақсы артықшылығы бар және декодер хабар туралы өте аз метамәліметтерді білуі тиіс (яғни, Е ығысу және өлшем) [21].

Бұл, әсіресе, деректердің ауыспалы түрлері кодталса, компромиссиялық оқу тәртібі жинақтау тәртібіне негізделуі тиіс.

Мысалы, жазу үшін тапсырыс астында қолданылады

```
tradeEncoder        .tradeId(1)
        .customerId(999)
        .tradeType(TradeType.Buy)
        .qty(100)
        .symbol("GOOG")
        .exchange("NYSE");
```

Жол атрибуттары үшін (символ және алмасу) оқу тәртібі бірінші символ болуы тиіс, содан кейін қолданба тәртібін өзгертсе, ол дұрыс емес өрісті оқитын болады, ал басқа оқу ұзақтығы ауыспалы төлсипат үшін бір рет қана болуы тиіс, себебі бұл қатынаудың ағындық үлгісі [21].

2.1.3 Қауіпсіз API-интерфейсі

Массивке байланысты тексеру үстеме шығыстарды қоса алады, бірақ SBE қауіпті API пайдаланады, және бұл қосымша үстеме шығындарды тексеруі жоқ.

Компилятор кодты генерация жасағанда, ол алдын ала материалды есептеп, константаларды пайдаланады. Бұған бір мысал болатын жағдай - off set өрісі генерацияланған кодта есептелмейді [21].

Әрбір ядро параллель жұмыс үшін бірнеше порттар бар және бірнеше нұсқаулар бар. SBE компиляторы осы қымбат нұсқаулардан бос кодты жасайды және көрсеткіштері бар базалық математикаға ие.

Қымбат нұсқаулықтан тегін код өте жылдам және ядроның барлық порттарын пайдалануға болады [21].

Java сериализациясы үшін код үлгісі

```
public void writeFloat(float v) throws IOException {
    if (pos + 4 <= MAX_BLOCK_SIZE) {
        Bits.putFloat(buf, pos, v);        pos += 4;    } else {
        dout.writeFloat(v);    }
}

public void writeLong(long v) throws IOException {
    if (pos + 8 <= MAX_BLOCK_SIZE) {
        Bits.putLong(buf, pos, v);        pos += 8;    } else {
        dout.writeLong(v);    }
}

public void writeDouble(double v) throws IOException {
    if (pos + 8 <= MAX_BLOCK_SIZE) {
        Bits.putDouble(buf, pos, v);        pos += 8;    } else {
        dout.writeDouble(v);    }
}
```

Java сериализациясы үшін SBEда код үлгісі

```
public TradeEncoder customerId(final long value)
{
    buffer.putLong(offset + 8, value, java.nio.ByteOrder.LITTLE_ENDIAN);    return this;}

public TradeEncoder tradeId(final long value)
{
    buffer.putLong(offset + 0, value, java.nio.ByteOrder.LITTLE_ENDIAN);    return this;}
```

Әртүрлі кластағы хабарламалар өлшемдері: marshal.SerializableMarshal класс түрінің өлшемі - 267; marshal.ExternalizableMarshal класс түрінің өлшемі -75; marshal.SBEMarshall класс түрінің өлшемі - 49. SBE ең ықшам және өте жылдам, оның авторлары Google Protobuffer хаттамасына қарағанда шамамен 20-50 есе жылдам екенін айтқан.

2.2 Әдістерді және қолданыстағы деректерді жіберу хаттамаларын сараптау және демонстрациялық талдау

Ғылыми жұмыстың басты ерекшелігі - деректердің үлкен көлемін тез және ықшамды кодтау үшін SBE хаттамасын пайдалану. Қазіргі уақытта қаржы жүйесі әртүрлі форматтағы хабарламалардың көп санын жіберу және алу арқылы жұмыс істейді. Қаржы биржаларынан алынған нарықтық деректердің арналары секундына ондаған немесе жүздеген мың хабарларды ала алады, ал берілген деректер көлемі әр уақытта артып келеді. FIX, FAST мәнінің тәгі сияқты ASCII таңбаларын кодтауды пайдаланған нарық деректерін алу үшін біздің уақытымызда маңызды рөл атқарады. Дәл осылардың қатарына осы салалардағы кеңінен танымал XML және JSON протоколдарын қосуға болады.

Қазіргі заманғы электрондық қор биржаларында операцияларды тезірек аяқтау мүмкіндігімен жабдықталмаған, өйткені FIX веб-сайтына сәйкес, қор биржаларында сұхбаттасқан фирмалардың сатып алушы тарапынан 75%-ы және сатушылар тарапынан 80%-ы, 1992 жылы басталған электрондық қашықтан сауда-саттық жүргізу үшін Financial Information Exchange

хаттамасын пайдаланады. «FIX» ProtoBuf пен FAST сияқты қазіргі заманғы екілік хаттамалардың сериализциялану мен десериализациялану жылдамдығынан әлдеқайда төмен.

Бұрын FIX коммуникацияны автоматтандыру, қор нарығында қаржы деректерін жіберу үшін жасалған болатын. Бірақ FIX протоколы қаржылық деректердің жоғарылатылған көлемін жіберудің ең сенімді құралы болмады, сондықтан оның ары қарай дамуы ретінде жаңа барлық талаптарға сай стандарт – FAST (FIX Adapted for STreaming) хаттамасы құрастырылған болатын.

Сонымен қатар, осындай проблемаларға басқа да шешімдер бар. Олардың танымалы Protocol Buffer болып табылады. Protocol Buffers - XML-ге балама ретінде Google-дың авторлары ұсынған деректердің сипаттама тілі. Әзірлеушілер Protocol Buffer XML-ден қарағанда әлдеқайда ыңғайлы, функционалды және жылдамырақ екеніне сенімді.

Goldfish қорының және Инвестицияларды басқару жөніндегі басқармасының өкілі Шломи Марко өзінің жоғары жиілікті сауда-саттық өткізілетін қорда жұмыс істейтінін хабарлайды. Олар өнімділігіне қатысты тұрақтылықпен ұстанады - бәрі миллисекундке кешіктірілуге дейін есептеледі. Хабарламаны жіберу барысында олар сервердегі басты бағдарлама Java болғанда, Google Protocol Buffer-ді пайдаланады, ал клиент бөлігі көбінесе C#. Сол сияқты, Protobuf жұмыста қолданылған.

Дегенмен, соңғы жылдары FIX әзірлеушілері сілтеме жасауды - Simple Binary Encoding стандартын пайдаланатын жаңа FIX-ті әзірледі. Мартин Томсонның айтуынша, SBE жобалау принциптерін пайдаланудың түпкілікті нәтижесі өте төмен және болжамды кідіріс проблемасы бар Google Protocol Buffer (GPB) қарағанда 16-25 есе артық қуатты өткізу қабілеті бар кодектер. Бұл микро-тестілеуде және қосымшаны қолдануда байқалды.

Protobuf-пен салыстырғанда нарықтық деректердің қарапайым хабарламасы ~25 нс-та кодталған немесе декодталған болуы мүмкін, Protobuf-қа сол хабарламаны дәл сол құралдармен және жабдықтармен дешифрлеу үшін 1000 нс қажет болар еді. XML және FIX баяу және жылдамдығынан төменгі орын алады. Қазіргі уақытта ақпараттық-бағытталған бағдарламалар үшін кеңінен тараған деректерді жіберудің форматы – JSON пайдаланылады.

JavaScript Object Notation (JSON) – бұл деректермен алмасу форматы, қарапайым өңделеді және бағдарламалармен генерацияланады, қолданушыларға оңай келеді. Ол JavaScript тілі негізінде әзірленген. JSON – C, C++, C#, Java, JavaScript, Perl, Python және тағы басқалар сияқты бағдарламалау тілдерімен сәйкес пайдалану кезінде орындалатын тілге тәуелді емес мәтін форматы. Сондай-ақ, JSON форматы браузер мен сервер арасындағы деректер алмасу тапсырмаларындағы және JSON серверлерінің өздері арасында шағын деректер құрылымдарын сериализациялау үшін қолайлы. Осыған байланысты, осы жұмыста SBE және JSON деректерді

жіберудің әр түрлі әдістерінің арасында салыстырмалы талдау және тестілеу жүргізілді.

2.2.1 Жобалау реті

Ең төмен дисперсиясы бар аз дәрежелі тоқтау деңгейіне жету үшін, проектілеу және зерттеумен айналысатын, жобалау негіздері комплексін орнату маңызы зор. Егер компромиссті шешім негіздерін жобалау қажет болса, онда бәсекелес үміткерлер арасында іріктеу туралы шешім қабылдауға көмектесуге міндетті. Осы жүйелік қорларды пайдалану нәтижелері зерттеуге жатқызылуы мүмкін. SBE зерттеуінде үлкен маңызға ие алты негізгі жобалау негіздері бар. Біріншісі *еркін көшіру* деп аталады. Желілер мен сақтау жүйелері хабарларды шифрланған және шифрдан шығарылған буферлермен тікелей жұмыс істейді.

Тәуелсіз көшірудің ережесі – аралық буферлерді кодтау және декодтау үшін қолдануға болмайды. Аралық буферді пайдаланған жағдайда, шығындар байттарды бір реттен артық көшіруге байланысты көбейеді. SBE кодектері базалық тікелей буферге және буферден кодтау, декодтау мақсатында аспект (ұғым) береді. Бұл шектеулер трансфер буферлеріне қарағанда, үлкен хабарларға тікелей қолдау көрсетілмеуімен түсіндіріледі. Үлкен хабарламалардың жинағын тарату және орнату үшін берілетін аралық өлшемінен үлкен хабарларға арналған фрагментация протоколы қажет.

Келесі түрі *өз типтерін түрлендіру* деп аталады. Деректерді басты буферде өз типтері ретінде кодтау процедурасында еркін көшірмені құру айтарлықтай әсер етеді. Мысалы, 64-биттік бүтін сан жалпы ассемблер нұсқауының x86_64 MOV құрастыру нұсқасында негізгі буферде тікелей кодтала алады. Егер осы типтің байт процедурасы басқа процессор үшін анықталса, онда байт x86 BSWAP пәрмені көмегімен негізгі буферде сақталғанға дейін регистрда ауыстыруға болады. Бұл принципті қолдану өрістерге C ++ және Java секілді маңызды деңгейдегі тілдерде немесе класстар мен құрылымдық өрістер үшін рұқсаттама алуға мүмкіндік береді.

Келесі, проектілеу принципінің үшінші типі - *еркін таңдау*. Айта кету керек, объектілерді таңдау процессордың кэштегі ақпарат сыртқа жайылып кету себебіне негіз болып табылады және өнімділіктің төмендеуі деп санауға болады. Осылайша, заттарды жинау немесе жою дұрыс болады. SBE кодектерін еркін таңдауды жобалауда «flyweight» үлгісін қолданады. Тікелей шифрлау және деректерді декодтау үшін «Flyweight» терезесі негізгі буфердің жоғары жағында орналасқан. Сәйкес түрдегі «flyweight» үлгі идентификаторы бар хабар тақырыбының төменгі жағында таңдалады. Үлгі идентификаторы бар хабардың тақырыбының төменгі жағында тиісті үлгідегі «Flyweight» таңдалады. Деректер өрістері деректерді өңдеу шекарасынан тыс жерде сақтау қажет, бұл жағдайда олар бөлек сақталуы керек.

Төртінші қағида - *ағындық рұқсат*. Кіші жүйелердің инновациялық жады ең қиыны бола түсті. Қол жеткізу моделі әдісі жады тиімділігі мен кезектілікті айтарлықтай белгілеуге мүмкіндік береді. Файлға инкрементальді қол жеткізудің жадысына негізделген үздік тиімділік және

одан да көп дәйекті жасырындық ағындық әдісті қабылдау арқылы қол жеткізіледі.

Базистік буферде алға қарай ілгерілеуге негізделген SBE кодектері кодтау және декодтау деректері үшін беріледі. Кейінгі түрі *бірыңғай сөзге қол жеткізу* деп аталады. Егер сөздерге арналмаған сөздер шағын шектерде болса, көптеген процессорлық архитектурала өнімділігімен байланысты айтарлықтай қиындықтарды көрсетеді. Бұл жағдайда сөздердің бастапқы мекен-жайы бар, оның көлемі байтпен көбейтіледі. Тек 4-ке және т.б. бөлінетін байт адрестерінен бастап, 8 және 32-биттік бүтін сандарына бөлінетін, 64-биттік бүтін сандар тек байт мекенжайларынан шығуы керек. Бірақ SBE схемалары «жылжымалы» теориясын қолдайды, ол хабарламада бастапқы өрістің орнын орнатады. Деректер кадрлеу хаттамасында 8 байтты (boundary) шекарасына инкапсулирленген деп есептеледі. Ықшам және тиімді хабарламаларды орындау үшін өрістерді типі мен көлемін азайту бойынша ретімен сұрыптау керек. Және проектілеу принципінің түпкілікті типі - *кері үйлесімділік*.

Ірі компанияларда немесе компания арқылы бір мезгілде барлық жүйелерді әрқашан жаңартуға болмайды. Байланыс үшін, жұмыс қабілетін кеңейту үшін, хабардың форматтары кері үйлесімді болуы керек, бұл жағдайда сол хабардың соңғы нұсқасын оқуға және керісінше оқи алатын ескі жүйесі болуы тиіс. Ең жаңа жүйелерде қолдануға барлық мүмкіндіктері бар, бірақ ескі жүйе оларды жаңартқанға дейін елемейтін, хабарларға жаңа қосымша өрістерді енгізуге мүмкіндік беретін, 2.3 суретте SBE-де жасалған кеңейту жүйесі көрсетілген.

Жаңа бөлінбейтін өрістерді толтыру қажет болғанда немесе негізгі құрылымдық өзгеріс қажет болған жағдайларда, хабардың ең жаңа түрі қолданған дұрыс болуы керек, себебі ол көпшілік хабардың қолданыстағы түрінің мағыналық кеңеюі деп саналмайды. SBE хабарларында бірыңғай тақырып бар, ол жүйелі болу үшін хабар негізінің түрін және нұсқасын анықтайды. Тақырып түбірлік өрістердің (root fields) қолдауымен келеді, олар тұрақты жылжулары бар белгілі бір ұзындыққа ие.

Түбір өрістер С синтаксисіне өте ұқсас. Деректер ең күрделі болып саналса, бұл жағдайда (root block) түбірлік блокқа ұқсас бір немесе бірнеше циклдік топтардың (repeating group) басқаруға барлық мүмкіндіктері бар. Қайталанатын топтар басқа қайталанатын топтық схемаларды енгізудің әрбір мүмкіндігіне ие. Ақыр соңында, (variable length strings) жолдың айнымалы ұзындығы хабардың соңына келеді. Одан басқа, өрістер - міндетті емес болуы мүмкін.

Сурет 2.3 – SBE хабарлама құрылымы

2.3 Ғылыми жұмыстың дамуы және деректерді жинау

Деректерді жіберу хаттамаларын шолу негізінде, белгіленген мақсатқа қол жеткізу үшін бірнеше жағдайды белгілеуге болады:

1) дұрыс пайдаланбау тәуекелдерін азайту үшін құрылғымен жұмыс істеу дағдысы мен тәжірибесі қажет;

2) клиент-серверлік қосымшасын жобалау;

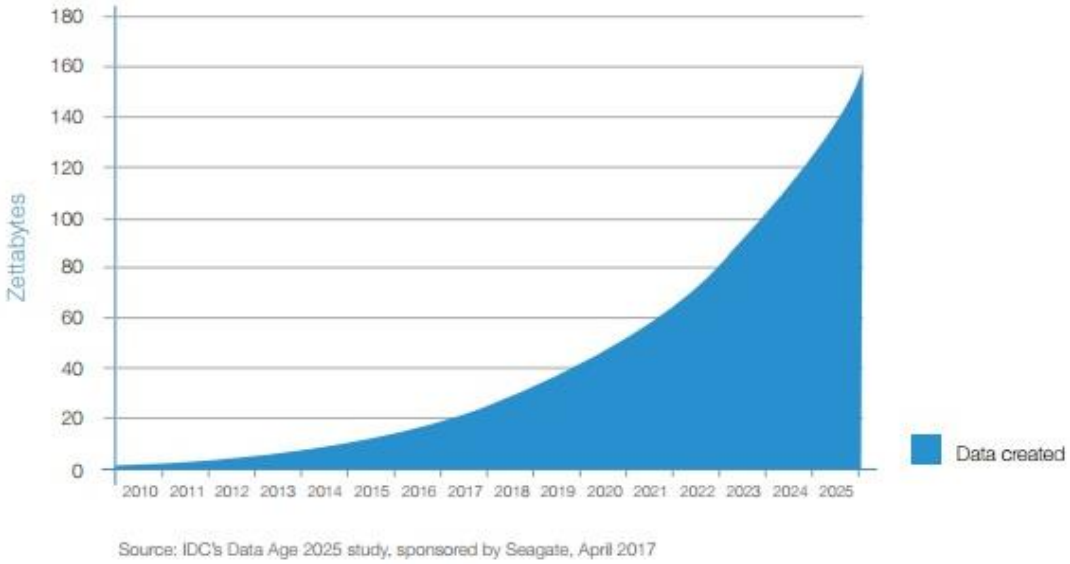
3) экономикалық деректердің үлкен көлемінде деректер базасына деректер енгізу қажет;

4) желі бойынша берілетін кодталған хабарламаларда деректерді жіберу әдісін жасау және зерделеу қажет, олар қабылдаушымен декодталады және деректер базасында сақталады;

5) экономикалық деректердің графикалық тұжырымдарын зерттеу қажет.

2.4 Simple binary encoding (SBE) деректерді жіберу хаттамасының соңғы қолдануларына талдау

Күн сайын жаңа технологиялардың көмегімен әр түрлі ақпаратты өңдеу процесіне тап боламыз. Деректерді алмасу бойынша барлық операциялар Big Data өсуіне әсер етеді, ол өткен жылға қарағанда 10% - ға өсті. «Біз күн сайын қанша деректер құрамыз? Таңғалдырарлық статистика» (How Much Data Do We Create Every Day? The Mind-Blowing Stats.) мақаласында «тек соңғы екі жылда әлемде деректердің 90 пайызы құрылды» айрықша баса айтылған [22].

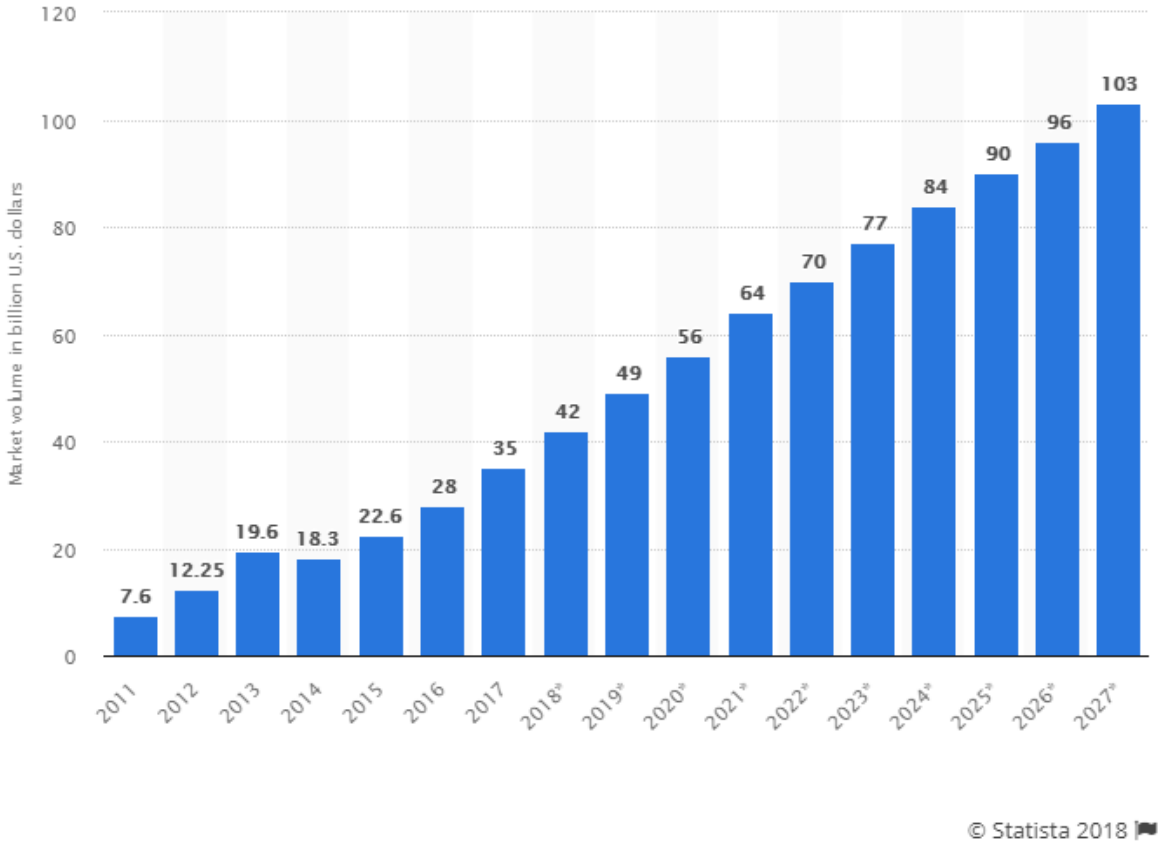


Сурет 2.4 - Жаһандық деректер базасының жылдық көлемі. Дерек-көзі: seagate.com

2017 жылдың сәуірінде Seagate веб-сайтында деректерге талдау жарияланған болатын, соған сәйкес, 2025 жылға қарай бүкіл әлем бойынша деректер мөлшері 160 зеттабайтқа дейін өсетін болады, бұл деректердің

басым бөлігі Интернет пайдаланушылар емес, бизнес тарапынан жасалатын болады. (Сурет 2.4) [23].

"Statista" порталында жарияланған статистика өзіне назар аудартады, онда 2011 жылдан бастап 2027 жылға дейінгі үлкен деректер туралы нарықтың өсу жорамалы келтірілген (Сурет 2.5). Порталдың мәліметтері бойынша, 2018 жылы жаһандық ірі деректер нарығындағы жылдық кіріс 42 миллиард АҚШ долларына жетеді деп күтілуде [24].



Сурет 2.5 - Big Data нарығының мөлшерінің болжамы (миллиардтаған АҚШ доллары). Дерек-көзі: statista.com

Негізінде қаржы жүйелерінің құрастырушылары болып деректермен алмасу барысында үлкен ақпараттың ағынымен жиі кездесетін сауда-саттық саласындағы жұмыс жасайтын ірі мекемелер болып табылады. Себебі, олар күнделікті жағдайда мәмілелердің баяу өңделу мәселесіне тап болады,

Қаржы әлемінде бұл жағдайды жоғары кідіріс деп аталды. Белгілі бір технологиялардың арқасында кәсіпорындардың көпшілігі желідегі мұндай кідіріспен күресуге тырысады және компьютерлік желілерде үздіксіз деректер алмасумен қамтамасыз етеді. Мұндай технологияларға деректерді жіберу әдістері мен хаттамалары жатады.

Хаттама – бұл компьютерлер сияқты электрондық құрылғылар арасында деректерді жіберуге арналған келісімдер, ережелер немесе рәсімдер жиынтығы. хаттамалары қаржылық қызмет субъектілері үшін алтын стандарт

болып табылады. FIX қаржы кеңістігінде сатып алушылар тарапынан да, сатушылар тарапынан да пайдаланылады.

FIX Құрама Штаттардан Еуропа мен Азияға таралады. Хаттамаларды халықаралық немесе салалық ұйымдар белгілейді. Олар брокерлер, банктер, инвестициялық пай қорлары, қор биржалары мен дилерлер бойынша түрленеді. FIX хаттамасы 25 жылдан астам уақыт бойы қолданылып келген және ақаулы қаржы сауда жүйесін толассыз ақпараттық ағын енгізу арқылы қалпына келтіруге көмектесті [20].

Деректерді өңдеу және жіберу кезінде Simple Binary Encoding хаттамасының басты ерекшелігі төмен латенттік болып табылады.

«Automated Trader Issue 44 Q1 2018» журналында «Simple Binary Encoding for high performance market data interfaces» (Нарықтық деректердің жоғары өнімді интерфейстеріне арналған қарапайым екілік кодтау) мақаласында жарияланған CME Group Чикаго Тауар биржалық ұйымындағы нарықтық деректердің интерфейстері Simple Binary Encoding көмегімен іске асырылғаны жөнінде айта кетілген [21].

SBE құралы қазіргі уақытта әлі де даму сатысында, бірақ оның артықшылықтарын көбісі бағалады. Бұл мақалада «Simple Binary Encoding (SBE) стандартын FIX ұйымы латенттілік қажеттілігін теңдестіру және өткізу жолақтарын пайдалану мақсатында әзірледі. FIX ұсыну хаттамасы бүгінде жылдам алмасу үшін жарамсыз болғандықтан, ол SBE хаттамасына ауыстырылды» [21].

Демек, SBE хаттамасында ішкі хабарларды да, сондай-ақ сауда әлемінде тарататын нарықтық деректерді де жеделдетуге шешім қабылданды. Мәселе жоғары сапалы саудада жатыр - микросекундтарда және тіпті наносекундтарда өлшенетін кідірістерді хабарлау үшін техникалық мәселе болып табылады.

SBE-мен бір деңгейде ұсынылатын жақсы шешім табу үшін балама табылмайтын FIX протоколының үш негізгі проблемасын белгілейміз. Бірінші мәселе: мыңдаған пайдаланушыларға таныс FIX бағдарламасының деңгейі өзгеріссіз қалады. Бұл талаптар SBE хаттамасының қолданыстағы FIX хабарламаларын білдіруі, сондай-ақ өнімділігін арттыру үшін оларды оңтайландыруға мүмкіндік беруі үшін қажет.

FIX-тегі екінші мәселе - хабарламаларды кодтау болашақ мақсаттарға сәйкес келмейді. Яғни, мәселе мынада жатыр, FIX-тің құрамында шамамен 20 деректер типі бар болса да, олардың барлығы адам оқи алатын ASCII мәтіні ретінде кодталады. Оны өзгерту үшін бұндай процесс белгілі бір уақыт қажет етеді, бұл өңдеу үшін бос уақыт шығыны. Және FIX хаттамасының соңғы проблемасы: кодтауда кейбір жол өрістері өзгертілген, яғни, жад макетінің жоғары өзгергіштігі және хабарламаның үлкен өлшемі, әдетте, CPU-ды тоқтата алатын кэште қателерді тудырады. Өзгергіштік, FPGA сияқты, аппараттық іске асырулар үшін кодтауды жарамсыз етеді [21].

Жоғарыда аталған мәселелердің барлығы SBE сияқты құралын құру және қолдану үшін басты себепкер болып табылады. SBE хаттамасын құру

кезеңінде өткеннен сабақтар алынды және олар төмен латенттілік үшін оңтайландырылды. Латенттілік - бұл пакетті бір нүктеден екінші нүктеге жіберу үшін қажет ететін уақыт [25].

Деректерді жедел жіберу протоколдары арасындағы ерекшеліктердің бірі - ол SBE сандық және басқа да деректерді сандық бағалауға берілетін сымдағы меншікті екілік типтерін қолданады. FIX-тің әрбір отызфазалы деректер типтері, SBE спецификасында анықталған, екілік кодтауға ие. SBE кодтау ASCII эквиваленттеріне қарағанда әлдеқайда ықшам [26].

CME Group деректері бойынша SBE-ні қолдану жетістіктері бекітілді. Олар:

- SBE өрістер мен типтер метадеректерде константалар ретінде орнатылуы мүмкін. Бұл үшін желілер арқылы «тұрақты мән» жіберуге міндетті емес, өйткені тұрақты мән жөнелтушіге де, алушыға да белгілі [26].

- Өрістер «Қосу/Өшіру» (немесе true/false жалаулары) сияқты тек мәндер массивін ұсынатын битетпен қолдауы мүмкін. Бит жиынтығы сымға 8-ден 64 битке дейінгі өлшемді (int) бүтін типі жіберіледі [26].

- Тұрақты жады. Яғни, кодтаушы немесе хабарлама декодері тек бір рет қана хабарлар буфері арқылы өтуі тиіс [26].

- Тағы бір жақсы тәжірибе – хабарламалар үстінен аралық буферде үлгілер түрінде кодтаушылар мен декодер жасау. Бір кодер нысаны және бір декодер нысаны әрбір хабарлама түрі үшін бар, олар қайталанып күні бойы қолданылады. SBE-мен іс жүзінде жад тұрақты жағдайға жақын болуы мүмкін [26].

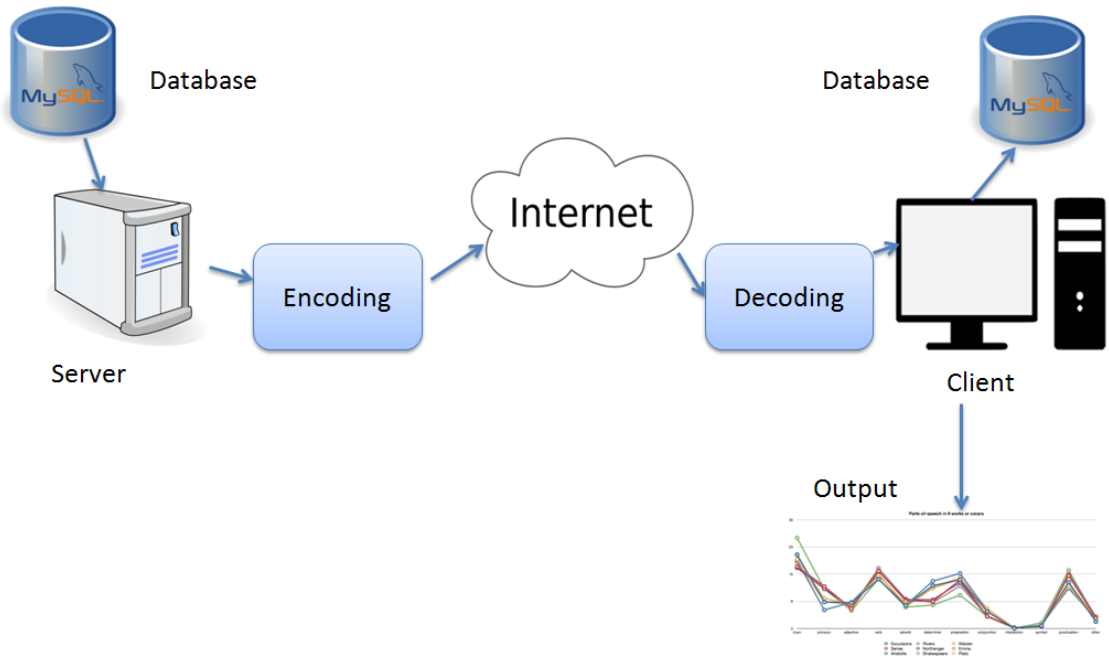
Globex электрондық сауда интерфейстерінің талаптарына сәйкес келетін SBE негізгі артықшылықтары:

- Жоғары өнімді кодтау/декодтау;
- Өткізу жолағын пайдалану дұрыс шешім;
- Стандартталған өрістер типі;
- Стандартталып хабарламаны анықтау;
- Хабарламалардың икемді құрылымы;
- Деректерге тікелей қол жеткізу;
- FIX хаттамасымен толық интеграция [26].

3 Бағдарламаны құру

3.1 Жүйенің архитектурасы

Көптеген клиенттер (қашықтағы процессорлар) орталықтандырылған серверден (хост-компьютерден) қызметтерді сұратады және алады. Сондай-ақ, портты ақпарат беруден босату үшін сокетті ашып, жабу мүмкін. Осыған байланысты, клиенттік компьютерлер компьютер пайдаланушыға сервер қызметін сұрауға және серверді қайтару нәтижелерін көрсетуге мүмкіндік беретін интерфейсті қамтамасыз етеді. Бұл процесс клиент-серверлік архитектураны сипаттайды. Демек, деректерді жедел жіберу жобасының архитектурасы салынды (Сурет 3.1).

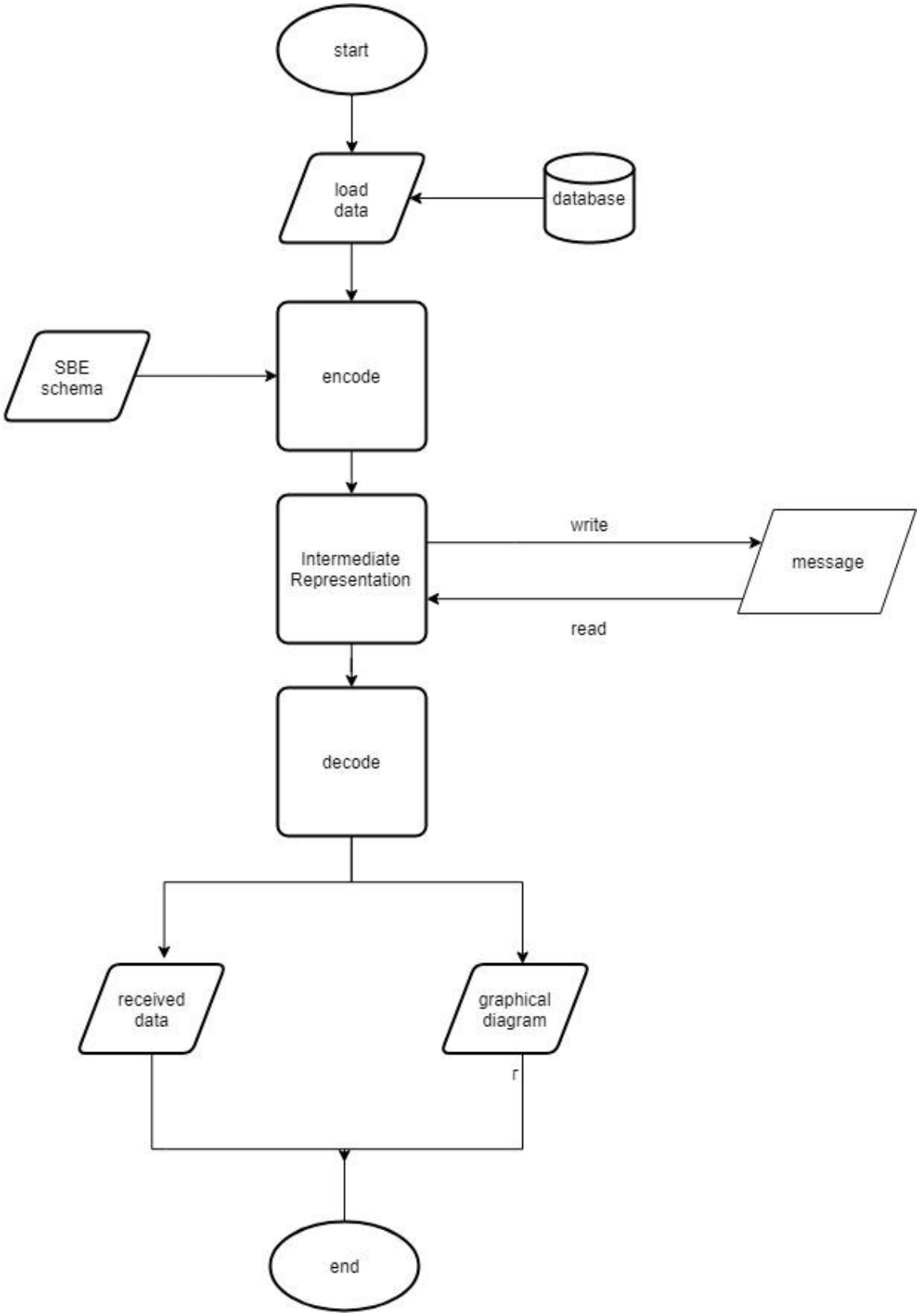


Сурет 3.1 - Жоба архитектурасы

Негізгі процесс клиенттен қосылуларды растауды күтетін серверден басталады. Оң жауап алу кезінде сервер деректер базасынан қаржылық деректерді алады, Simple Binary Encoding көмегімен кодтайды және клиентке жібереді. Ақпараттың үлкен көлемін алатын Клиент оларды декодтайды және деректер базасына сақтайды. Сонымен қатар, қаржылық деректердің графикалық қорытындысын ұсынылады.

Бағдарламалық өнімді іске асыру үшін бағдарламалау тілі - Java таңдалды. Java жоғары деңгейлі объектілі-бағытталған бағдарламалау тілі болып табылады, сондай-ақ бағдарламалық платформа болып табылады. Бұл бағдарламалау тілі арнайы құралдар мен кластардың арқасында желілік бағдарламалауды қарапайым етеді. Клиент-серверлік қосымшалар арқылы деректерді жіберу TCP/IP қолданбалы хаттамаларын пайдалануға негізделген. Сақтау үшін MySQL ДҚБЖ таңдалған, бірнеше бір мезгілде сұраныстарды қолдау, көп шоғырлану, және тіркелген және айнымалы

ұзындықты жазу кіреді бірқатар артықшылықтары бар. Бағдарламалық қамтудың жұмыс жасау блок-схемасы 3.2-суретте салынды.



Сурет 3.2 - Бағдарламалық қамтудың жұмыс жасау блок-схемасы

3.2 Сервер жұмысы

Серверді клиентпен қосу үшін жоғары өнімді енгізу-шығару операцияларын жүзеге асыруға арналған Java New I/O кітапханасы таңдалды. Сонымен қатар, серверлер мен клиенттер протоколы сияқты желілік қосымшаларды жылдам және оңай әзірлеуге мүмкіндік береді. Бұл TCP және UDP сокет серверлері сияқты желілік бағдарламалауды жеңілдетеді және жеделдетеді. Бұл кітапхананы зерттеуде Java I/O-мен салыстырғанда бірқатар артықшылықтар анықталды (3.1 - кестені қараңыз).

3.1 кесте - Кітапханалардың негізгі айырмашылықтары

Java IO	Java NIO
Ағын-бағытталған	буфер-бағытталған
Блоктаушы IO (енгізу-шығару)	Бұғатталмаған IO
	Селекторлар

Java NIO буфер-бағдарлы тәсіл деректер кейіннен өңделетін буферде оқылуымен ерекшеленеді . Буферде алға және артқа қарай жылжуға болады, бізге қаншалықты қажет. Бұл өңдеу кезінде біраз икемділік береді. Дегенмен, егер буферде барлық деректер болса, оны толығымен өңдеу керек. Үлкен деректерді буферде оқығанда, буферде деректер өңделмегеніне көз жеткізіңіз. Java NIO бұғаттамайтын режимі ағындарға арнадан оқу үшін деректерді сұратуға және тек қазіргі уақытта деректер болмаса, қол жетімді немесе мүлдем ештеңе алуға мүмкіндік береді. Сонымен қатар, NIO кітапханасын пайдалану арқылы өткізу қабілеті жақсарады, кідіріс деңгейі төмен және ресурстарды аз тұтынуды қамтамасыз етеді. 3.3-суретте серверлік бөлікте open() әдісінің сипаттамасы көрсетілген.

NIO негізгі компоненттері: selector және selectionKey, арналар,буферлер. Selector-Selector сыныбының объектісі. Selector әрбір данасы сокет арналарының көп санын бақылай алады, және осылайша, көп қосылыстар. Бір қызығы арнада болған кезде (мысалы, клиент байланысуға тырысады немесе оқу және жазу операциялары болады), Selector сұранысты өңдеу үшін қолданбаны хабардар етеді.

Селектор бұл SelectionKey класының даналары болып табылатын кілттерді жасау арқылы жасайды [27]. Әрбір кілт сұрау салу және осы сұрау түрі туралы ақпаратты қамтиды. Тип келесілердің бірі болуы мүмкін: қосылу әрекеті (клиент тарапынан), қосылу қабылдауы (сервер), оқу операциясы, жазу операциясы . А қосымшасында SelectionKey қолдану және әрбір типтегі процесс шарты жазылған.

Socket channel-бұл деректерді желі арқылы жіберуге мүмкіндік беретін жаңа socketchannel класының даналары. open () әдісі socketchannel данасын жасайды configureBlocking (false) арнаны бұғатталмаған ретінде орнатады . Серверге қосылу байланыс әдісі арқылы жүргізіледі. Сонымен қатар, сервердің IP-мекен-жайы және байланыс порты көрсетіледі. Selector жасау

үшін open() selector класын шақыруға болады. Ақырында, register әдісі сокет арнасының селекторын байланыстырады.

OP_ACCEPT пайдалану селектор тек клиент сервермен байланыс орнатуға тырысатыны туралы хабарлайды. Басқа ықтимал опциялар: клиент пайдаланатын OP_CONNECT, OP_READ және OP_WRITE.

```
public void open(int port){
    if(DEBUG)System.out.println("araylim.server.Server.open()");
    if(selector != null) return;
    if(serverSocketChannel != null) return;

    try{
        Class.forName("com.mysql.jdbc.Driver");
        connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/forex","root","root");
        statement = connection.createStatement();



        selector = Selector.open();
        serverSocketChannel = ServerSocketChannel.open();
        serverSocketChannel.configureBlocking(false);
        serverSocketChannel.socket().bind(new InetSocketAddress(port));

        serverSocketChannel.register(selector, SelectionKey.OP_ACCEPT);

        Platform.runLater(() -> {
            logs.add("server started");
        });
    }catch(Exception e){
        Platform.runLater(() -> {
            logs.add("error : couldn't start server");
        });
    }
}
```

Сурет 3.3 - клиентпен қосылу

Айта кету керек, selector клиент байланыс орнатуға тырысады оқиға күтеді. Бұл уақытта сервер қолданбалары selector және әрбір кілт үшін жасалған кілттерді алады, ол оның түрін тексереді. Кілттің өңделгенін байқай аласыз, ол remove әдісін шақыра отырып, кілттер жиынтығынан жойылуы керек. Егер кілт түрі acceptable (isAcceptable()=true) болса, сервер клиент арнасын accept әдісін шақыру арқылы тауып, оны бұғатталмаған ретінде орнатады, осылайша OP_READ параметрін пайдалана отырып, оны selector-ге жазады. Сондай-ақ, OP_WRITE немесе OP_READ | OP_WRITE опцияларын пайдалануға болады.

3.3 SBE деректер құрылымы

Деректер базасын толтыру үшін қаржылық деректерді зерделеу жүргізілді. SQL-сұраныстарды орындау үшін Java тілінде жазылған көптеген кластар мен интерфейстерден тұратын қолданбалы бағдарламалық интерфейс болып табылатын JDBC driver коннекторы пайдаланылды. Деректер қоры Тарихи Forex деректерімен толтырылған [27]. Forex-бұл валюта айырбастау жүргізілетін халықаралық қаржы нарығы. Базада 6 атрибуттар бар: open, high, low, close, date және time. OHCL сессияны ашу, жабу кезінде баға болып табылады , сессия кезінде майлы және ең төменгі баға, сондай-ақ M1

таймфрейм сипаттайды процесс күні мен уақыты(бір минут) сессия. Базадағы деректер саны 5049687 құрайды. Құрылым күрделі емес, алайда көптеген қаржылық деректер бар. Деректер өте жоғары жылдамдықпен кестедегі мәтіндік файлдан жолды оқитын LOAD DATA INFILE командасының көмегімен толтырылды.

3.4 SBE сұлбасының сипаттамасы

Simple Binary Encoding (SBE) – бұл схемаларға негізделген екілік форматқа хабарларды кодтау және декодтау әдісі. Хабарламалар FIX / SBE хаттамаларына негізделген жүйелер арасында деректер алмасу үшін ақпарат бірлігі болып табылады. SBE хабарламаларды, тақырыптарды және басқа да элементтерді анықтау үшін XML форматын пайдаланады. [28].

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<sbe:messageSchema xmlns:sbe="http://www.fixprotocol.org/ns/simple/RC3"
                   package="araylim.sbe"
                   id="1"
                   version="0"
                   semanticVersion="5.2"
                   description="schema"
                   byteOrder="littleEndian">
    <types>
        <composite name="messageHeader" description="Message identifiers and length of message root">
            <type name="blockLength" primitiveType="uint16"/>
            <type name="templateId" primitiveType="uint16"/>
            <type name="schemaId" primitiveType="uint16"/>
            <type name="version" primitiveType="uint16"/>
        </composite>
    </types>

    <sbe:message name="forexData" id="1" description="description of forex data">
        <field name="datetime"  id="1" type="uint64"/>
        <field name="open"      id="2" type="float"/>
        <field name="high"      id="3" type="float"/>
        <field name="low"       id="4" type="float"/>
        <field name="close"     id="5" type="float"/>
    </sbe:message>
</sbe:messageSchema>
```

Сурет 3.4 - SBE схемасының сипаттамасы

Жүйе теріледі және пайдаланушылар жаңа түрлерін жасай алады. 3.4-суретте қаржылық деректер үшін schema хабарламасы бейнеленген.

XML пішіміндегі түпкі құжат messageSchema элементі болып табылады. Бұл элементтің келесі төлсипаттары бар, мысалы: package, ID, version, semanticVersion, description және byteOrder [29].

Package - C ++ үшін аттар кеңістігі және Java үшін пакеттер атауы болып табылады. ID - сұлба үшін бірегей идентификатор. Version - тіркелген хабарламалар мен түрлердің нұсқасының нөмірі, әдепкі бойынша ол нөлге тең. SemanticVersion - жол түрінде семантикалық нұсқа ақпараты. Description - барлық хабарлама сұлбаасының сипаттамасы. ByteOrder - хабарламалардың барлық түрлеріне әдепкі тапсырыс (bigEndian немесе LittleEndian) бойынша байт саны [30].

SBE үшін хабарламаларды кодтау және декодтау негізгі қарапайым типтер жиынтығына бағытталған. Ол мыналарды қамтиды: int8 (8-биттік кодтау), int16 (16-биттік кодтау), int64 (64-биттік кодтау) және т.б.

Кодталған типтердің құрылымы Composite болып табылады. Composite - жариялау реті бойынша кодтау. Кейбір Composite элементтер хабарламаның үстіңгі деректемесін жариялау (декларациясы) үшін, сондай-ақ айнымалы жол ұзындығымен кодтау және топтық өлшемді кодтау үшін пайдаланылады. <Composite> элементінің атрибуттары бар: name, description.

Қайталанатын топтар, сондай-ақ айнымалы ұзындықтағы деректер message элементінің өрісінде болады. <Field>, <group> және <data> тәртібі әрбір хабарламаға деректерді қояды. Толық <Field> мәлімдемесі <group> және <data> мәнінен асуы керек. Бұл реттеме барлық тіркелген ұзындық өрісі хабарламаның алдында тұруын қамтамасыз етеді. Қайталанатын топтардан кейін ауыспалы ұзындығы бар деректер сақталады. <Message> элементінің атрибуттары бар: name, description, ID.

Элементтің өрісі (Field) хабардағы өрістің белгіленген өлшемін көрсетеді. Әр атрибуттың аты <name> элементінде жазылған. Бұл хабарламаларда олардың бесеуі бар: datetime, open, high, low, close. Әрбірі хабарламаның идентификаторы (id) және деректер типін (type) сипаттайды. DateTime үшін uin64 таңдалды (64-биттік белгісіз бүтін сан). Ал қалғандары үшін float type түрге ие [31].

SBE tool құралы - кодектерді жасау және хабар үлгісін тексеру үшін пайдаланылатын командалық жол утилитасы. Құрал Java тәләнде жазылған және орындалатын JAR-файл ретінде жұмыс істей алады [31]. Хабар схемасы SBE tool құралына енгізілуі мүмкін және әртүрлі тілдегі дереккөздерге бітеуіш жасау үшін құрастырылған, біздің жағдайда Java бағдарламалау тілі және оны келесі командамен қамтамасыз етеді:

java [- Doption=value] -jar sbe.jar <message-declarations-file.xml>

Хабарламалар framing хаттама ішінде жіберіледі деп болжанады. Рамка хабарламаның үстіңгі деректемесі мен хабарламаның өзі бар буфердің өлшемін анықтайды (3.5-суретті қараңыз).

```
+-------------------------------------------------------------+
|        +----------+--------------------------------+        |
|Frame   |Msg Header|Message Body                    |        |
|        +----------+--------------------------------+        |
+-------------------------------------------------------------+
```

Сурет 3.5 - Хабарлама рамкасы

OSI үлгісінің әртүрлі деңгейлеріне жататын және OSI моделінің алтыншы деңгейіне қатысты хабарлардың кодектерінен шығып кететін, рамкада сеанс немесе тасымалдау деңгейінің өрістері болуы мүмкін.

3.5 Деректерді кодтау және декодтау

SBE протоколында схема хабарламаларын енгізу ретінде қабылдайтын құрал бар, содан кейін ол бітеуіштерге (stubs) сәйкес тілді генерациялайды. Бітеуіштер хабарламаларды қаржылық деректер берілетін буферден кодтау және декодтау үшін пайдаланылады.

Протоколды іске асыру басталғанда біз жасалып жатқан бітеуіш файлдарды аламыз және оларды клиент-сервер бағдарламасында қолданамыз. Сервер клиентпен қосылуды күтеді, егер клиент қосылымды растаса, деректерді дерекқордан алады, оны SBE көмегімен кодтайды және оны клиентке жібереді. Хабарламаны кодтау процесі үшін алдымен тақырыпты кодтау қажет, содан кейін дерекқордан қаржы деректерін алу процесі орындалады.

```
while(rs.next()){

    ByteBuffer byteBuffer = ByteBuffer.allocateDirect(64);
    UnsafeBuffer directBuffer = new UnsafeBuffer(byteBuffer);
    int bufferOffset = 0;

    MESSAGE_HEADER_ENCODER
        .wrap(directBuffer, bufferOffset)
        .blockLength(FOREX_DATA_ENCODER.sbeBlockLength())
        .templateId(FOREX_DATA_ENCODER.sbeTemplateId())
        .schemaId(FOREX_DATA_ENCODER.sbeSchemaId())
        .version(FOREX_DATA_ENCODER.sbeSchemaVersion());

    bufferOffset += MESSAGE_HEADER_ENCODER.encodedLength();

    int id  = rs.getInt("id");
    Date date = rs.getDate("date");
    Time time = rs.getTime("time");
    float open = rs.getFloat("open");
    float high = rs.getFloat("high");
    float low = rs.getFloat("low");
    float close = rs.getFloat("close");
    int volume = rs.getInt("volume");

    LocalDate datePart = new LocalDate(date);
    LocalTime timePart = new LocalTime(time);
    DateTime dateTime = datePart.toDateTime(timePart);

    long datetime = dateTime.getMillis();
```

3.6 сурет - Хабарлама тақырыбын кодтау

Хабарлама тақырыбы бірнеше өрістерді қамтиды. Онда декодтар қай кодек хабарлама үшін үлгі ретінде болуын анықтайды. TemplateId - хабарлама үлгісінің түрінің идентификаторы болып табылады. Block length (блоктың ұзындығы) - қайталанатын топтарды немесе деректер айнымалысы алдында блоктың түбірлік хабарламалардың ұзындығы. Version - схема нұсқасы, схема үшін жарамды кеңейтімдер туралы хабарлайды (3.6-сурет).

Содан кейін, біз хабарламаның негізін кодтаймыз (3.7-суретті қараңыз):

```
FOREX_DATA_ENCODER
        .wrap(directBuffer, bufferOffset)
        .id(id)
        .datetime(datetime)
        .open(open)
        .high(high)
        .low(low)
        .close(close)
        .volume(volume);


ByteBuffer source = directBuffer.byteBuffer();

dataTracking.put(channel, source);
key.interestOps(SelectionKey.OP_WRITE);
currentRow++;
```

Сурет 3.7 - Хабарламаның негізін кодтау

SBETool.java кодтау буферін аудару үшін Java интерфейсін қамтиды. Сондай-ақ, SBE, DirectBuffer класын пайдалана отырып, байт буферімен жұмыс істеп, Java-ға арналған абстракцияны ұсынады [31]. Бітеуіш буферді жылжумен орайды және оны дәйекті түрде басынан бастап оқиды. Деректер базасынан деректерді алғаннан кейін, алдымен тақырыптың аты кодталуы керек, содан кейін хабарлама.

Растаудан кейін клиент буфердегі деректерді алады, қаржы деректерін кодтайды және оны дерекқорда сақтайды. Декодтау процесінде тақырыпты декодтау қажет, содан кейін хабардың негізін декодтау үшін пайдаланылуы мүмкін үлгі ізделеді. Оған қоса, сақталған деректер JavaFX кітапханасы арқылы шам диаграммасында және сызықтық диаграммада ұсынылуы мүмкін.

SbeTool көмегімен декодтау процесі:

- Бастапқыда біз хабарламаның тақырыбын декодтаймыз, содан кейін дерекқорда қаржы деректерін және негізін сақтауға болады (3.8 және 3.9-суреттерді қараңыз).

```
UnsafeBuffer directBuffer = new UnsafeBuffer(readBuffer);

int bufferOffset = 0;
MESSAGE_HEADER_DECODER.wrap(directBuffer, bufferOffset);
int templateId = MESSAGE_HEADER_DECODER.templateId();
int actingBlockLength = MESSAGE_HEADER_DECODER.blockLength();
int schemaId = MESSAGE_HEADER_DECODER.schemaId();
int actingVersion = MESSAGE_HEADER_DECODER.version();

bufferOffset += MESSAGE_HEADER_DECODER.encodedLength();
```

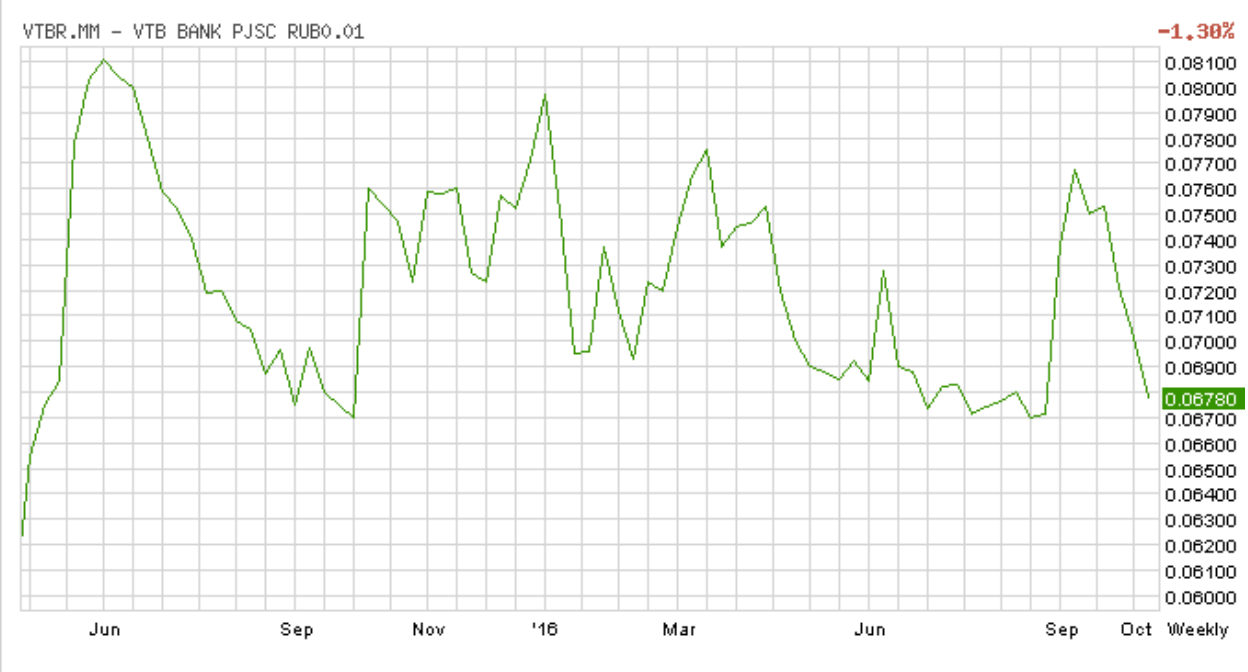
Сурет 3.8 – Хабарлама тақырыбын декодтау

содан кейін қаржылық деректерді деректер базасына сақтаймыз (сурет 3.11 және 3.12).

```
StringBuilder sb = new StringBuilder();
FOREX_DATA_DECODER.wrap(directBuffer, bufferOffset, actingBlockLength, actingVersion);

try{


    Date date = new Date(FOREX_DATA_DECODER.datetime());
    Time time = new Time(FOREX_DATA_DECODER.datetime());

    preparedStatement.setLong(1,FOREX_DATA_DECODER.id());
    preparedStatement.setDate(2,date);
    preparedStatement.setTime(3,time);
    preparedStatement.setFloat(4,FOREX_DATA_DECODER.open());
    preparedStatement.setFloat(5,FOREX_DATA_DECODER.high());
    preparedStatement.setFloat(6,FOREX_DATA_DECODER.low());
    preparedStatement.setFloat(7,FOREX_DATA_DECODER.close());
    preparedStatement.setLong(8,FOREX_DATA_DECODER.volume());

    preparedStatement.execute();
}catch(Exception e){
```

Сурет 3.9 – Хабарламаның негізін декодтау және дерекқорға сақтау

Деректер базасына деректерді сақтау аяқталғаннан кейін, клиент қаржылық деректердің графикалық қорытындысын ұсынады. Деректерді шығару екіге бөлінеді: сызықтық кесте түрінде және жапон шырақтары түрінде. Іске асыру үшін JavaFX кітапханасы пайдаланылды. Ол графикалық және медиа-пакеттер жиынтығы болып табылады, бұл әзірлеушілерге түрлі платформалар үшін дәйекті жұмыс істейтін бай клиенттік қосымшаларды жобалауға, жасауға, тестілеуге, ретке келтіруге және өрістетуге мүмкіндік береді. 3.10-суретте қаржылық деректердің сызықтық графигі бейнеленген:



Сурет 3.10 - Сызықтық графиктерді шығару

Х осі бойынша уақыт, у осі бойынша жабу кезіндегі баға болады. Одан әрі 3.11-суретте жапон шырағы түріндегі графика бейнеленген, онда барлық төрт 4 атрибутты (ашу, максимум, минимум, жабылу):

Сурет 3.11 - Жапон шамдары түрінде шығару

Шамның кең денесі белгілі бір уақыт кезеңінде ашу және жабу бағасын білдіреді, ал фитильдер - Шамның жоғарғы және төменгі сызығы - осы кезең үшін ең жоғарғы және ең төменгі баға. Жапон шамдары нақты уақыт кезеңі үшін нақты деректерді көрсете алады.

3.6 Биржа мен ресми сайтқа шолу

Bitstamp - банктік карталар сияқты дәстүрлі төлем әдістері арқылы ақша әкелуі мүмкін бірнеше биржалардың бірі. Осылайша, битокиндерді сатып алу үшін Bitstamp пайдалануға болады, содан кейін өз қалауы бойынша басқара алады.

Биржаны Bitstamp Ltd компаниясы иеленеді. Оның кеңселері Люксембург, Лондон және Нью-Йоркте орналасқан. Bitstamp - Еуропадағы алғашқы толық лицензияланған крипто-валюта биржасы. Ол қаржы саласындағы Люксембург қадағалау комиссиясымен (CSSF) реттеледі. Бұл факт басқа биржалардың аясында Bitstamp-ті тиімді бөледі, олардың көпшілігі реттелмейді, бұл белгілі бір тәуекелдерді тудырады. CSSF лицензиясы компанияның жауапты қызметтер алдында мезгіл-мезгіл есеп беретіндігіне кепілдік береді. Bitstamp "төлем органы" ретінде тіркелген, сондықтан әрдайым биржадан ақша жіберіп, алуға болады.

Bitstamp 2015 жылдың басында интернет шабуылдан кейін бұзылды. Осыдан кейін компания өзінің сауда платформасын толығымен қайта өңдеді. Олар клиенттердің қаражатының 98% суық сақтау орындарында сақтайтынын және тек 2% биржада тікелей қолжетімді екенін мәлімдейді. Бұл қауіпсіздік жүйесін айтарлықтай жақсартты.

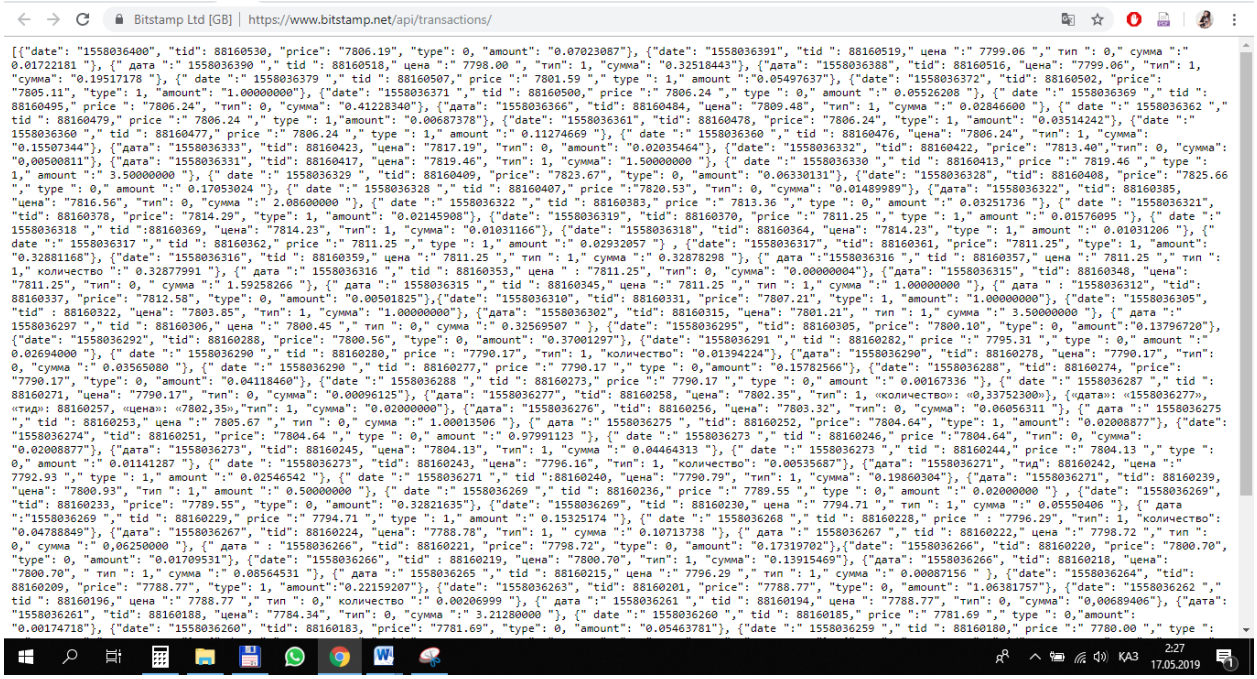
Шілде айында 2017 жылы Bitstamp Swissquote-мен келісім жасады. Швейцариялық банк пен брокер өз клиенттеріне биктоиндермен сауда жасауға мүмкіндік беру үшін биржаға сенім білдіруді шешті. Бұл көп Bitstamp сенімділігі туралы айтады. Банк Bitcoin сауда жүйесін MetaTrader4 (MT4) платформасымен біріктірген жоқ. Барлық сауда-саттық қызметі шектеулі банктің жеке платформасында жүзеге асырылады.

Bitstamp пайдаланушылар пікірлерінің аралас сипаты бар. Көптеген адамдар сервиспен қанағаттанады, бірақ қолдау қызметінің жұмысы қанағаттандырмайтын адамдар да бар. Poloniex сияқты, бәлкім, Bitstamp пайдаланушылардың жүзуіне дайын емес.

Bitstamp EUR және USD қарсы крипто-валюта сатуға мүмкіндік береді. Өкінішке орай, платформа көптеген басқа биржалар сияқты крипто-валюта көп емес. Bitstamp биткоиннен басқа, сіз Litecoin және Ripple жұптарын таба аласыз. Егер сіз басқа коиндер қызықтырса, сіз Bitstamp-ны осы үш крипто-валюталардың біріне айырбастау үшін пайдалана аласыз, содан кейін оларды қызықтыратын крипто-валюталарды қолдайтын биржаға аудара аласыз.

Bitstamp маржиналды сауданы қолдамайды.

Bitstamp-ға Фиатты айырбастау үшін комиссиясы 0,25%-ға жетеді. Көптеген басқа биржалар сияқты, мәмілелер көлемі неғұрлым көп болса, комиссия соғұрлым төмен болады. Bitstamp маркетмейкерлер (басқа трейдерлердің позицияларын агрессивті сатып алудың орнына, биржада ордерді орналастыратын адамдар) үшін жеңілдіктер ұсынбағандықтан ерекшеленеді. Көптеген крипто-валюта биржалары алаңда өтімділікті қамтамасыз ететін трейдерлерге жеңілдіктер береді. Бұл ретте алымдар көлемі орташа нарықтық жағдайға толық сәйкес келеді. Келесі 3.12-суретте Bitstamp биржасының деректері сақталынған орны https://www.bitstamp.net/api/transactions/ сілтемесі келтірілген.



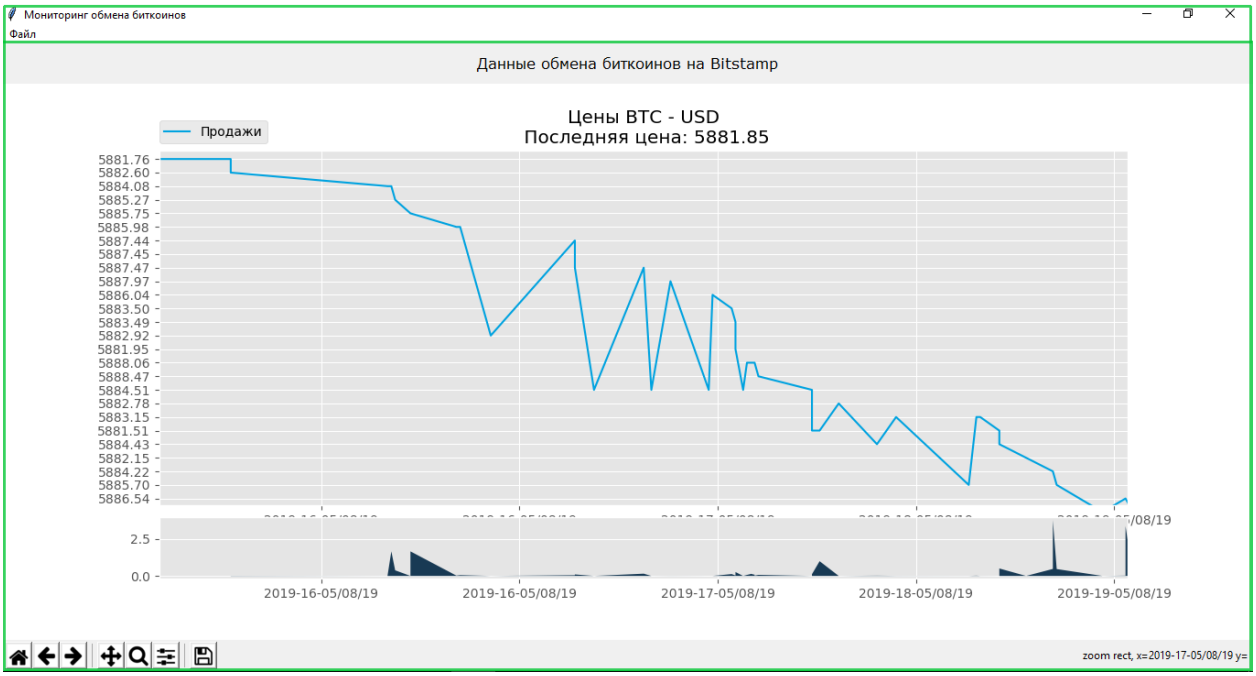
Сурет 3.12 – Bitstamp деректері

Bitstamp платформасына кіру веб-интерфейс арқылы беріледі. Ол үндеуде ыңғайлы, графиктерді құрастыруға баса назар аударылады. Бұл тұрғыда Bitstamp cTrader платформасына ұқсайды, бірақ мұндай бай функционалдығы жоқ. Платформада көптеген техникалық индикаторлар мен құралдар іске асырылды. Осылайша, осы параметр бойынша Bitstamp көптеген бәсекелестерін айналып өтеді.

Bitstamp банктік аударымдар, Банктік карталар және криптондық валюта қабылдайды. Осылайша, биржаға фиат қою өте оңай, бұл, мүмкін, оның басты артықшылығы.

Bitstamp - Еуропадағы алғашқы реттелетін крипто-валюта биржасы. Компания пайдаланушы қаражатының 98% - ын суық әмияндарда сақтайды, яғни қауіпсіздікті жоғары деңгейде қолдайды. Bitstamp-Банктік карталар мен аударымдарды қабылдайтын бірнеше биржалардың бірі. Ол EUR және USD қарсы үш ірі крипто-валюта сатуға мүмкіндік береді. Шын мәнінде, Bitstamp крипто-валюта әлеміне кіру үшін тамаша мүмкіндік береді, бірақ оған альткоиндарды сату мүмкін емес.

Келесі 3.13 және 3.14 суреттерде бағдарлама арқылы құрылған диаграммаларды қарастырамыз. Бұл диаграммалар әр түрлі интернет желісіндегі қазіргі кезде дамып келе жатқан Bitcoin (BTC) теңгерімдік төлем жүйесінің сату бағасының өсуі мен түсуін көрсетеді. Bitstamp - банктік карталар сияқты дәстүрлі төлем әдістері арқылы ақша әкелуі мүмкін бірнеше биржалардың түрлерінің бірі. Суретте Bitstamp биржасындағы bitcoin криптовалютасының доллар (USD) бағамындағы күйі көрсетілген.



Сурет 3.13 - Bitstamp биржасындағы биткоин күйінің диаграммасы

Сурет 3.14 - Bitstamp биржасындағы биткоин күйінің жапон шамдары диаграммасы

Келесі суретте бірнеше биржаның күйі әр түрлі диаграммаларда көрсетілген (3.15 сурет).

Сурет 3.15 – Биржалардың әр түрлі диаграммалардағы күйі

3.7 SBE және JSON салыстырмалы талдауы

Қазіргі уақытта қарапайым екілік кодтаудың (Simple Binary Encoding) толық аналогы болатын ешқандай хаттама жоқ. Бірақ деректерді жіберу үшін ұқсас хаттамалар бар. Салыстырмалы талдау үшін JSON деректерді жіберу әдісі таңдалды. Ол JavaScript-пен өзара әрекеттесу кезінде деректердің ең ыңғайлы форматтарының бірі. Серверден деректерді алу және клиенттік бөлікке жберу қажет болған жағдайда қолданылады. Осылайша, аралық формат – желі бойынша жіберілуі үшін JSON ең ыңғайлы және кең таралған болып саналады.

Сериализацияландыру үшін деректерді жіберу жылдамдығын қадағалау үшін бірдей қаржылық деректер пайдаланылды. Деректер базасының құрылымы күрделі емес, бірақ ол үлкен көлемде деректерге ие. SBE неғұрлым күрделі құрылымдарға арналғанын ескере отырып, оның қарапайым деректер құрылымдарының дерекқорларымен әрекеттері қызықтырды.

Төменде қаржылық деректердің сериализациялануы мен десериализациясы және олардың дерекқорлармен өзара әрекеттесуі сипатталады:

Server side:

```
JsonObject json = new JsonObject();
json.addProperty("id", id);
json.addProperty("datetime", datetime);
json.addProperty("open",open);
json.addProperty("high", high);
json.addProperty("low", low);
json.addProperty("close", close);
json.addProperty("volume", volume);

ByteBuffer source = ByteBuffer.wrap(json.toString().getBytes());
```

Client side:

```
JsonParser parser = new JsonParser();
JsonObject json = parser.parse(data_string.trim()).getAsJsonObject();
```

Save data in database:

```
Date date = new Date(json.get("datetime").getAsLong());
Time time = new Time(json.get("datetime").getAsLong());

preparedStatement.setLong(1,json.get("id").getAsLong());
preparedStatement.setDate(2,date);
preparedStatement.setTime(3,time);
preparedStatement.setFloat(4,json.get("open").getAsFloat());
preparedStatement.setFloat(5,json.get("high").getAsFloat());
```

```
preparedStatement.setFloat(6,json.get("low").getAsFloat());
preparedStatement.setFloat(7,json.get("close").getAsFloat());
preparedStatement.setLong(8,json.get("volume").getAsLong());

preparedStatement.execute();
```

Сонымен қатар, кодтау мен сериализацияландыру арасындағы айырмашылықтар бар екенін атап өткіміз келеді. Сериализацияландыру кез келген объектіні жолға айналдыру процесі болып табылады және десериализация осы жолды өзінің туынды нысанына айналдырады. Және кодтау қауіпсіздік аспектілеріне, деректерді қорғауға жатады, деректерді декодтаудан басқа ешкім түсінбейтін кодталған форматқа түрлендіреді, деректерді кодтау үшін алгоритмді қолданады. Қарапайым құрылымына қарамастан, SBE JSON-ге қарағанда деректерді шифрлау және кодтауда жылдамдығының көбірек екендігін көрсетті.

3.2-кестеде және 3.1-диаграммада нәтижелерді көруге болады:

3.2-кесте - салыстырмалы талдаудың нәтижелері

Деректер саны	SBE	JSON
1	~30000ns	~40000ns
1000	~271ms	~392 ms
10000	~1998ms	~2412ms

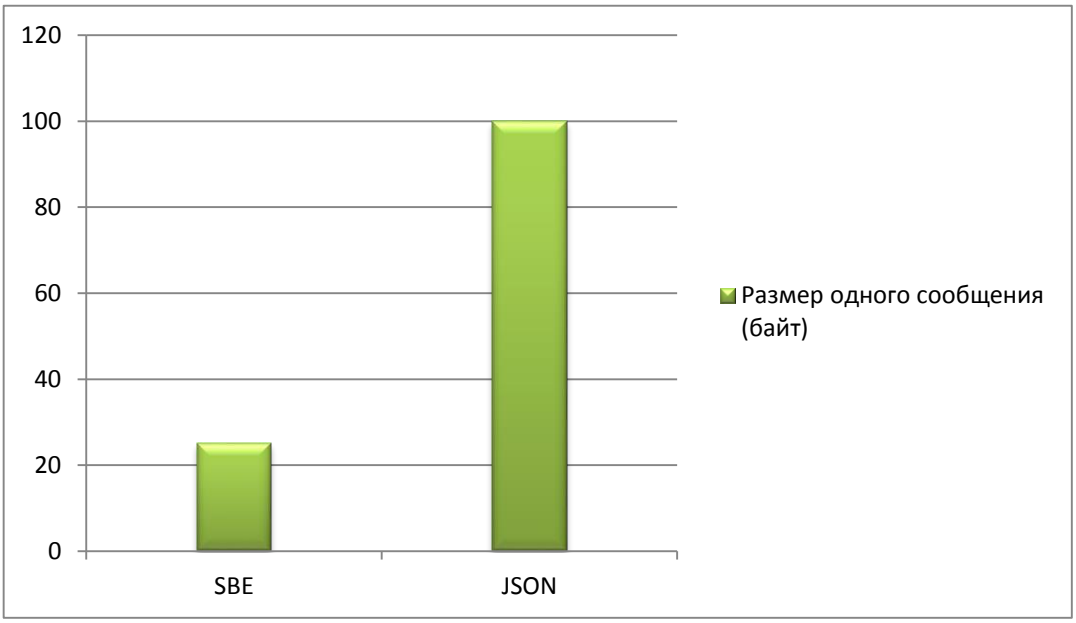
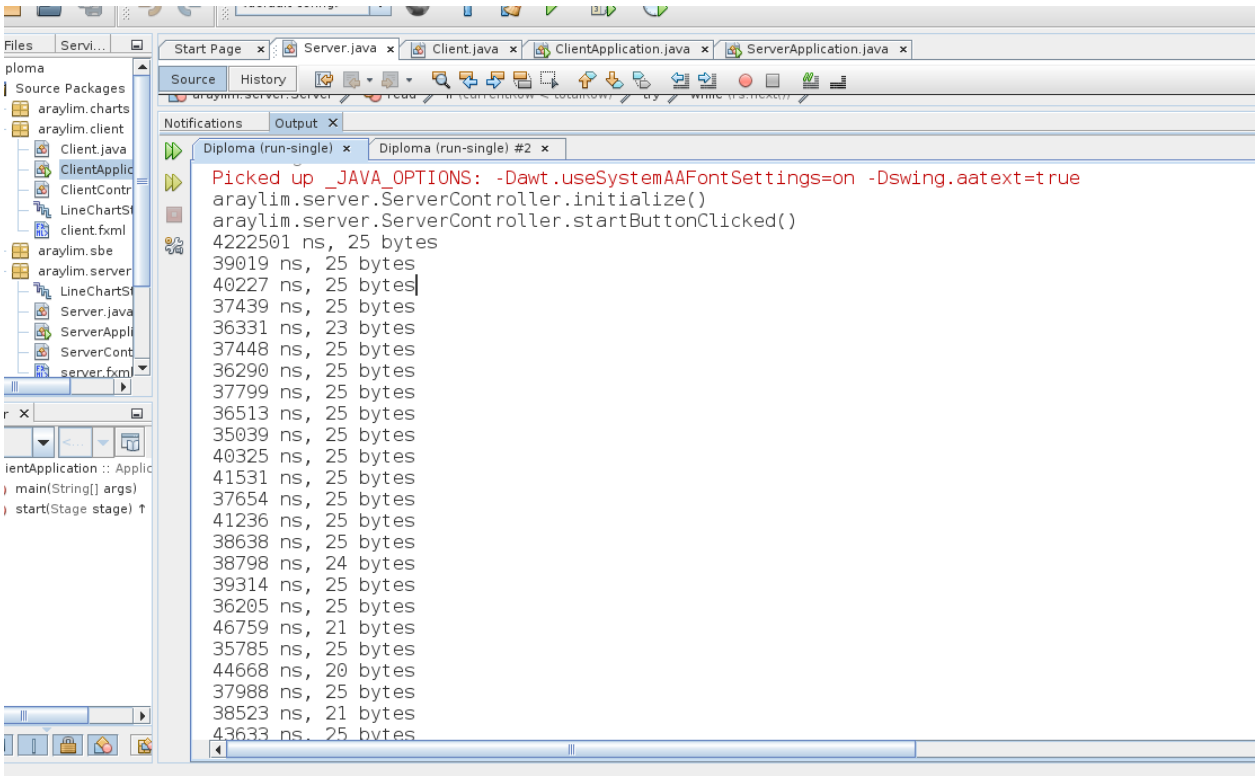
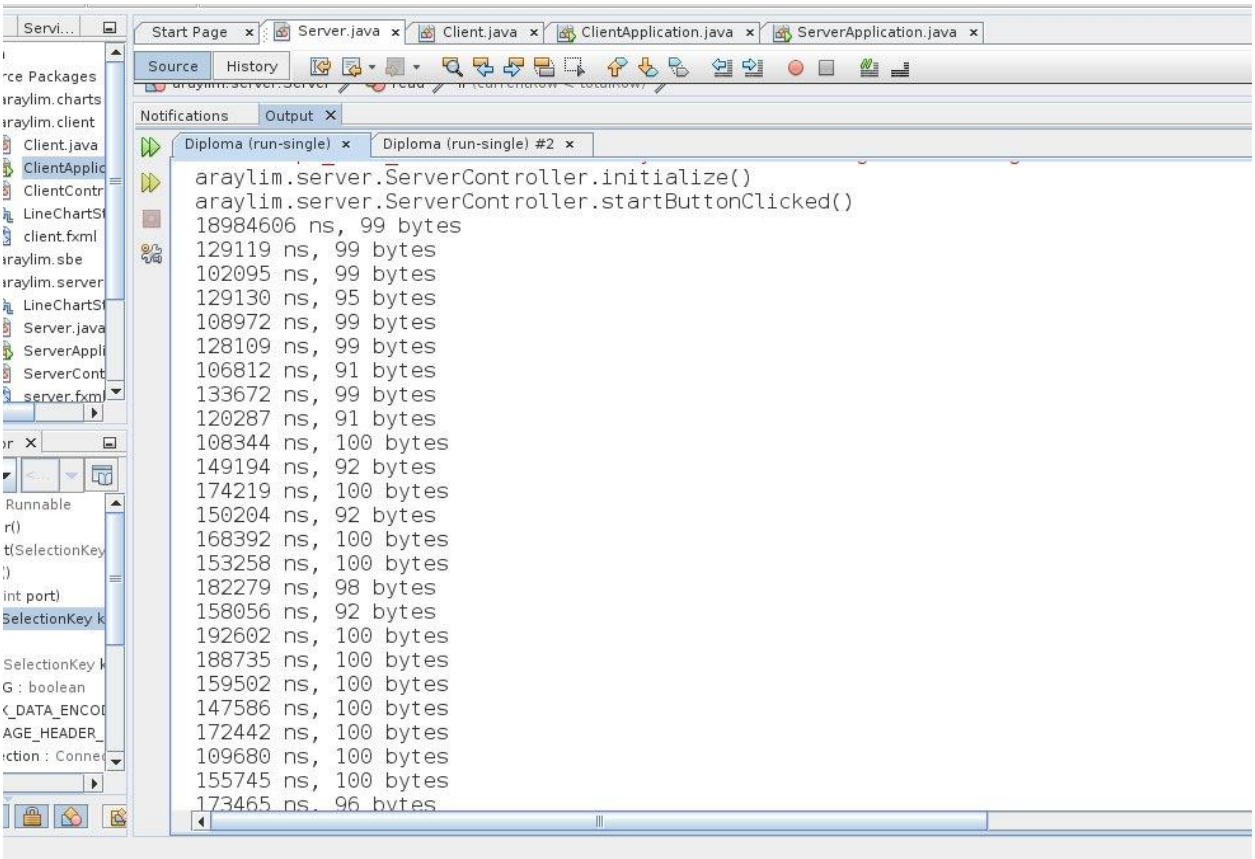


Диаграмма 3.1 - Жіберу кезінде бір хабарламаның көлемі



Сурет 3.16 - SBE хабарының өлшемі



Сурет 3.17 - JSON хабарының өлшемі

Нәтижелер SBE протоколы JSON протоколына қарағанда жылдам жұмыс істейтіндігін көрсетті. Сондай-ақ, серверден клиентке жіберілетін бір хабарламаның өлшемі JSON-нан SBE хабарламасының өлшемі 4 еседен аз.

3.16 және 3.17 суреттерде SBE және JSON арқылы жіберілген әрбір хабарламалардың өлшемдерінің нәтижелері көрсетілген.

Қорытынды

Бұл жұмыстың нәтижесі - SBE хаттамасын қолдану арқылы қаржылық ақпараттарды көп мөлшерде жедел жіберу туралы сұранысты әзірлеу болды. Негізгі процесс клиенттен қосылуды растауды күткен сервермен басталады. Оң жауап алған кезде, сервер дерекқордан қаржы деректерін алады, оны Simple Binary Encoding арқылы кодтайды және оны клиентке жібереді. Ақпараттың көп мөлшерін алатын клиент оны кодтан шығарады және оларды дерекқорда сақтайды. Салыстырмалы талдаудың нәтижелері Simple Binary Encoding қарапайым екілік кодтауды қолдану жылдамдықтың ұлғаюын және қаржы деректерінің үлкен көлемін өңдеуін дәлелдеді.

Зерттеу жұмысының нәтижесі бойынша қор биржаларында қолданылатын қаржылық ақпараттарды тез арада жіберу үшін SBE және JSON хаттамаларын салыстыруы жүргізілді. Нәтижелер SBE протоколы JSON-нан жылдамырақ екенін көрсетті. Сондай-ақ, серверден клиентке SBE үшін көшіру кезінде бір хабарламаның өлшемі JSON-нан 4 есе аз. Салыстырмалы талдаудың нәтижелері қарапайым екілік кодтауды қолдануда қаржы деректерінің үлкен көлемін жіберу және өңдеу жылдамдығын арттыру мақсатына көз жеткізді.

Bitstamp криптовалюта биржасындағы деректері алынып, дәл осы уақыттағы мәліметтердің, яғни, биткоин бағамының күйін мониторинг жүргізуге болатын интерфейске сызықтық графиктер және жапон шамдары ретінде алынды. Магистрлік диссертация тақырыбы бойынша басқа да тапсырмаларды орындау үшін нәтижелер табысты қолданылды.

Пайдаланылған әдебиеттер тізімі

1. https://www.forbes.com/sites/gilpress/2013/05/09/a-very-short-history-of-big-data/

2. Большие данные, 2018. https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5

3. Что такое Big data: собрали всё самое важное о больших данных., 2017. https://rb.ru/howto/chto-takoe-big-data/

4. Объем данных всего мира к 2025 году увеличится в 10 раз., 2017. https://aboutdata.ru/2017/04/27/volume-of-data-by-2025/

5. Что такое Big Data: классификация и примеры., 2018. https://neurohive.io/ru/osnovy-data-science/big-data/

6. Awanish. «Big Data Tutorial: All You Need To Know About Big Data!», 2010.

7. Margaret Rouse. «Big data tutorial: Everything you need to know.», 2018.

8. Marshall Cline. «C++ FAQ: «What's this «serialization» thing all about?», 2015.

9. https://en.wikipedia.org/wiki/Comparison_of_data_serialization_formats#Syntax_comparison_of_human-readable_formats 2019-02-20

10. https://github.com/tfgm/sbedecoder, 2019.

11. https://docs.python-guide.org/scenarios/serialization/ 2018-12-22

12. https://www.investopedia.com/terms/f/financial-information-exchange.asp

13. http://fixprotocol.io/fix-simple-binary-encoding/

14. https://github.com/FIXTradingCommunity/fix-simple-binary-encoding

15. https://www.fixtrading.org/standards/sbe/

16. https://ru.wikipedia.org/wiki/Криптовалюта

17. https://alpari.com/ru/beginner/glossary/cryptocurrency/

18. https://ru.investing.com/crypto/, 2019.

19. https://vse-dengy.ru/fondovyiy-ryinok/tehnicheskiy-analiz/vidyi-birzhevyih-grafikov.html, 2019.

20. D. Mendelson, F. Malabre, Simple Binary Encoding For High Performance Market Data Interfaces, 2018.

21. A Look Into FIX Protocols, https://medium.com/xtrd/a-look-into-fix-protocols-72ec15868e65

22. B. Marr., How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read., 2018. https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#acdaf660ba99

23. Data Age 2025: The Evolution of Data to Life-Critical Don't Focus on Big Data; Focus on the Data That's Big., 2017. https://www.seagate.com/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf

24. Forecast of Big Data market size, based on revenue, from 2011 to 2027 (in billion U.S. dollars), 2018.

25. Латентность, https://ru.wikipedia.org/wiki/%D0%9B%D0%B0%D1%82%D0%B5%D0%BD%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C

26. Mechanical Sympathy, Simple Binary Encoding, 2014. https://mechanical-sympathy.blogspot.com/

27. BestWebSoft (2013, 10 қаңтар) *Forex Historical Data* [Онлайн]. http://www.histdata.com/download-free-forex-historical-data/?/metatrader/1-minute-bar-quotes/EURUSDo

28. M. Thompson. SBE: Design Principles, 2016. https://github.com/real-logic/simple-binary-encoding/wiki/Design-Principles

29. М. Томсон (2014, 6 қаңтар) *Real-logic/simple-binary-encoding* [Онлайн]. https://github.com/real-logic/simple-binary-encoding/wiki/FIX-SBE-XML-Primer

30. М. Томсон (2014, 5 мамыр) *Mechanical Sympathy* [Онлайн]. http://mechanical-sympathy.blogspot.com/2014/05/simple-binary-encoding.html

31. JavaPortal.ru (2013, 12 мамыр) *Что такое JDBC ?* [Онлайн]. http://www.javaportal.ru/java/tutorial/tutorialJDBC/intro.html