КЫРГЫЗ-ТҮРК МАНАС УНИВЕРСИТЕТИ
ТАБИГЫЙ ИЛИМДЕР ИНСТИТУТУ
КОМПЬЮТЕР ИНЖЕНЕРИЯСЫ БАГЫТЫ

ТҮРК ТИЛИНЕН КЫРГЫЗ ТИЛИНЕ ТЕКСТ КОТОРУУ ПРОГРАММАСЫ

(МАГИСТРДИК ДИССЕРТАЦИЯ)

Бахориддин ДУШАБАЕВ

БИШКЕК 2010

КЫРГЫЗ-ТҮРК МАНАС УНИВЕРСИТЕТИ
ТАБИГЫЙ ИЛИМДЕР ИНСТИТУТУ
КОМПЬЮТЕР ИНЖЕНЕРИЯСЫ БАГЫТЫ

ТҮРК ТИЛИНЕН КЫРГЫЗ ТИЛИНЕ ТЕКСТ КОТОРУУ ПРОГРАММАСЫ

(МАГИСТРДИК ДИССЕРТАЦИЯ)

Бахориддин ДУШАБАЕВ

Жетекчи
Доц. Др. Райымбек СУЛТАНОВ

БИШКЕК 2010

İntihal Yapılmadığını Belirten İfade

Ben bu tezdeki bütün bilgileri akademik ve etik kurallarına göre aldığımı ve sunduğumu belirtiyorum. Bu çalışmaya özgün olmadan kullandığım bütün materyal ve bilgilere akademik ve etik kurallar gereğince atıfta bulunduğumu ve hiçbir şekilde intihal yapmadığımı belirtiyorum.

İSİM, SOYAD: **Bahoriddin DUŞABAYEV**
İMZA:
TARİH: **18.06.2010**

Плагиат жасалбагандыгы тууралуу билдирүү

Мен бул эмгекте алынган бардык маалыматтарды академиялык жана этикалык эрежелерге ылайык колдондум. Тагыраак айтканда бул эмгекте колдонулган бирок мага тиешелүү болбогон маалыматтардын бардыгын тиркемеде так көрсөттүм жана эч кайсы жерден плагиат жасалбагандыгына ынандырып кетким келет.

АТЫ, ЖӨНҮ: **Бахориддин ДУШАБАЕВ**
КОЛУ:
ДАТАСЫ: **18.06.2010**

KIRGIZİSTAN TÜRKİYE

MANAS ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE

Mühendislik Anabilim Dalı, Bilgisayar mühendisliği Bilim Dalı'nda 0851Y01002 numaralı Bahoriddin DUŞABAYEV'in hazırladığı "Türkçeden Kırgızcaya metin çeviren yazılımın geliştirilmesi" konulu Yüksek Lisans Tezi ile ilgili tez savunma sınavı, .../...../ 20.... günü …… - ……..saatleri arasında yapılmış, sorulan sorulara alınan cevaplar sonunda adayın tezinin/çalışmasının …….. (başarılı/başarısız) olduğuna ……(oybirliği/oy çokluğu) ile karar verilmiştir.

Üye (Tez Danışmanı)
Akademik Unvanı, Adı Soyadı
Üniversitesi

Üye
Akademik Unvanı, Adı Soyadı
Üniversitesi

Üye
Akademik Unvanı, Adı Soyadı
Üniversitesi

Üye
Akademik Unvanı, Adı Soyadı
Üniversitesi

Üye
Akademik Unvanı, Adı Soyadı
Üniversitesi

...../...../ 20.....

ЧЕЧИМ

Кыргыз-Түрк Манас университетинин Табигый Илимдер Институтунун экзамендик инструкциясынын ...- жобосуна ылайык,..... № жыйында уюшулган комиссия, компьютер инженериясы бөлүмүнүн магистранты “Түрк тилинен кыргыз тилине текст которуу программасы” темасында жазган дипломдук проектин анализдеп, ...-ж. саат .....дө жактоого кабыл алды.

Магистрант .... минута убакыт ичинде дипломдук проектин жактап, комиссия *көпчүлүк добуш менен/бир добуштан Кабыл алынбайт/Кабыл алынсын/Кайра оңдолсун* деген чечим чыгарды.

Жюри төрагасы

Жюри мүчөсү Жюри мүчөсү

Жюри мүчөсү Жюри мүчөсү

...../...../ 20.....

ÖZ

Yazar	: Bahoriddin DUŞABAYEV
Üniversite	: Kırgızistan-Türkiye MANAS Üniversitesi
Anabilim Dalı	: Mühendislik
Bilim Dalı	: Bilgisayar Mühendisliği
Tezin Niteliği	: Yüksek Lisans Tezi
Sayfa Sayısı	: xiv + 56
Mezuniyet Tarihi	: 18 / 06 / 2010
Tez Danışman(lar)ı	: Doç.Dr. Rayımbek SULTANOV

TÜRKÇEDEN KIRGIZCAYA METİN ÇEVİREN YAZILIMIN GELİŞTİRİLMESİ

Bitirme tezi çalışmasında Türkçeden Kırgızcaya metin çevirme algoritması ve o algoriitmaya gire geliştirilen Türkçeden Kırgızcaya metin çeviren yazılımın çalışma yöntemleri hakkında bilgi verildi. Geliştirilen yazılımın önemine de değinildi.

Çeviri nedir, çevirinin geçmişi, otomatik çeviri, otomatik çevirinin geçmişi ve geleceği, faydası hakkında bilgi verildi. Günümüzde en yaygın kullanılan çeviri algoritmalarının çalışma yöntemlerine da değinildi. Yazılımı geliştirmede kullanılan programlama dili ve veritabanı hakkında da bilgi verildi. En sonunda geliştirilen algoritmanın çalışma yöntemlerine değinildi. Yazılımın bazı önemli kaynak kodları verildi.

Anahtar Sözcükler

Makine Çevirisi, Dil, Türkçe, Kırgızca, Cümle, Program, Yazılım

КЫСКАЧА МАЗМУНУ

Даярдаган	: Бахориддин ДУШАБАЕВ
Университет	: Кыргыз-Түрк Манас Университети
Адистик	: Инженерия
Багыты	: Компьютердик Инженерия
Иштин сыпаты	: Магистрдик Диссертация
Беттердин саны	: xiv + 56
Бүтүрүү датасы	: 18 / 06 / 2010
Илимий Жетекчи	: Доц.Др. Райымбек СУЛТАНОВ

ТҮРК ТИЛИНЕН КЫРГЫЗ ТИЛИНЕ ТЕКСТ КОТОРУУ ПРОГРАММАСЫ

Бул магистрдик диссертацияда түрк тилинен кыргыз тилине текст которуу алгоритми жана ал алгоритмдин негизинде иштелип чыккан түрк тилинен кыргыз тилине текст которгон программанын иштеши каралган. Ушундай эле иштелип чагылган программдык жабдыктын маанилүүлүгү жөнүндө да сөз болгон.

Которуу деген эмне, анын тарыхы, автоматташтырылган которуу жөнүндө, анын тарыхы жана келечеги, пайдасы жөнүндө токтолуп өтүлгөн. Учурдагы эң кеңири таралган которуу ыкмаларынын иштөө приципте рине да токтолуп өтүлгөн. Программдык жабдыкта колдонулган программалоо тили жана берилиштер базасы жөнүндө да маалымат берилген. Эң аягында иштелип чыккан алгоритмдин кантип иштеши жөнүндө сөз болгон. Программдык код берилиштер базасын колдонуп, текстти кантип которoруна токтолуп өтүлгөн.

Ачкыч Сөздөр

Машина Котормосу, Тил, Түркчө, Кыргызча, Сүйлөм, Программа, Программдык Жабдык

АБСТРАКТ

Автор	: Бахориддин ДУШАБАЕВ
Университет	: Кыргызко-Турецкий Университет МАНАС
Направление	: Инженерия
Отделение	: Компьютерная Инженерия
Уровень работы	: Магистерская Диссертация
Количество страниц	: xiv + 56
Выпускной Дата	: 18 / 06 / 2010
Научный руководитель	: Доц.Др. Райымбек СУЛТАНОВ

РАЗРАБОТКА ПРОГРАММЫ ДЛЯ ПЕРЕВОДА ТЕКСТА С ТУРЕЦКОГО НА КЫРГЫЗСКИЙ

В данной магистерской диссертации рассматриваются существующие алгоритмы перевода текста с турецкого на кыргызский язык, и предлагается алгоритм работы программы перевода текста с турецкого на кыргызский язык. Также рассматривается значимость разработанного программного обеспечения.

В работе раскрыто понятие «перевод», освещено история перевода, а также рассмотрено сущность автоматизированного перевода, его развитие и значимость. В рамках данного исследования затронуты принципы самых широко распространенных методов перевода, предоставлены данные о языке программирования и базе данных, использованных в разработке программного обеспечения. В заключении раскрыт принцип работы разработанного алгоритма, а также то, каким образом программный код при помощи базы данных переводит текст.

Ключевые Слова

Машинный Перевод, Язык, Турецкий, Кыргызский, Предложение, Программа, Программное Обеспечение

ABSTRACT

Author	: Bahoriddin DUSHABAYEV
University	: Kyrgyz-Turkish MANAS University
Direction	: Engineering
Department	: Computer Engineering
Property of thesis	: Master's thesis
Number of Pages	: xiv + 56
Graduation Date	: 18 / 06 / 2010
Scientific Advisor	: Doc.Dr. Rayimbek SULTANOV

SOFTWARE FOR TEXT TRANSLATION FROM TURKISH TO KYRGYZ

This Master's thesis project includes translation algorithm of Turkish text in to Kyrgyz language and according to this algorithm produced software which translates Turkish text in to Kyrgyz language. At the same time explained the main role of the software.

Replied for some question like: What is translation, it's history, automation translation, the futures and usefulness. Also explained about wide spread method of the translation principles and program language which used during programming, information about database. At the conclusion there is information about how the algorithm will work. Source code using the database explains how to translate the text.

Key Words

Machine Translation, Language, Turkısh, Kyrgyz, Sentence, Programme, Software

БАШ СӨЗ

Магистрдик диссертация. 56 бет, 8 сүрөт, 6 адабият

Максаты: Түркчө берилген тексттин мүмкүн болушунча эң жакшы, кыргызчага которулган котормосун алуу. Түркчөдөн кыргызчага текст которгон программдык жабдык ушул күнгө чейин иштелип чыгыла элек болуп, бул проекттин ишке ашышы менен алгачкы түркчөдөн кыргызчага текст которгон программлык жабдык иштелип чыккан болот.

Актуалдуулугу: Киши каалаган маалыматка өз убагында жетүү, кыска убакыттын ичинде каалаган макаланы же башка бир тилде жазылган китептерди, технологиянын таасири астында ылдамдык менен өнүккөн тетиктердин колдонмо китепчелерини, келген э-почталарды, веб баракчаларыны жана башка тилде жазылган ар кандай тексттерди котормочуга барбай туруп, эң аз ката менен өз тилине которууну каалайт!

Күнүбүздөгү технология чоң ылдамдык менен өнугуп баратат жана биз андан артта калбашыбыз керек, эң идеалдуу шарттарда жашап жатсак дагы, тил билүү проблемабыз болбосо дагы, ал үчүн керегинден көп убакыт коротпой туруп керек болгон маалыматтарга жетүүбүз керек. Бирок бул жерде алдыбызга чыга турган эң чоң тоскоолдук "которуу"нун кыйынчылыктары.

Бизге керек болгон, башка бир тилде жазылган маалыматтарды котормочуга берип, белгилүү бир акча төлөп котортурсак да болот. Бирок бул бизге убакыт жана акча жагынан ашыкча чыгаша болот. Азыр маалымат доору болгондуктан маалыматка бардыгынан мурун жетишибиз керек, ошондуктан убакытты жоготуу бизге өтө чоң зыян келтириши мүмкүн. Адам котормосу көп убакыт талап кылышы менен бирге, оригинал тексттин чыныгы маанисинин өзгөрүүсүнө да себеп болот. Себеби адам баласы ар дайым өзүнүн ойуну билдирүүнү жакшы көрөт. Котормочу да адам баласы болгондуктан текстти которууда өз ойуну кошуусу табигый көрүнүш. Оригинал тексттеги маанинин бузулушу кээде ал жерде айтылган нерсенин туура эмес түшүнүлүшүнө алып келет.

Машина, башкача айтканда компьютер котормосу ушул себептүү пайдалуу. Каалаган текстти каалаган убакытта котортура алабыз. Компьютер бизден эч качан акча талап кылбайт. Эч качан чарчабайт. Текстти которуп

жатканда өзүнүн ойуну кошпойт. Бул да болсо маанини өзүндөй сакталышы болуп эсептелет.

Магистрдик ишти аягына чыгаруумда менден ар кандай жардамыны аябай жана ар кайсы жактан колдоп тургандыгы үчүн илимий жетекчим Доц.Др. Райымбек СУЛТАНОВко өз ыраазычылыгымды билдиргим келет. Ушундай эле Табигый илимдер институту мугалимдерине, компьютердик инженерия бөлүмүнүн мугалимдерине да чоң ыраазычылыгымды билдирем.

МАЗМУНУ

КЫСКАРТУУЛАР

Кыскартуулар	Библиографик маалымат
АМС	Автоматташтырылган Маалымат Системасы
ИМАС	Илимий-Техникалык Маалыматтын Автоматташтырылган Системасы
МК	Машиналык Которуу
МТК	Машинага Таянган Которуу
SQL	Structured Query Language – Структурланган Сурамжылоо Тили
ANSI/ISO	Amrican National Standards Institute / International Standards Organization
MSSQL	Miscrosoft SQL
OOP	Object Oriented Programming – Объектке Багытталган Программалоо
JVM	Java Virtual Machine – Java Виртуал Машинасы
VB	Visual Basic
JDBC	Java Database Connectivity
IL	Intermediate Language – Орто Деңгээлдеги Тил
ADO	ActiveX Data Objects
COM	Component Object Model
GUI	Graphical User Interface – Колдонуучунун Графикалык Интерфейси
TDS	Tabular Data Stream
ODBC	Open Database Connectivity
OLAP	Online Analitical Processing
SOAP	Simple Object Access Protocol
ETL	Extract, Transform, Load
SSIS	SQL Server Integration Services

КОЛДОНУЛГАН СҮРӨТТӨРДҮН ТИЗМЕСИ

БИРИНЧИ БӨЛҮМ

КИРИШ ЖАНА ПРЕДМЕТТИК БӨЛҮМДҮН АНАЛИЗИ

1.1. КИРИШ

Каралып жаткан проект клиенттик жана административдик көрсөткүчтөргө негизделген программалык продукттун иштелип чыгышына арналган. Мында предметтик бөлүм боюнча негизги түшүнүктөр жана учурдагы колдонулуп жаткан которуу системаларынын принциптери каралган жана анын визуалдык түрү көрсөтүлгөн. Ошондой эле информациялык системаны динамикалык долборлоо үчүн программалык элемент катары колдонулган алгоритмдер каралды.

Проекттин максаты – бул учурдагы программалык жабдыктарды колдонуп, азыркы маалымат технологиялардын талабына жооп бере турган програмдык жабдыкты иштеп чыгуу.

Технология ушул күндө ылдамдык менен өсүп баратат. Биз да ушул технологиялардан албетте артта калгыбыз келбейт. Эң идеалдуу шарттарда жашагап жатканыбызды эсепке алсак жана тил билүү проблемалары болбосо дагы бул үчүн керегинден көп убакыт сарптабай керектүү маалыматтарга жетишишибиз керек. Бирок бул жолдо каршыбызга чыккан эң чоң тоскоолдук “которуу” болуп эсептелет.[1]

Адам баласы бир текстти которуп жатканда чарчайт. Жашоодо көнүп калган жана мээсине таасирин көргөзгөн ойлорун которуп жаткан текстке төккүсү келет, себеби бул ар бир кишинин ичинде болгон сезим. Ушуга окшогон проблемаларды чечүү үчүн компютерге таянган автоматташтырылган которуу керектелет. Компьютер чарчабайт, 24 саат иштей алат, эс алууну талап кылбайт. Үйрөнгөн эч бир нерсесин унутпайт, жашоо факторлорунан тажрыйба ала албайт, пайда көрүүнү каалабайт, бирок бизге пайда келтирет. Тексттердин тууралыгыны түзө алат, ар бир саат жана ар бир күн берилиштер базасыны кеңейтирип, каалаган убакытта колдоно алат. Которуу жасап

жатканда изилденбейт, өзүнүн ой-пикирлерин кошпойт. Ушул себептүү негизги тексттин маани өзгөрбөйт. Дүйнөдө болуп жаткан окуяларды билүү үчүн бир гана клавишке басышыбыз жетиштүү. Каалаган маалыматты жана китепти каалаган убакытта алып окуш мүмкүн, каалаган кишилер менен байланыштарды түзүүгө болот[1].

Проектте төмөндөгү маалыматтар камтылды:

Биринчи главада предметтик бөлүм (которуу системасынын иштөө принциби) боюнча негизги түшүнүктөр, проектти долборлоодо колдонулган ыкмалар баяндалган.

Экинчи главада жогорку бөлүмдө баяндалган ишти ишке ашырууда колдонулуучу программалык жабдыктар жана алгоритмдер тууралуу маалыматтар келтирилген. Бул программдык жабдыктардын колдонулушу, өзгөчөлүктөрү жана кемчиликтери мисалдар аркылуу ачылып, көрсөтүлдү.

Үчүнчү главада проектти ишке ашыруу үчүн түзүлгөн берилиштер базасы жана програмдык код бөлүгү берилген жана иштөө принциби баяндалган.

Проекттин максаты: учурдагы программалык жабдыктарды колдонуп, азыркы маалымат технологияларынын талабына жооп бере турган програмдык жабдыкты иштеп чыгуу.

Маселенин коюлушу: Кыргыз жана Түрк тилдеринин түзүлүштөрүн анализдөө. Которуу ыкмаларынын иштөө принциптерин изилдөө жана которуу үчүн тандап алынган ыкманын иштөө принцибин анализдөө.

Актуалдуулугу: Азыркы заманда дүйнө жүзү боюнча маалымат системасы бардык сфераларда, анын ичинде бир тилден экинчи тилге которууда да кеңири колдонулууда. Бирок кыргыз тилине которуу системасы иштеп чыгыла элек. Каралып жаткан проект бул жолдо которуу ишин

жеңилдетүүгө бираз да болсо жардамчы болот, жана кыргыз тилинин өнүгүшүнө өз салымын кошот.

1.2. КОТОРУУ ТҮШҮНҮГҮ

1.2.1 Которуу деген эмне?

Технологиянын ушул күндө ылдамдык менен өсүп баратат. Биз да ушул технологиялардан албетте артта калгыбыз келбейт. Эң идеалдуу шарттарда жашаганыбызды экенибизди эсепке алсак жана тил билүү проблемалары болбосо дагы бул үчүн керегинден көп убакыт сарптабай керектүү маалыматтарга жетишишибиз керек. Бирок бул жолдо каршыбызга чыккан эң чоң тоскоолдуктардан бири «которуу» болуп эсептелет[1].

Бир тилдеги сөз каршылыгы жана сүйлөм структурасыны анын грамматикалык түзүлүшүнө карата, башка тилдин грамматикалык эрежелери чегинде маанисини жоготпой туруп которуу операциясына которуу деп айтылат. Кээде чеберчилик, кээде оор иш, кээде оңой иш, профессионал түрдө жана ушуга окшош сөздөр менен айтылат. Чеберчилик болсун же болбосун, жыйынтыгында бир эле талап бар. Которуунун мантыктуу жана туура болушу[1].

Которуунун тууралыгы, аны кайра терс которгондо айтылган маанини сактап калышы менен өлчөнөт. Башкача айтканда которуунун мааниси жана таасири которуу жасалган тексттен күчтүү же күчсүз болбошу керек. Бир котормонун тууралык пропорциясы анын барабардык таасири менен өлчөнөт, мисал үчүн, “o kötü değildir” сүйлөмүнүн котормосу “ал жакшы” деп которулса ката болуп эсептелет, себеби терс которуу жасалганда “o iyidir” болуп которулат жана котормо жасалуучу сүйлөмдүн таасири жоголот. Демек туура котормосу “ал жаман эмес” болушу керек, ушундайча котормо пропорциясы сакталат[1].

1.2.2 Туура которуу

Туура которууга аныктама бериле турган болсо ушул күндө практика жүзүндө муну ишке ашыруу мүмкүн эмес. Жаңы жаңы фразеологизмдердин жаралышы жана ага байланыштуу сөздүктөр негизине маалымат агымыны контрол этүү котормочунун жасай ала турган иши эмес жана анын иши да эмес. Ушул себептүү, котормодо которула турган булактын тууралыгы азайып анын табигый структурасынын жоголушуна себеп болууда. Бул өзгөчөлүктөр, кээ бир тексттерде эсепке алынбаса дагы, кененирээк алып караганда аны эсепке албай койууга мүмкүн эмес[1].

1.2.3 Туура эмес которуу

Которула турган тексттеги сөздөрдү бирме-бир бөлүп алып аны экинчи бир тилдеги маани каршылыгын таап которсо да болот. Эгер бул жол менен туура сүйлөм түзүлө алса, котормо туура болот, болбосо туура эмес котормо болот. Тажрыйбалуу котормочулар которуунун эрежелерин жакшы билишет. Тез которуу үчүн сүйлөмдөрдү кыскартырып, маанисин өзү каалаган багытка буруп өзүнүн ойлорун кошкондон да чегинишпейт. Бул да болсо которулган тексттин маанисинин тап-такыр өзгөрүп кетишине алып келет[1].

1.3. АВТОМАТТАШТЫРЫЛГАН КОТОРУУ

1.3.1 Автоматташтырылган которуу түшүнүгү

Барабардык мыйзамын сактап, булак тексттин бардык жагын катасыз түрдө анализдей ала турган абалга келтирип, каалаган тилге, бүтүндөй маанисин бузбай туруп которуу амалына автоматташтырылган которуу деп аталат[1].

Машина котормосу, бир тилден экинчи бир тилге сүйлөм структурасы, мүчөлөр жана грамматикалык эрежелерин, бардык оңдоонун характеристикаларын туура таанып, эң туура маанини берген котормону алуучу программа болуп эсептелет[1].

Автоматташтырылган которуу системасын куруунун пайдалуу болушу тартышылгыс нерсе. Себеби ал бардык жерде иштей алат, чарчабайт, бат иштейт, ката кылбайт, ишеничтүү, маалыматты керектөөгө кеткен убакытты дээрлик нолго түшүрөт. Элдердин бири-бирин таанышына жардамчы болот жана китепти 6 ай ордуна 1 саатта которо алат. А бирок котормочулар болсо, каалаган убактыбызда которо алышпайт, 5-6 саат иштегенден кийин чарчашат жана туура эмес которуу жасашы мүмкүн[1].

1.3.2 Эң таасирдүү которуу – кыска которуу

Түз которуулар китеп окууда аябай пайдалуу болушу мүмкүн[1].

Түз которуу ишеничтүү болот жана бул ишенич өз таасирин улут ичинде көрсөтө алат. Ушул себептүү, бул аябай маанилүү тема болуп эсептелет жана ал бүтүн жашообузга таасир көргөзө ала тургандай күчкө ээ[1].

Машина которуусу эч кандай жеке пикирди колдонбой туруп, эң туура котормо жасай алат[1].

Бул темада, малекеттик жана жеке секторлордун карашы бир чындыкты ортого койот. Бул да болсо, машина которуусунун эл аралык мааниде кабыл кылынышы болуп эсептелет. Бардык изилдөөчүлөр бул нерсени кабыл кылышкан[1].

Түрдүү түшүнүктөрдө грамматикага таянып, дүйнө деңгээлинде багытыны белгилеп алган автоматташтырылган которуу, Японияда ордуну алып жана ички жакындоолорду жасаган, Европада чоң көлөмдөгү жергиликтүү инвестициялар башталган жана Америкада ачык ойлор ортого коюлган. Мындан кийин котормочулар бардык которууларды стандарттык түрдө катасыз жасай алышса дагы, маалымат агымыны контролдой алышпайт[1].

1.3.3 Автоматташтырылган которуунун тарыхы

Компьютер доору башталгандан бери, бир тилден башка бир тилге которуу жасаган системаларга өзгөчө маани берилген. Бул жөнүндө көп тартышылган, көп инветициялар сарпталган, кызыктуу натыйжаларга баруу жолдорун көрсөткөн бир канча китептер жазылган. Түрдүү ой-пикирлер ортого койулуп тартышылган, бирок ушул күнгө чейин жасалган системаларга карай турган болсок, күтүлгөн натыйжаларга барууда кемчиликтер көп экени байкалат. Инвестициялардын көп бөлүгү Америкадагы SYSTRAN, Европадагы GLOBALİNK жана EUROTRAN га жасалган, бирок ортого койулган жыйынтыктар, 1962 жвлдарынан башталган жыйынтыктарга салыштырмалуу аз болгон. Бул доордо котормо проблемасыны чучүү үчүн бир гана үмүт адам баласы ойлоп тапкан жана технологиянын укмушу болгон компьютерлер болуп эсептелет[1].

Котормочулар теңдеше албайтурган жана адам баласы ойлоп тапкан компьютерлерде жасалган түрдүү котормо программалары, шексиз түрдө бул доор талап кылган эң негизги системалардан бири болот. Бул доордун технологиясы, кааласак да каалабасак да бардык чөйрөлөрдө өз ордун табууда жана программдык пакеттерде, мамлекеттик мекемелерде жана менчик секторлордо колдонула баштады[1].

Жашообуздун бир канча чөйрөлөрүндө таасирин көрсөтүп, жашообузду, байланыштарыбызды кеңейткен компьютерлер, которуу багытында эмнегедир ийгиликтүү жыйынтыктарга ээ боло алышпады[1].

1962 жылдарынан бери, оптимисттик божомолдоолор негизинде, миңдеген конференциялар, 10 миңдеген окумуштуулар, тилчилер, компьютер адистери автоматташтырылган которуу системасын жасоого аракеттенишүүдө. Бирок анчалык жакшы жыйынтыктарга барылган жок. Бул тема өнүгүп келаткан программдык ыкмалар менен көбүрөөк тартышыла турган абалга келип, жалаң гана изилдөөлөрдүн акырындашына жана кээде багытын өзгөртүүсүнө себеп болгон, бирок жыйынтыкты өзгөртүрө алган эмес.

SYSTRAN өңдүү чоң фирмаларда, автоматташтырылган которуу жыйынтыгынын тууралык пропорциясы 42 %, Европада 36 % тен жогору болгон эмес. Бул көрсөткүчтөргө таянып, жасалган программалар 10 % болсун 60 % болсун жалпы жетишсиздикти көрсөтөт. Бирок автоматташтырылган которуунун жасалышы үчүн 90 % тик жыйынтык алынышы керек[1].

Ушул күндө которуу амалы чеберчилик, оор жумуш, жогорку деңгээлдеги билимди талап кылган амал менен түшүндүрүлбөйт. Себеби компьютер баштаган интернет доору үчүн тилдер арасындагы аралыктын эң азга кыскартырыш керек болуп калды. Бул аралыкты кыскартыруу жана муну менен бирге маалымат агымыны контролдоо үчүн, автоматташтырылган которуу системасы керек болот[1].

Автоматташтырылган которуу мурун да көп тартышылган. Эң биринчи ой-пикирлер Европада жана Россияда 1950 жылдарында ортого койулган. Чоң державалардын тарыхта жүргүзгөн согуштары натыйжасында, тил менен байланыштуу проблемаларды көбүрөөк баштан өткөрүшкөндүгү себептүү автоматташтырылган которуу системаларыны колдогон инвестициялар жасаган фонддордун көп бөлүгү Европа, Россия, Америка, Канада өлкөлөрүнө таандык[1].

Автоматташтырылган которуу системаларын түзүүгө башталгандан бери, бул багыттагы изилдөөлөр, Батыш өлкөлөрүнүн тилдерин өз ичине алчу жана азыр дагы өз ичине алат. Бирок 20 жылдык ийгиликсиз изилдөөлөрдөн кийин, эң маанилүү изилдөөлөрдө салымы болгон жана эң ишеничтүү окумуштуу Жанет Пак, автоматташтырылган которууну түзүүнү мүмкүн эмес кылган, 30 беттик бир отчет менен жыйынтыктады. Бул отчет, бардык инвестицияларды токтото турган абалга алып келип, эң көлөмдүү инвестиция менен автоматташтырылган которуунун пайдалуу болушуну ойлогон Америка Деңиз Күчтөрүнү үмүтсүздүккө түшүрдү. Дүйнө жүзүндө да Канада Аба Күчтөрү жана башка жергиликтүү инвесторлор Жанет Пактын отчетуна таянып, бул долбоор үстүндө жасалган иштерди токтотушту. Бул абал 10 жыл ушундайча турду жана андан кийин компьютерлердин ылдамдык менен

өнүгүшү жана программалоо багытында Жасалма Акыл жана 1980 жылында жылдыздар аралык согуш жөнүндөгү программалардын чыгышы менен автоматташтырылган которууну түзүү кайрадан жандана баштады бирок башка бир башталгыч менен[1].

Америка баш болуп, Европа, Советтер Союзу, Япония жана Кытай тарабынан улуттук мааниге ээ болгон долбоор катары жана мамлекеттин колдоосу менен кайрадан баштатылды. 1982 жылында бардык өлкөлөргө, "Эгер тилиңерди сүйсөңөр жана келечекте да сүйлөшүлө турган тил болушуну кааласаңар сүйлөгөн тилиңердин автоматташтыруу платформаларында иштей алышы үчүн керек болгон изилдөөлөрдү баштагыла!" деп, Англисчеден өз тилине автоматташтырылган которуу системасыны түзүү жоопкерчилиги, МК конференцияларында улам-улам кайталанды[1].

Мындан сырткары Американын космостук борбору НАСА, узун убакыттан бери өзүнчө, бул ишти ишке ашырууга ниеттенген болчу. Бирок жылдыздар аралык согуштун, 11 миллион саптан турган программанын жазылышы жана долбоордун ичинде колдонулган тилдерди, электрон толкундарды тутуучулар менен узатылып, контролдонгон системалардын автоматташтырылган которооу системасы тарабынан таанылышы жана соңунда жок кылынышы аябай оор болчу. Бул жолу системаны эмес, ар бир 60 миң сап программада, программага тиешелүү 1-2 ката жана кезек котормого келбей туруп жок кылынып, аябай кымбатка түшүшү мүмкүн болчу. Кезек программа жазган фирмаларга келгенде, көпчүлүк чөйрөлөргө өз таасирин көрсөтүп жаткан Windows системасы, улам-улам жасалган жана жасала турган программалардын, башка тилдерге которулушуну пландаштырган жана кеңейүү политикасына таянып автоматташтырылган система менен бат эле жасалышы жана ишеничтүү болушу менен бирге Азия жана Европа базарында Microsoft фирмасынын экономикалык таасирини чоңойтуп жаңыланма өзгөчөлүгүнө таяндырып, Windows тун продукттарынын автоматташтырылган которуу менен башка тилдерге которулушу, бул жолу дүйнөнүн эң чоң фирмасы тарабынан тартышылып тарыхта жасалган МК системаларынын

түзүлүшүнү ыңгайлуу көрбөгөн, Windows ко окшогон аябай чоң бир система жана ар бир темада ийгликтүү болгон Microsoft, автоматташтырылган которууну жаны ыкмалар менен ишке ашырылышына бел байлап, керек болгон иштер, пландар жана финансы колдоосунда баштады. Эмнегедир пландар көрүнгөндөй оңой жүрбөдү. Тарыхта жасалган иштерди карап чыкканда дагы чечүү жолу табылбаган соң бул иш токтотулду[1].

Кээ бир айтылган сөздөргө караганда, Microsoft 15 % пропорцияны ала алган, андан кийин белгисиз структуралар менен жыйынтык ала албаган соң, Европада жасалган эң жөнөкөй бирок бат иштеген Trados деп аталган Memory системалык программасын сатып алып, автоматташтырылган которуу изилдөөлөрүн ушундайча жыйынтыктады. Trados ту эң акыркы вариант катары өзгөчөлөндүргөн Microsoft, түрдүү тилдер үчүн берилиштер базасын түзүүнү улантты жана ушул күндө дагы улантып келет. Бирок автоматташтырылган которуу долбоорунун токтотулушунун себептери айтылган жок[1].

Жасалган иштер жыйынтыгында, Жанет Пактын отчетуна каршы чыккан борборлор, мындан кийин автоматташтырылган которуу системасыны бул доордо жана бул технология менен ала албайбыз деп Пактын отчетуна макул болушуп жана мурдатан бери багыттандырган чоң көлөмдөгү инвестицияларын токтотушту[1].

Мындан кийинки изилдөөлөр жалаң гана тарыхта кылынган иштерди анализдөөдө. Мындан кийин автоматташтырылган которуу эмес, Машинага Таянган Которуу (МТК) системалары жасалып жатат[1].

Убактынча, МТК системаларыны сунуштаган "Чамеский" дагы, практика жүзүндө керектелген жыйынтыкты алдындагылардан таба албаган соң, котормочулардын стандарт которуусунун кандай болушуну, МК конференцияларында тартыша турган абалга келди, практика жүзүндө жалаң гана Европада эң аз которуу пропорциясына келүү жыйынтыгына барышты[1].

Бирок технологиянын өнүгүшүнө карала турган болсо, Trados өңдүү программалардын мүмкүнчүлүктөрүнүн аз болушуна байланыштуу колдонулушу аябай өкүнүчтүү. Бардык киши муну билет, бирок сунуштала турган башка автоматташтырылган система да жок[1].

1.3.4 Автоматташтырылган которуунун келечеги

Автоматташтырылган которууга, түрдүү бир канча улутта жасалган иштер жана изилдөөлөр, чоң деңгээлде мааниге ээ болгон жана бул жолдо чоң кадамдар жасалышы үчүн чоң көлөмдөгү каржылоолор сарпталган. Ушул себептүү эл аралык жалпы проблемага айланган. Технологияга лидерлик кылган өлкөлөрдө, башкача айтканда Европада, керек болсо Америкада бул ишке чоң маани берилген жана бул иштин фундаменттери көңүл жарытаарлык даражада алдыга жылуулар жасалган. Тарыхка таянып турган болсо, Канада лидерлик ролуну алган жана белгилүү жыйынтыктары себептүү жана тажрыйба кылынган өзгөчөлүгүнү сактоо менен бирге ылдамдык менен өнүккөн[1].

Түштүк Кореяда МК да башынан өткөргөндөрүнө карала турган болсо, менчик сектордун Японияга каршы жасаган инвестицияларды, изилдөөлөрдү аябай көлөмдүү кылууда жана алыскы чыгышта, Европа жана Америкага каршы колго кириткен ийгиликтеринин жыйынтыктарыны алууга баштады[1].

Бул баскычта түрдүү которууга жардамчы болгон программалар (МТК) күн өткөн сайын столдор үстүнөн орун алып баштады жана кичүү колдонуу чөйрөлөрүндө болсо дагы, негиздүү изилдөөлөрдүн кабарлары алынып баштады[1].

20 жылдан кийин, ар бир лабораторияда жана ишканаларда ар түрдүү которуу жасаган программаларды көрүү болсо мантыктуу көрүнөт. Бирок автоматташтырылган которууну жасаганга чейин жардамчы программалардын бир гана эрежеси “ишеничтүү которууну ортого койсо дагы, түшүнүктүү которуу жасасын“[1].

Мындан кийин автоматташтырылган которуу системаларынын, компьютерге таянган которуу системаларынын чыгышы күтүлүүдө жана ушул себептүү колдоого алынып жатат. Автоматташтырылган которуу системалары мындан кийин жеке түрдөгү изилдөөлөргө байланыштуу болушу күтүлүүдө. Ар түрдүү инвестиция жөнүндө сөз болбойт. Эгер жеке түрдөгү изилдөөлөр, кабыл кылына турган баскычтарга келсе, инвестициялар тезинен баштатылат[1].

1.3.5 Автоматташтырылган которууну колдонуу чөйрөлөрү

Күнүмдүк макалалар, тарыхий булактар, web баракчалары, электрондук почталар, жаңылануу керек кылган фабрикаларга маалымат берүү, чыгарылган фильмдердин котормолору, күн сайын өзгөргөн моделдери менен базардан орун алган телефондордун маалымат китепчелери жана жашообуздун бир канча чөйрөсүндөн орун алган жана алууга баштаган байланыш жабдыктарынын маалымат китепчелеринин котормолору автоматташтырылган которууну талап кылат. Эң таасирдүүлөрү болсо:[1]

1. Туризм сектору
2. Чогулуш конференциялары
3. Эл аралык жолугушуулар
4. Китептерди которуу
5. Министрликтер тарабынан чоң көлөмдөгү маалыматтардын талап кылынышы
6. Маалыматтар агымыны контролдоо

Бул 6 баскычты камсыз кылган ар кандай автоматташтырылган платформа ушул күндө жок жана бар болушунун канчалык даражада пайдалуу болушуна да шек жок[1].

1.3.6 Адам баласы которууда эмне үчүн көп ката кетирет?

Адам баласы бир текстти которуп жатканда чарчайт. Ар бир сөз жана фразеологизмдерди эч качан толук биле албайт. Жашоодо көнүп калган жана

мээсине таасирин көргөзгөн ойлорун которуп жаткан текстке төккүсү келет, себеби бул ар бир кишинин ичинде болгон сезим. Бирок Trados өңдүү программалар жардамы менен которуу ишини белгилүү бир стандартка алып келиш мүмкүн, бирок бул жетиштүү эмес жана адам баласынын эң акыркы үмүтү болгон Trados өңдүү Memory программалары ушунчалык даражада билимдин өнүгүшүнө карабастан, өзүнүн терс жактарын көрсөтүп келет. Мисал үчүн, бир бет англисче текстти, 10 түрдүү которуу менен натыйжалантырган адам баласы которуусу ишеничтүү боло албайт. Ушул себептүү китептер окулбай турган абалга келген, көпчүлүк изилдөөлөр үчүн англисче булактардан пайдаланылууда[1].

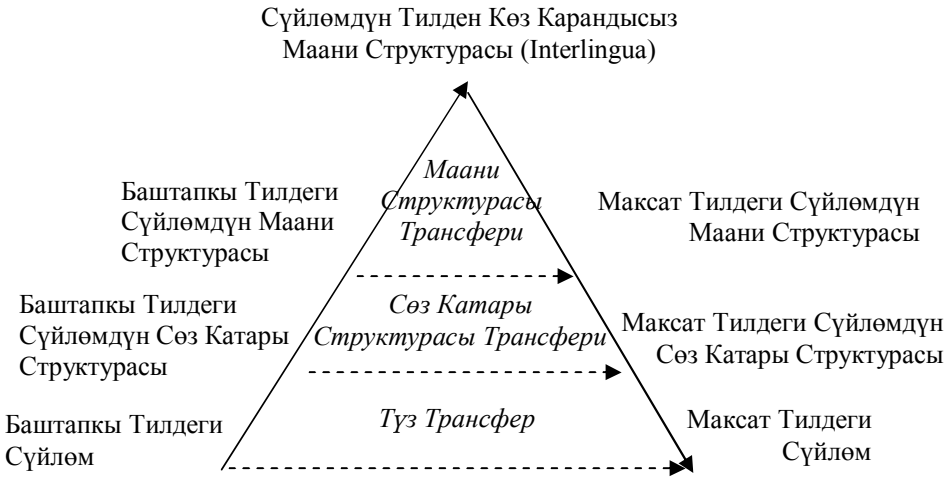
1.3.7 Эмне үчүн автоматташтырылган которуу ушунчалык маанилүү?

Компьютер чарчабайт, 24 саат иштей алат, эс алууну талап кылбайт. Үйрөнгөн эч бир нерсесин унутпайт, жашоо факторлорунан тажрыйба ала албайт, пайда көрүүнү каалабайт, бирок бизге пайда келтирет. Тексттердин тууралыгыны түзө алат, ар бир саат жана ар бир күн берилиштер базасыны кеңейтирип, каалаган убакытта колдоно алат. Которуу жасап жатканда изилденбейт, өзүнүн ой-пикирлерин кошпойт[1].

Ушул себептүү баштапкы тексттин маниси өзгөрбөйт. Дүйнөдө болуп жаткан окуяларды билүү үчүн бир гана клавишке басышыбыз жетиштүү. Каалаган маалыматты жана китепти каалаган убакытта алып окуш мүмкүн, каалаган кишилер менен байланыштарды түзүүгө болот[1].

1.4. КОТОРУУ ЫКМАЛАРЫ

Текстти бир тилден экинчи бир тилге которууда Которуу Пирамидасындагы Сүйлөмдүн Тилден Көз Карандысыз Болгон маани Стуктурасына карата, Сөз Катары Структурасына карата жана Түз которуу ыкмалары менен которса болот. Ал 1-сүрөттө келтирилген.[2]



1 – сүрөт. Которуу Пирамидасы[2]

Бул ыкмалардын иштөө принциби төмөнкүчө:

1.4.1 Тилден Көз Карандысыз Болгон Маани Структурасын Колдонуп Которуу[2]

• Мындай которуу ыкмасы үчүн көп булак керек болот жана буларды табуу же түзүү кымбатка түшөт жана оңой болбогон иш

• Баштапкы сүйлөмдүн тилден көз карандысыз маани структурасын аныктоо

– Баштапкы тил үчүн морфологиялык анализчи

– Баштапкы тил үчүн морфологиялык белгисиздикти жоготуучу

– Максаттык тил үчүн сөз катары структурасы анализчиси

– Баштапкы тил үчүн маанилерди анализдөөчү

– Маани структурасынын тилден көз карандысыз структурага которулушу (онтология деп аталган дүйнө жөнүндөгү билимди камтыган булак талап кылынат)

• Тилден көз карандысыз маани структурасындан максаттык сүйлөмдү түзүү.

– Тилден көз карандысыз маани структурасынын максаттык сүйлөмгө которулушу

1.4.2 Сөз Катары Структурасы Трансфери[2]

• Баштапкы сөйлөмдүн сөз катары структурасы анализ кылынып сөз катарынын структурасынын аныкталышы.

– Баштапкы тил үчүн морфологиялык анализчи

– Баштапкы тил үчүн морфологиялык тил үчүн морфологиялык белгисиздикти жоготуучу

– Баштапкы тил үчүн сөз катары структурасы анализчиси

• Баштапкы сүйлөмдүн сөз катары структурасыны максаттык тилдеги сөз катары структурасына трансфери.

– Структуралар трансфери

– Эки тараптуу сөздүк

• Максаттык тилдеги сөз структурасындан максаттык сүйлөмдү жаратуу

– Сөз катары структурасындан сөздөрдүн кезектеринин аныкталышы

– Морфологиялык жаратуучу

1.4.3 Түз Трансфер[2]

• Түз Трансферде жалаң гана төмөнкү жөнөкөй амал баскычтары колдонулат.

– Баштапкы сүйлөмдүн морфологиялык анализи

– Баштапкы сүйлөм үчүн морфологиялык белгисиздикти жоготуучу колдонулушу

– Баштапкы сүйлөмдүн морфологиялык даражадагы сөздөрдүн максаттык тилдеги морфологиялык даражадагы сөздөргө болгон трансфери

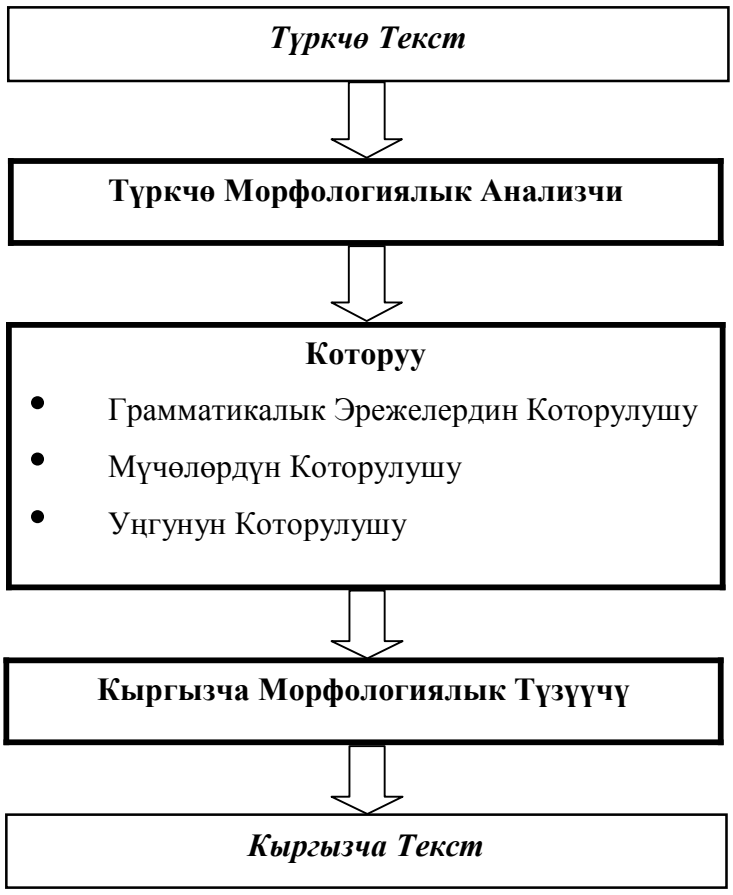
– Максаттык тилдеги сөздөрдүн кезектеринин кайрадан көздөн кечирилиши

– Максаттык тилдеги морфологиялык даражадагы сөздөрдү жаратуу

1.4.4 Тандап алынган которуу ыкмасы

Проектти ишке ашырууда которуунун “Түз трансфер” ыкмасы тандап алынды. Мында берилген тексттеги сүйлөмдөрдүн баштап морфологиялык

анализи жүргүзүлөт. Сүйлөмдөгү сөздөрдүн ар бири бир-бирден алынып ар бир сөз үчүн сөздүн мүчөлөрү жана уңгусу аныкталат. Андан соң бул аныкталган уңгу жана мүчөлөр өзүнчө-өзүнчө экинчи тилге которулат жана бириктирилет. Бул амал тексттин аягына чейин улантылат. Бул которуу ыкмасынын жалпы модели 2-сүрөттө келтирилген.



2 – сүрөт. Түркчөдөн Кыргызчага которуу ыкмасы[2]

ЭКИНЧИ БӨЛҮМ

АЛГОРИТМДИК ЖАНА ПРОГРАММАЛЫК ЖАБДЫКТАРДЫН СТРУКТУРАСЫ

2.1. ББББС MSSQL ДИН КОЛДОНУЛУШУ

Microsoft SQL Server – Microsoft корпорациясы тарабынан иштеп чыгылган, реляцион берилиштер базаларын башкаруучу система. Негизги колдонулуучу сурамжылоо тили – Microsoft жана Sybase биргелешир түзгөн Transact-SQL. Transact-SQL ANSI/ISO стандардынын структурланган сурамжылоо тилинин (SQL) реализациясы болуп эсептелет. Microsoft SQL Server чоң эмес жана орто өчөмдөгү берилиштер базаларынан тартып мекемелердин чоң өлчөмдөгү берилиштер базалары менен иштөө үчүн колдонулат, өз рыногундагы башка берилиштер базасын башкаруу системалары менен атаандашат[3].

2.1.1 MSSQL'дин тарыхы

MS SQL Server (7.0 версиясына чейин) программдык коду Sybase SQL Server дин кодунун негизинде түзүлгөн, ал Microsoft тун Oracle, IBM жана кийинчерээк Sybase дагы атаандашкан берилиштер базасыны өндүрүүчүлөр базарына чыгуусуна шарт түзүп берди. Unix, VMS жана башкалар үчүн чыгарылган Sybase SQL Server 3.0 дун эквиваленти болгон, OS/2 үчүн чыгарылып SQL Server 1.0 (1989 жылдары) деген аталышты алган программанын биринчи версиясын түзүү жана базарга чыгаруу үчүн Microsoft, Sybase жана Ashton-Tate биригишти. 1992 жылы Microsoft SQL Server 4.2 чыгарылды жана Microsoft OS/2 операциондук системасынын 1.3 версиясынын курамына киргизилди. Windows NT операциондук системасы үчүн чыгарылган Microsoft SQL Server дин 4.21 версиясынын расмий чыгарылышы бир эле убакытта Windows NT (3.1 версиясы) нин да чыгарылышы болду. Бир гана NT архитектурасы үчүн жана өндүрүү процессинде Sybase дин катышуусусуз

өндүрүлгөн Microsoft SQL Server 6.0 SQL Server дин биринчи версиясы болду[3].

Windows NT операциондук системасы базарга чыккан убакыттарда, Sybase жана Microsoft бири-биринен ажырады жана өзүлөрүнүн жеке программдык продуктуларынын моделдерин жана маркетинг схемаларын иштеп чыгышты. SQL Server дин Windows үчүн чыгарылган бардык версияларынын укуктарына Microsoft ээ болууга жетишти. Андан кийин Sybase өз продуктусунун атын Microsoft SQL Server менен адаштырылышынан качуу үчүн Adaptive Server Enterprise деп өзгөртүрдү. 1994 жылына чейин Microsoft, Sybase ден Microsoft SQL Server ге өтүүгө эскертүү катары үч билдирүү алды[4].

Бөлүнүштөн кийин компаниялар бир канча өз алдынча программалардын чыгарылыштарын жасашты. SQL Server 7.0 башкаруунун чыныгы колдонуучунун графикалык интерфейсине ээ болгон эң биринчи берилиштер базасы сервери болду. Sybase тарабынан түшкөн автордук укукту бузуу жөнүндөгү айыптоону жокко чыгаруу үчүн жетинчи версиядагы код толугу менен кайрадан жазып чыгылды[3].

SQL Server 2005 версиясы – 2005 жылында сунулду. Версиянын чыгарылышы Visual Studio 2005 тин чыгарылышы менен бирдей убакытка туура келди. Microsoft SQL Server «чектелген» версиясы - Microsoft SQL Server Express дагы бар; ал көчүрүп алууга жана аны колдонгон программалар менен бирге бекер таратууга ачык[4].

SQL Server дин мурунку версияларынын (SQL Server 2000) чыгарылыштан тарта SQL Server 2005 тин курамына кирүүчү иштеп чыгуунун интегрирленген чөйрөсүнүн жана кошумча подсистемалар катарынын өсүшү жолго коюлган. SQL Server Integration Services (SSIS), эскертүү сервери, көп өлчөмдүү берилиштер моделини аналитикалык кайра иштөө чөйрөсү (OLAP) жана релеванттык информацияларды чогултуу (бардык кызматтар Microsoft Analysis Services курамына кирет), ушундай эле бир канча бирдирүү

кызматтары, тагыраак айтканда Service Broker жана Notification Services компоненттеринин курамына кируучу чыгарылыштардын ETL технологиясына өзгөртүүлөр киргизилди. Мындан сырткары, жакшыртуулар жана өндүрүмдүүлүктөргө жетишилди[4].

2.1.2 MSSQL дин функционалдуулугу

Microsoft SQL Server сурамжылоо тили катары SQL дин бир канча кеңейтмелерге ээ болгон SQL-92 (SQL үчүн ISO стандарты) чыгарылышынын Transact-SQL деп аталган версиясын колдонот. T-SQL сакталган процедуралар үчүн кошумча синтаксистерди колдонууну жана транзакцияларды камсыз кылууну мүмкүн кылат. Microsoft SQL Server жана Sybase ASE тармак менен өз ара аракеттенүүнү үчүн Tabular Data Stream (TDS, таблицалык берилиштерди берүү протоколу) деп аталган программалык деңгээлдеги протоколду колдонот. Ушундай эле Microsoft SQL Server жана Sybase берилиштер базалары менен иштөө мүмкүнчүлүгүн камсыз кылган ар кандай программалар менен иштөөнү камсыз кылуу максатында TDS протоколу FreeTDS проектисинде чыгарылган[3].

Ушундай эле Microsoft SQL Server Open Database Connectivity (ODBC) ни камсыз кылат – ББББС менен өз ара аракеттенүүчү программа интерфейси. SQL Server 2005 версиясы SOAP протоколун колдонуп колдонуучу менен веб-сервистери аркылуу байланышууну камсыз кылат. Ал Windows ко арналбаган клиенттик программа менен SQL Server ди кроссплатформа түрүндө байланыштыруу мүмкүнчүлүгүн берет. Microsoft ушундай эле Java нын башкаруусу астында иштеген программаларды (IBM WebSphere сыяктуу) Microsoft SQL Server 2000 жана 2005 менен байланыштуруучу сертификатталган JDBC драйверин чыгарды[3].

SQL Server чагылдырууну жана берилиштер базасын кластеризациялоону камсыз кылат. SQL дин кластер сервери – бул бирде конфигурацияланган серверлер ийкемдүүлүгү; мындай схема иш жүгүнү бир канча серверлер арасында бөлүштүрүүгө жардам берет. Бардык серверлер бир

гана виртуалдык атка ээ, жана берилиштер иш цикли аралыгында машинанын IP адрес кластерлери боюнча бөлүштүрүлөт. Ушундай эле серверлердин кластерлеринин бири бузулуп же болбосо иштебей калганда иш башка серверге которулат[4].

SQL Server берилиштердин керегинен көп көбөйтүлүшүн үч сценарий боюнча камсыз кылат[3]:

Сүрөт: сервердин кабыл алуучуга жибере турган берилиштер базасынын «сүрөт»үн алуу үчүн жүргүзүлөт[3].

Өзгөртүрүлгөн убакыт: берилиштер базасынын бардык өзгөрүүлөрү колдонуучуга үзгүлтүксүз берилет[3].

Башка серверлер менен синхронизациялоо: бир канча берилиштер базаларынын серверлери өз ара синхрондолушат. Бардык берилиштер базаларындагы өзгөрүүлөр бири-биринен көз карандысыз жүрөт, синхрондолгондо болсо берилиштер өз ара бириктирилет. Көбөйтүүнүн бул тиби берилиштер базаларынын арасында айкалышууну камсыз кылууну карайт[3].

SQL Server 2005 .NET Framework ту камсыз кыла тургандай кылып түзүлгөн. Анын жардамы менен баардык китепканалар жыйнагын колдонуп, ушундай эле Common Type System ди (Microsoft .NET Framework то берилиш типтери менен кайрылуу системасы) дагы колдонуп берилиштер базасы процедуралары .NET платформасында жазса болот. .NET Framework башка процесстерден айырмаланып, SQL Server 2005 үчүн жалпы бир система болуп эсептелет, ал SQL Server ди башкарууда Windows тун ички каражаттарын колдонуунун ордуна эстен кошумча орун бөлүп берет жана каражаттарды куруп берет. SQL Server структураларында колдонуу үчүн ресурстарды бөлүштүрүү алгоритмдери өзүнчө оңдолгондуктан бул – Windows тун жалпы алгоритмдерине караганда өндүрүмдүүлүктү көбөйтөт[3].

Microsoft жана башка компаниялар Microsoft SQL Server берилиштер базасын колдонгон бизнес программаларды колдонгон көп сандаган программдык жабдыктарды иштеп чыгууда. Microsoft SQL Server 2005 ушундай эле өз ичине .NET платформасы тилдеринде (мисал үчүн, VB.NET же C#) иштеп чыгылып сакталган процедураларды жана түрдүү программа функцияларын реализациялоого уруксат берүүчү Common Language Runtime (CLR) Microsoft .NET ти камтыйт. Microsoft мурунку продуктулары Microsoft SQL Server ге функционалдык байланышууну алуу үчүн бир гана API лерди колдонот[3].

2.1.3 MSSQL менен иштөө (берилиштер базасынан берилиштерди чыгаруу)

MSSQL'де иштөө үчүн бир кичине мисал алып, ошол боюнча керектүү функцияларды карайбыз.

Керектүү командалар: Create Database – берилиштер базасын түзүү;

Create Table – таблица түзүү;

Insert Into – көрсөтүлгөн таблицага жазылыштарды кошот.

From – кайсы таблица менен иштей тургандыгын билдирет.

Where – сурама түзүүдө шарт коюу үчүн колдонулат.

Order by – берилиштерди берилген талаага жараша сорттойт.

Group by – берилиштерди берилген талаага жараша группалайт.

Эми айтылган мисалды жазууга киришебиз:

```
CREATE DATABASE mydb;
```

CREATE TABLE employees([id] [int] IDENTITY(1,1) NOT NULL, [name] [nvarchar](50) COLLATE Cyrillic_General_CI_AS NULL, [surname] [nvarchar](50) COLLATE Cyrillic_General_CI_AS NULL, [address] [nvarchar](50) COLLATE Cyrillic_General_CI_AS NULL, [name] [position](50) COLLATE Cyrillic_General_CI_AS NULL,) ON [PRIMARY];

INSERT INTO employees VALUES (1,'Bob','Smith','128 Here St, Cityname','Marketing Manager');

INSERT INTO employees VALUES (2,'John','Roberts','45 There St ,Townville','Telephonist');

INSERT INTO employees VALUES (3,'Brad','Johnson','1/34 Nowhere Blvd, Snowston','Doorman');

Жыйынтыкта mydb деп аталган берилиштер базасы жана ал базанын ичине employees (жумушчулар)таблицасы түзүлдү. Бул таблицага тиешелүү жазылыштар киргизилди.

2.2. C# ПРОГРАММАЛОО ТИЛИ

C# Microsoft тун келечекти убада кылган ".NET чөйрөсүнүн пайдубалын түзгөн .Net Фреймворкуну" камсыздоо-өнүктүрүү максатында өндүрүлгөн программалоо тили болуп эсептелет. C# тын эң башкы архитектору болгон Андерс Хэйлсберг, ошол эле убакытта 80 жылдарда эң жакшы жетишкендиктерге жеткен Turbo Pascal ды жазган программалоо генийси. Бул факторлордун негизинде C# бир канча абройлуулар тарабынан кабыл алынган жана дүйнөнүн эң жакшы жетишкендиктерге ээ болгон эки тили, башкача айтканда C ve C++ тун негизинде өндүрүлгөн[5].

C# Windows программаларын түзүү үчүн C++ жана Visual Basic программалоо тилдерине алтернатив түрдө Visual Studio .NET менен бирге келген программалоо тили болуп эсептелет. C#, бизге Visual Studio до бир гана C++ менен жетишибиз мүмкүн болгон бардык жерлерге жетишибиз менен

бирге Visual Basic сыяктуу ыңгайлуу колдонууну сунат. Негизинен C# ка .NET үчүн түзүлгөн .NET код структурасына эң ыңгайлуу тил катары карасак болот. Бул себептен C# .NET фреймворкуну эң оңой жана эң жакшы колдонот. Орто даражадагы программалоо тил болушу болсо, Visual Basic жана C++ тун эң жакшы өзгөчөлүктөрүн камтышы жана толугу менен OOP (Object Oriented Programming – Объектке Багытталган Программалоо) чөйрөсүн сунууну камсыздайт[6].

2.2.1 C# тын тарыхы

1990 жылдардын аягында Microsoft тун программалоо генийси Андерс Хайлсберг C# тилини жаратты. 2000 жылдын ортолорунда C# тын алгачкы версиясы базарга чыкты. C# программалоо тили; C, C++ жана Java менен байланышы бар болгон тил түрүндө жаратылды. Бул бир кокустук эмес. Булар дүйнөнүн эң кеңири колдонулган жана эң көп киши жактырган үч тил болуп эсептелет. Хайлсберг дагы Строуструп жана Гослинг сыяктуу дөңгөлөктү кайра ойлоп табуунун ордуна бар болгон бир тилди өнүктүрүүнү туура тапты[5].

Эгер бул тилдер арасында бир байланыш куруу керек болсо, C# программалоо тили; C тилинин синтаксисини жана C++ тилинин объектке багытталган программалоо ыкмаларыны жана жолдорун алган[5].

C# менен Java ортосундагы байланыш болсо өзүнчө бир абал. C# дагы Java сыяктуу C жана C++ тилдеринден өзгөчөлүктөрдү алган, бирок ушул эле учурда Java сыяктуу платформадан көз карандысыз кодду максат алып долбоорлонгон. Башкача айтканда, C# тили, Java тилинин туундусу эмес. Эң жакшы жагы болсо C, C++, C# же Java тилдеринден бирини жакшы билгендерге, башка бардык тилдерде аларга чоң мүмкүнчүлүктөр сунушталат[5].

Бардык бул башка тилдер менен болгон байланышына карабастан C# өзү менен бирге аябай көп жаңылык жана өзгөчөлүктөрдү алып келген. Буга

чейинки башка ийгиликтүү тилдерде дагы ушундай тренд көрүлгөн. Бул жерде C#: компонентке багытталган тил (component-oriented language) катары каралган. C# компоненттерин жазуу үчүн бир бүтүн болгон камсыздоону камтыйт. Булардын ичинен эң негизгиси; аралаш тилдер болгон чөйрөдө иштей билүү касиети болуп эсептелет[5].

Көңүл буруп каралса, бири-биринен туунду алынган ар бир тил мурунку тилдеги кемчиликти жоготуп жаңы кадам таштаган. C++; C деги объект кемчилигини, Java; C++ тагы платформадан көз карандысыздык касиетини жана эң аягында C# бардыгында болгон өзгөчөлүктөрдү дагы, Java да болбогон биргелешип иштөөнү дагы алган[5].

2.2.2 C# та программалоонун өзгөчөлүктөрү

C# ты .NET фреймворкунан бөлүп караонун ордуна бир бүтүн бөлүк түрүндө караган туура болот. Себеби .NET менен түзүлгөн CLR (Common Language Runtime) – JVM (Java Virtual Machine) ге окшогон жана бир-биринен айырмаланган тилдердин IL (Intermediate Language) ге компиляция кылынып, бирге иштөөсүн камсыздаган структура түрүндө аныктама берсек болот), .NET коддорунун бирге иштөөсүнө шарт түзүп берет. Бул C# болобу, C++ болобу, Visual Basic болобу, программабызды кайсы платформада жазышыбызга карабастан структурабыз дайыма бир эле болот дегенди билдирет[6].

.NET жалпы максаттарга жооп берген Windows платформасы программаларында бир бурулуш жасады. .NET сунган бир канча артыкчылыктар бар; колдонулушу оңой болгон объект модулдоо, Garbage Collection жардамы менен жаратылган объекттерин автоматтык түрдө тазаланышы, SQL жана ADO жардамы менен берилиштерге жетүү, ASP .NET баракчаларынын түзүлүшүнө жараган китепканалар жана буларга окшогон бир канча структура. Бирок, ар бир чөйрө үчүн ал чөйрөнүн өзгөчөлүктөрүнө жетүү албетте белгилүү механизмдерди талап кылат. C# тын эң негизги пайдасы, C# .NET фреймворку үчүн нолдон баштап долбоорлоо жана бул тилдин

синтаксиси жана өзгөчөлүктөрү .NET тин сунган жеңилдиктерден оңой гана пайдалануу үчүн өндүрүлгөн[6].

Мисал бере турган болсок; System.String .NET класс китепканасындагы бир класс жана которуштуруу абалдары, substring дердин оңдолушу жана string туюнтмаларынын кесилиши сыяктуу бир канча ыкмаларга ээ. C# компилятору автоматтык түрдө тааныган өзүнүн саптык ачкычтарына ээ[6].

```
string MyStr1=”Visual”;
string MyStr2=”Studio”;
string MyStr3=”.net”;
string MyStr4=MyStr1+” “+MyStr2+” “+MyStr3;
MyStr4.ToUpper();
//жыйынтыгында MyStr4 кө VISUAL STUDIO .NET маaниси берилет[6].
```

.NET фреймворку, колдонулган ресурстарды бошотууда чоң жетшкендиктерге ээ. Бул жетишкендиктердин сыры managed code ичиндеги өзгөрмөлөрдү стектин ордуна managed heap те сактоосу болуп эсептелет. C# бул сыяктуу амалдарды биздин ордубузга автоматтык түрдө аткарат. Албетте .NET платформасында Managed C++ жана VB.NET дагы бирдей өзгөчөлүктөрдөн C# сыяктуу пайдалана алат. Бул структуранын орундалышы болсо, Visual Basic жана C++ тун структурасында орундалган бир топ өзгөрүүлөр аркылуу мүмкүн болгон. Бул да Visual Basic жана C++ көзү менен каралганда бир жакшыртуу түшүнүгүнө алып келет. Структуралары канчалык жакшыртылса дагы жаңы бир платформа үчүн VB жана C ге мурдатан келген программа өндүрүү тилдери түрүндө кароо туура болот. Бул эки тил бул мааниде .NET платформасында программа өндүрүүдө мурдатан келген адаттар жана синтаксис структураларынын негизинде бүт бойдон .NET үчүн долбоорлонбогон; бул да болсо программа түзүп жатканда C# ка караганда бул программа өндүрүү тилдеринин кемчиликтүү абалга алып келет. Бул себептүү VB.NET синтаксисинин дагы эле толук бойдон объектке багытталган программалоого карата долбоорлонбогонунун тартышылышыны ортого чыгарган[6].

C# болсо алдыбызга тап-таза дептер барагы сыяктуу чыгып жатат. Мунун себеби болсо Microsoft тун C# ты толугу менен .NET тин сызган сызыгынан чыкпагандай кылып өнүктүрүүсүндө. Объектке багытталган программалоо үчүн керек болгон синтаксис дизайны жана программа жазууну жеңилдеткен структурасы C# ты .NET фреймворкунда иштеши үчүн эң жакшы жетишкендиктерге ээ болгон абалга алып келди[6].

C# толугу менен объектке багытталган программалоо тили. Бул өзгөчөлүктөр бир гана C# алып жүргөн өзгөчөлүктөр эмес албетте. C++ да объектке багытталган программалоо структурасына ээ, ошондой эле VB жагы объектке багытталган түрдө жасалган. Буга карабастан C++ жана VB ке карап коюлган маселеге караганда бир канча проблемалар алдыбызга чыгат[6].

C++ ту алып караганда, алдыбызга толугу менен объектке багытталган программалоо үчүн өнүктүрүлгөн жана долбоорлонгон бир программалоо тили чыгат, бирок ал өнүктүрүлгөн убакыттарда объектке багытталган программалоо структурасында коддун кайрадан колдонуу деген түшүнүк жок болчу. Классикалык C++ коду бардык тилдерди камсыздай албайт жана классикалык C++ тун иштеши үчүн туундусуну алышыбыз керек болгон класстын чыныгы программдык кодундан класс наамдарына жетишибиз керек болот. Бир канча убакыттан кийин COM дун негизинде тилдердин ийкемдүүлүгүнө жетишилди жана программалоо тилдери (VB,VJ++,C жана C++) оңой эле COM объекти түрүндө жазылган класстарга жете алышты. Бирок COM үйрөнүүгө татаал болгон структурага ээ деген кемчиликти дайыма алып жүрдү жана толугу менен классикалык объектке багытталган программалоо жолунда туунду алууну камсыздай албады[6].

Башка жактан караганда, VB COM жазуу платформасы камсыздаган жана COM дун татаал структурасыны программа түзүүчүлөрдөн жашырган бир структурага ээ. Бирок бир канча объектке багытталган программалоо профессоналдары айткандай VB чыныгы мааниде объектке багытталган программалоо моделине ээ эмес, себеби туунду алууну камсыздай албайт. C# болсо бардык тараптын артыкчылыктарын бизге сунууда. Толук түрдө туунду

алууну камсыздайт, ушундай эле муну менен бирге .NET фреймворкту программдык код файлдарына жетпей туруп туунду алууну камсыздоо үчүн компиляцияланган абалда болгон китепкана файлдарыны (assembly) орундатат. Мунун негизинде .NET программалоо тарыхында биринчи жолу программалоо тилдери арасында диагонал өтүүлөргө уруксат берген жана толук түрдө кайрадан колдонуу мүмкүнчүлүгүн камсыздаган бир платформа болду[6].

Туунду алуу – бир класстын ичинде башка класстарды негиз алышыбызды камсыздаган объектке багытталган программалоо моделинин бир өзгөчөлүгү болуп эсептелет. Мисал үчүн, стадарт түрдө колдонулган бир меню, тулбар жана статусбар структурасы бар, бир формага керектүү стандарттарды жайгаштырган, башка бир формага да ушул сыяктуу өзгөчөлүктөрдү колдонууну каалаганда, кылынышы керек болгон иш, биринчи жасаган форманы негиз алуу жана жаңы формага туундусун берүү болуп эсептелет. Жаңы форма, туундусу алынган формадагы бардык өзгөчөлүктөрдү камтыган жаңы бир форма түрүндө алдыбызга чыгат. Программалоо жагынан караганда бул амал бизге убакыттан утуу жана кайрадан колдонуу мүмкүнчүлүгүн берген структура болот. C# орто деңгээлдеги программалоо тили болуп эсептелет. Эгер бир тизме жасай турган болсок, C# тын бул тизмедеги орду төмөнкүдөй болот:[6]

Assembly
C,C++
C#,J++
Visual Basic
VBA, Scripting Тилдери

Тизмеде жогорудан ылдый карай канчалык ылдый карай түшкөн сайын тилдердин үйрөнүлүшү ошончолук оңойлошот. Болушунча аз код жазып жана болушунча аз убакыт сарптап өзүбүз каалаган проекттерди орундата алабыз. Ассемблер менен программа жаза турган болсок, мурда көрбөгөндөй перформанс алдыбызга чыгат, бирок ассемблер үйрөнүлүшү жана программа жазуу эң оор платформалардан бири болуп эсептелет. Ошол себептүү

ассемблер тили менен жазылган программалар жалпысынан эң жогору перформансты биринчи орунга койуш керек болгон кээ бир тетик драйверлери, оюн подпрограммалары жана операциондук системалардын жалпы файлдары болуп эсептелет (win.com сыяктуу). GUI (Graphical User Interface) түзүү керек болгондо тандоо албетте Visual Basic болушу керек. GUI лерде перформанс биринчи орунда эмес, себеби бул жерде чектөө фактору колдонуучулар, башкача айтканда колдонуучулардын чычканды бат басышы болуп эсептелет[6].

C# Visual Basic менен C++ арасында болгон боштукту болушунча азайтуу үчүн долбоорлонгон. C# C++ менен жазыла алган жана жазылып жатканда C++ сыяктуу татаал болбогон, бир эле учурда Visual Basic тен да жогору деңгээлде программа жазууну каалаган программисттерди максат алган. C++ жана VB арасындагы боштукту толтурган жана анчалык деле канааттандыраарлык болбогон J++ бул абалга карабастан C# менен көп жактан окшоштуктарга ээ. C# C++ тун синтаксисине көбүрөөк окшогон синтаксиске ээ жана C++ тун бир канча өзгөчөлүгүнү орундата алат, мисал үчүн, inheritence (multiple болушуна карабастан). Ушул себептүү C++ программисттери C# ты оңой эле үйрөнүп ала алышат. Visual Basic программисттеринин алдына жаңы бир синтаксис жана бир канча объектке багытталган түшүнүк чыгат, бирок C# ты үйрөнүү C++ ту үйрөнүүдөн алда канча оңой болгондуктан бат эле үйрөнүп кете алышат. C# тын C++ жана Visual Basic арасындагы бир тил экендигине жакшы бир мисал катары эсти башкарууну көрсөтсө болот. Visual Basic теги жалгыз эсти башкаруу: объекттерге иш бүткөндө object=nothing деш жетиштүү болот, муну кылуу унутулганда ортого чыга турган бир гана жаман нерсе, система ресурстары жана эстин керек болгондон бираз көбүрөөк убакыт колдонулушу болуп эсептелет. Башка жактан караганда, C++ долбоорчуларга кайсы өзгөрмөнүн эсте канча убакыт калышыны аныктоо мүмкүнчүлүгүнү берүү жана муну менен эстен мындан да эффективдүү колдонулушуну камсыз кылып гана калбай, буга мажбур кылат. C++ тагы жазылган программалардын эсти башкаруу каталарына кабылышы жана эсте жоготууларга учуроого жол ачышы Visual Basic те көрүлбөйт. C# эсти башкарууда Visual Basic тин жолуну

жолдойт, башкача айтканда жоопкерчилик тилдин өзүндө. Бирок VB тин терсине C# та программист эсти өзү башкаргысы келсе, жазган кодунун бир бөлүгү unsafe деп аныкталышы мүмкүн жана C++ та болгондой эс үчүн орун бөлүп, андан кийин бир бөлүгүн бошотушу мүмкүн[6].

Ушул убакытка чейинки айтылгандарга баа бере турган болсок, Microsoft тун .NET платформасына ийкемдүү кылып өндүрүлгөн C# ка, Visual Basic же C++ тилдери менен программа түзгөн долбоорчулардын бир структура астында чогулган бир түзүлүш түрүндө карасак болот. Болушунча аз убакыт сарптап, болушунча аз код жазып керегинче күчтүү тиркемелерди түзүү үчүн C# эң жакшы платформа болуп эсептелет[6].

ҮЧҮНЧҮ БӨЛҮМ

АЛГОРИТМДИК ЖАНА ПРОГРАММАЛЫК ЖАБДЫКТАРДЫН СТРУКТУРАСЫ

3.1. БЕРИЛИШТЕР БАЗАСЫН ТҮЗҮҮ

Баштап Translator атында берилиштер базасы түзулдү. Анын ичинде mucholor_kg, mucholor_tr, main жана sozduk таблицалары орун алат.

Translator берилиштер базасын түзүү үчүн CREATE DATABASE `Translator`; командасы берилет.

Translator берилиштер базасы түзүлгөндөн кийин аны актив түргө келтирүү үчүн USE translator; командасы берилет. Мындан кийинки кылына турган иш таблицаларды түзүү.

3.1.1 Кыргызча мүчөлөрдү сактоочу таблицаны түзүү

Кезек mucho_kg таблицасын түзүүдө. Бул таблицада кыргызча мүчөлөр, ар бир мүчөнүн группасы жана бул мүчөлөрдүн варианттарын көрсөтүүчү подгруппасы жөнүндө маалыматтар сакталат.

```
CREATE TABLE [dbo].[mucho_kg](
[id] [int] IDENTITY(1,1) NOT NULL,
[type] [int] NOT NULL,
[lastlettertype] [int] NOT NULL,
[singarmonizm] [int] NOT NULL,
[mucho] [nvarchar](50) COLLATE Cyrillic_General_CI_AS NOT NULL
) ON [PRIMARY]
```

3.1.2 Түркчө мүчөлөрдү сактоочу таблицаны түзүү

mucho_kg таблицасындагыдай эле түркчө мүчөлөр жөнүндөгү маалыматтарды сактоочу mucho_tr таблицасын түзүү керек болот. mucho_kg жана mucho_tr таблицаларынын структуралары бир-бирине окшош болушу керек, бир гана айырмасы mucho_kg кыргызча мүчөлөр жөнүндөгү

маалыматтарды сактаса, mucho_tr түркчө мүчөлөр жөнүндөгү маалыматтарды сактайт.

```
CREATE TABLE [dbo].[mucho_tr](
[id] [int] IDENTITY(1,1) NOT NULL,
[type] [int] NOT NULL,
[lastlettertype] [int] NOT NULL,
[singarmonizm] [int] NOT NULL,
[mucho] [nvarchar](50) COLLATE Cyrillic_General_CI_AS NOT NULL
) ON [PRIMARY]
```

3.1.3 Сөздүк таблицасын түзүү

Dictionary таблицасында болсо, түркчө сөздөрдүн кыргызча котормолору сакталат жана алардын кайсы сөз түркүмүнө тиешелүү экендиги жөнүндөгү маалымат сакталат.

```
CREATE TABLE [dbo].[Dictionary](
[wordID] [int] NOT NULL,
[tr_word] [nvarchar](50) COLLATE Cyrillic_General_CI_AS NULL,
[kg_mean] [nvarchar](50) COLLATE Cyrillic_General_CI_AS NULL,
[part_of_speechID] [int] NULL) ON [PRIMARY];
```

3.1.4 Кыргызчага которулган сөз жана мүчөнү анализдөөчү таблицаны түзүү

Main таблицасында болсо кыргызча сөздөрдүн бүтүшү мүмкүн болгон акыркы эки тамгасынын варианттары сакталган. Бул таблицанын жардамы менен которулган сөзгө кайсы мүчө уланышы табылат.

```
CREATE TABLE [dbo].[main](
[ID] [int] NOT NULL,
[Main] [nvarchar](4) COLLATE Cyrillic_General_CI_AS NULL,
[f1] [tinyint] NULL,
[f2] [tinyint] NULL,
[f3] [tinyint] NULL,
[f4] [tinyint] NULL,
.....
[f96] [tinyint] NULL,
[f97] [tinyint] NULL,
[f98] [tinyint] NULL,
[f99] [tinyint] NULL) ON [PRIMARY]
```

3.2. ПРОГРАМДЫК КОДДУ ТҮЗҮҮ

Түркчө текстти кыргызчага которууда негизди ишти аткарган 5 класс түзүлдү.

3.2.1 Берилиштер базасына байланыш түзүүчү класс

Бул класс керек болгон берилиштер базасына байланыш түзүү үчүн колдонулат. Ушундай эле ал берилиштер базасындагы таблицалар үчүн суроо-талаптарды да жүргүзөт. Тиешелүү таблицалар менен суроо-талаптарды жүргүзүүдө ал таблицалар менен иштеген тиешелүү класстар бул классты колдонуп суроо-талаптарын жүргүзөт. Башкача айтканда бул класс берилиштер базасы сервери менен программа ортосунда ортомчу болуп берет. Бул класс менен иштөө үчүн эң биринчи бул класстын объекти түзүлөт, мында бир кнча конструктордон керектүүсүн тандап алып анын объекти түзүлөт. Андан кийин берилиштер базасына берилүүчү суроо-талап QueryStr(string query) методу чакырылып, аргумент катары SQL суроо-талап берилет. Берилген ал суроо талапты орундатуу үчүн Query() методу чакырылат, бул метод эч кандай аргумент албайт жана бир гана жумушту аткарат, ал жумуш суроо-талапты орундатуу болуп эсептелет. Data(int row, string field) методунун жардамы менен жүргүзүлгөн суроо-талаптын негизинде алынган жыйынтыктын ичинен аргумент катары берилген саптагы белгилүү талаанын маанисин алууго болот. DataCount() методу болсо суроо-талап негизинде алынган жыйынтыкта канча сап бар экенин аныктоо үчүн колдонулат, саптардын саны бүтүн сан түрүндө кайтарылат.

SqlConnectionClass.cs

```
class SqlConnectionClass
{
   private string ConnectionString;
   private SqlConnection Connection;
   private string QueryString;
   private SqlCommand SqlCommand;
   private SqlDataReader reader;
   public object result;
```

```
        public SqlConnectionClass(string DatabaseName)
        {
            ConnectionString = "Initial Catalog=" + DatabaseName + ";Integrated
Security=True
            Connection = new SqlConnection();
            QueryString = "";
            SqlCommand = new SqlCommand();
        }

        public SqlConnectionClass(string DatabaseName,string username, string
password)
        {
            ConnectionString = "Initial Catalog=" + DatabaseName + ";Integrated
Security=True;UserID="+username+";Password="+password;
            Connection = new SqlConnection();
            QueryString = "";
            SqlCommand = new SqlCommand();
        }

        public SqlConnectionClass(string DatabaseName, string datasource)
        {
            ConnectionString = "Data Source=" + datasource + ";Initial Catalog="
+ DatabaseName + ";Integrated Security=True";
            Connection = new SqlConnection();
            QueryString = "";
            SqlCommand = new SqlCommand();
        }

        public SqlConnectionClass(string DatabaseName, string
datasource,string username, string password)
        {
            ConnectionString = "Data Source=" + datasource + ";Initial Catalog="
+ DatabaseName + ";Integrated Security=True;UserID="+username+";Password=" +
password;
            Connection = new SqlConnection();
            QueryString = "";
            SqlCommand = new SqlCommand();
        }

        private void ConnectDB()
        {
            Connection.ConnectionString = ConnectionString;
            Connection.Open();
        }

        public void QueryStr(string query)
```

```
{
   this.QueryString = query;
}

public void Query()
{
   this.ConnectDB();
   this.SqlCommand = new SqlCommand(QueryString, Connection);
   this.reader = this.SqlCommand.ExecuteReader();
   this.reader.Close();
   this.Connection.Close();
}

public void Data(int row, string field)
{
   this.ConnectDB();
   this.SqlCommand = new SqlCommand(QueryString, Connection);
   this.reader = this.SqlCommand.ExecuteReader();
   int i=0, intfield;
   try
   {
      while (this.reader.Read())
      {
         if (i == row)
         {
            intfield = this.reader.GetOrdinal(field);
            this.result = this.reader.GetValue(intfield);
            break;
         }
         i++;
      }

   }
   finally
   {
      this.reader.Close();
      this.Connection.Close();
   }
}

public int DataCount()
{
   int numRows = 0;
   this.ConnectDB();
   this.SqlCommand = new SqlCommand(QueryString, Connection);
   this.reader = this.SqlCommand.ExecuteReader();
   try
```

```
        {
            while (this.reader.Read())
            {
                numRows++;
            }
        }
        finally
        {
            this.reader.Close();
        }
        this.reader.Close();
        this.Connection.Close();
        return numRows;
    }
}
```

3.2.2 Берилиштер базасындагы таблицалар үстүндө амал аткаруучу класстар

Ek, Mucho, Main жана DictionaryTable таблицаларынын иштөө прициптери бири-бирине окшош. Бул класстар тиешелүү таблицалар үстүндө аткарыла турган амалдарды аткаруу үчүн кызмат кылышат. Ekle(...) методу таблицага жаңы жазылыш киргизүү үчүн колдонулат. Update(string alan, string deger) методу таблицадан тандап алынган саптын alan аттуу таласындагы маалыматты deger маанисине өзгөртүрүү амалын аткарат. Sil() методу тандап алынган жазылышты өчүрүү үчүн колдонулат. GetData(string alan) методу тандап алынган саптан alan аттуу талаасындагы маалыматты алуу үчүн колдонулат. GetSayi() методу таблицадагы жазылыштардын санын табуу үчүн суроо-талап жиберет. Бул методдор бардык класстарда окшош. Бул класстардын код бөлүгү төмөнкүчө:

Ek.cs

```
class Ek
    {
        private SqlConnectionClass db;
        private int id;
        public int type;
        public int lastlettertype;
        public int singarmonizm;
```

```
        public string mucho;
        public int sayi;

        public Ek(string databasename)
        {
            this.mucho = null;
            this.type = this.lastlettertype = this.singarmonizm = 0;
            this.db = new SqlConnectionClass(databasename);
        }

        public void Ekle(int type, int lastlettertype, int singarmonizm, string ek)
        {
            db.QueryStr("insert into [mucho_tr]([type], [lastlettertype], [singarmonizm], [trek]) values (" + type + "," + lastlettertype + "," + singarmonizm + ",N'" + ek + "')");
            db.Query();
        }

        private void Update(object value, string field)
        {
            TypeCode typecode = Type.GetTypeCode(value.GetType());
            if (typecode == TypeCode.String)
                value = "'" + value + "'";
            db.QueryStr("update [mucho_tr] set [" + field + "]=" + value + " where [id]=" + this.id);
            db.Query();
        }

        public void Sil()
        {
            db.QueryStr("delete from [mucho_tr] where [id]=" + this.id);
            db.Query();
        }

        private object GetData(string field)
        {
            db.QueryStr("select [" + field + "] from [mucho_tr] where [id]=" + this.id);
            db.Data(0, field);
            return db.result;
        }
        private string GetData1(string field)
        {
            db.QueryStr("select [" + field + "] from [mucho_tr] where [id]=" + this.id);
            db.Data(0, field);
            return db.result.ToString();
```

```
        }

        public void GetCount()
        {
            db.QueryStr("select count(*) from [mucho_tr]");
            this.sayi = db.DataCount();
        }

        public void GetCountByEk()
        {
            db.QueryStr("select * from [mucho_tr] where
[trek]=N'"+this.mucho+"'");
            this.sayi = db.DataCount();
        }

        public void GetCountByType()
        {
            db.QueryStr("select * from [mucho_tr] where [type]=" + this.type);
            this.sayi = db.DataCount();
        }

        public void GetCountByLastlettertype()
        {
            db.QueryStr("select * from [mucho_tr] where [lastlettertype]=" +
this.lastlettertype);
            this.sayi = db.DataCount();
        }

        public void GetCountBySingarmonizm()
        {
            db.QueryStr("select * from [mucho_tr] where [singarmonizm]=" +
this.singarmonizm);
            this.sayi = db.DataCount();
        }

        public void GetCountByTypeAndLastlettertype()
        {
            db.QueryStr("select * from [mucho_tr] where [type]=" + this.type + "
and [lastlettertype]=" + this.lastlettertype);
            this.sayi = db.DataCount();
        }

        public void GetCountByLastlettertypeAndSingarmonizm()
        {
            db.QueryStr("select * from [mucho_tr] where [lastlettertype]=" +
this.lastlettertype + " and [singarmonizm]=" + this.singarmonizm);
            this.sayi = db.DataCount();
```

```
        }

        public void GetCountByTypeAndLastlettertypeAndSingarmonizm()
        {
            db.QueryStr("select * from [mucho_tr] where [type]=" + this.type + "
and [lastlettertype]=" + this.lastlettertype + " and [singarmonizm]=" +
this.singarmonizm);
            this.sayi = db.DataCount();
        }

        public void GetId(int row)
        {
            db.QueryStr("select [id] from [mucho_tr] order by [id] desc");
            db.Data(row, "id");
            this.id = Convert.ToInt32(db.result);
        }

        public void GetIdByType()
        {
            db.QueryStr("select [id] from [mucho_tr] where [type]=" + this.type +
" order by [id] desc");
            db.Data(0, "id");
            this.id = Convert.ToInt32(db.result);
        }

        public void GetIdByLastlettertype()
        {
            db.QueryStr("select [id] from [mucho_tr] where [lastlettertype]=" +
this.lastlettertype + " order by [id] desc");
            db.Data(0, "id");
            this.id = Convert.ToInt32(db.result);
        }

        public void GetIdBySingarmonizm()
        {
            db.QueryStr("select [id] from [mucho_tr] where [singarmonizm]=" +
this.singarmonizm + " order by [id] desc");
            db.Data(0, "id");
            this.id = Convert.ToInt32(db.result);
        }

        public void GetIdByTypeAndLastlettertype()
        {
            db.QueryStr("select [id] from [mucho_tr] where [type]=" + this.type +
" and [lastlettertype]=" + this.lastlettertype + " order by [id] desc");
            db.Data(0, "id");
            this.id = Convert.ToInt32(db.result);
```

```
        }

        public void GetIdByTypeAndSingarmonizm()
        {
            db.QueryStr("select [id] from [mucho_tr] where [type]=" + this.type +
" and [singarmonizm]=" + this.singarmonizm + " order by [id] desc");
            db.Data(0, "id");
            this.id = Convert.ToInt32(db.result);
        }

        public void GetIdByLastlettertypeAndSingarmonizm()
        {
            db.QueryStr("select [id] from [mucho_tr] where [lastlettertype]=" +
this.lastlettertype + " and [singarmonizm]=" + this.singarmonizm + " order by [id]
desc");
            db.Data(0, "id");
            this.id = Convert.ToInt32(db.result);
        }

        public void GetIdByTypeAndLastlettertypeAndSingarmonizm()
        {
            db.QueryStr("select [id] from [mucho_tr] where [Type]=" + this.type
+ " and [lastlettertype]=" + this.lastlettertype + " and [singarmonizm]=" +
this.singarmonizm + " order by [id] desc");
            db.Data(0, "id");
            this.id = Convert.ToInt32(db.result);
        }

        public void GetIdByMucho()
        {
            db.QueryStr("select [id] from [mucho_tr] where [trek]=N'" +
this.mucho + "' order by [id] desc");
            db.Data(0, "id");
            this.id = Convert.ToInt32(db.result);
        }

        public void GetEk()
        {
            this.mucho = this.GetData1("trek");//this.GetData("ek").ToString();
        }

        public void GetEkType()
        {
            this.type = Convert.ToInt32(this.GetData("type"));
        }

        public void GetLastlettertype()
```

```
        {
            this.lastlettertype = Convert.ToInt32(this.GetData("lastlettertype"));
        }

        public void GetSingarmonizm()
        {
            this.singarmonizm =
Convert.ToInt32(this.GetData("singarmonizm"));
        }

        public void UpdateMucho(string value)
        {
            this.Update(value, "ek");
        }

        public void UpdateType(string value)
        {
            this.Update(value, "type");
        }

       public void UpdateLastlettertype(string value)
        {
            this.Update(value, "lastlettertype");
        }

        public void UpdateSingarmonizm(string value)
        {
            this.Update(value, "singarmonizm");
        }
    }
```

Mucho.cs

```
class Mucho
    {
        private SqlConnectionClass db;
        private int id;
        public int type;
        public int lastlettertype;
        public int singarmonizm;
        public string mucho;
        public int sayi;

        public Mucho(string databasename)
        {
            this.mucho = null;
            this.type = this.lastlettertype = this.singarmonizm = 0;
```

```
            this.db = new SqlConnectionClass(databasename);
        }

        public void Ekle(int type, int lastlettertype, int singarmonizm, string
mucho)
        {
            db.QueryStr("insert into [mucho_kg]([type], [lastlettertype],
[singarmonizm], [mucho]) values (" + type + "," + lastlettertype + "," +
singarmonizm + ",N'" + mucho + "')");
            db.Query();
        }

        private void UpdateMucho(object value, string field)
        {
            TypeCode typecode = Type.GetTypeCode(value.GetType());
            if (typecode == TypeCode.String)
                value = "'" + value + "'";
            db.QueryStr("update [mucho_kg] set [" + field + "]=" + value + "
where [id]=" + this.id);
            db.Query();
        }

        public void Sil()
        {
            db.QueryStr("delete from [mucho_kg] where [id]=" + this.id);
            db.Query();
        }

        private object GetData(string field)
        {
            db.QueryStr("select [" + field + "] from [mucho_kg] where [id]=" +
this.id);
            db.Data(0, field);
            return db.result;
        }

        public void GetCount()
        {
            db.QueryStr("select * from [mucho_kg]");
            this.sayi = db.DataCount();
        }

        public void GetCountByType()
        {
            db.QueryStr("select * from [mucho_kg] where [type]=" + this.type);
            this.sayi = db.DataCount();
        }
```

```
        public void GetCountByLastlettertype()
        {
            db.QueryStr("select * from [mucho_kg] where [lastlettertype]=" +
this.lastlettertype);
            this.sayi = db.DataCount();
        }

        public void GetCountBySingarmonizm()
        {
            db.QueryStr("select * from [mucho_kg] where [singarmonizm]=" +
this.singarmonizm);
            this.sayi = db.DataCount();
        }

        public void GetCountByTypeAndLastlettertype()
        {
            db.QueryStr("select * from [mucho_kg] where [type]=" + this.type + "
and [lastlettertype]=" + this.lastlettertype);
            this.sayi = db.DataCount();
        }

        public void GetCountByLastlettertypeAndSingarmonizm()
        {
            db.QueryStr("select * from [mucho_kg] where [lastlettertype]=" +
this.lastlettertype + " and [singarmonizm]=" + this.singarmonizm);
            this.sayi = db.DataCount();
        }

        public void GetCountByTypeAndLastlettertypeAndSingarmonizm()
        {
            db.QueryStr("select * from [mucho_kg] where [type]=" + this.type + "
and [lastlettertype]=" + this.lastlettertype + " and [singarmonizm]=" +
this.singarmonizm);
            this.sayi = db.DataCount();
        }

        public void GetId(int row)
        {
            db.QueryStr("select [id] from [mucho_kg] order by [id] desc");
            db.Data(row, "id");
            this.id = Convert.ToInt32(db.result);
        }

        public void GetIdByType()
        {
```

```
            db.QueryStr("select [id] from [mucho_kg] where [type]=" + this.type
+ " order by [id] desc");
            db.Data(0, "id");
            this.id = Convert.ToInt32(db.result);
        }

        public void GetIdByLastlettertype()
        {
            db.QueryStr("select [id] from [mucho_kg] where [lastlettertype]=" +
this.lastlettertype + " order by [id] desc");
            db.Data(0, "id");
            this.id = Convert.ToInt32(db.result);
        }

        public void GetIdBySingarmonizm()
        {
            db.QueryStr("select [id] from [mucho_kg] where [singarmonizm]=" +
this.singarmonizm + " order by [id] desc");
            db.Data(0, "id");
            this.id = Convert.ToInt32(db.result);
        }

        public void GetIdByTypeAndLastlettertype()
        {
            db.QueryStr("select [id] from [mucho_kg] where [type]=" + this.type
+ " and [lastlettertype]=" + this.lastlettertype + " order by [id] desc");
            db.Data(0, "id");
            this.id = Convert.ToInt32(db.result);
        }

        public void GetIdByTypeAndSingarmonizm()
        {
            db.QueryStr("select [id] from [mucho_kg] where [type]=" + this.type
+ " and [singarmonizm]=" + this.singarmonizm + " order by [id] desc");
            db.Data(0, "id");
            this.id = Convert.ToInt32(db.result);
        }

        public void GetIdByLastlettertypeAndSingarmonizm()
        {
            db.QueryStr("select [id] from [mucho_kg] where [lastlettertype]=" +
this.lastlettertype + " and [singarmonizm]=" + this.singarmonizm + " order by [id]
desc");
            db.Data(0, "id");
            this.id = Convert.ToInt32(db.result);
        }
```

```
        public void GetIdByTypeAndLastlettertypeAndSingarmonizm()
        {
            db.QueryStr("select [id] from [mucho_kg] where [Type]=" + this.type
+ " and [lastlettertype]=" + this.lastlettertype + " and [singarmonizm]=" +
this.singarmonizm + " order by [id] desc");
            db.Data(0, "id");
            this.id = Convert.ToInt32(db.result);
        }

        public void GetIdByMucho()
        {
            db.QueryStr("select [id] from [mucho_kg] where [mucho]=N'" +
this.type + "' order by [id] desc");
            db.Data(0, "id");
            this.id = Convert.ToInt32(db.result);
        }

        public void GetMucho()
        {
            if (this.GetData("mucho") == null)
                this.mucho = "";
            else
            this.mucho = this.GetData("mucho").ToString();
        }

        public void GetMuchoType()
        {
            this.type = Convert.ToInt32(this.GetData("type"));
        }

        public void GetLastlettertype()
        {
            this.lastlettertype = Convert.ToInt32(this.GetData("lastlettertype"));
        }

        public void GetSingarmonizm()
        {
            this.singarmonizm                                                =
Convert.ToInt32(this.GetData("singarmonizm"));
        }

        public void UpdateMucho(string value)
        {
            this.UpdateMucho(value, "mucho");
        }

        public void UpdateType(string value)
```

```
        {
            this.UpdateMucho(value, "type");
        }

        public void UpdateLastlettertype(string value)
        {
            this.UpdateMucho(value, "lastlettertype");
        }

        public void UpdateSingarmonizm(string value)
        {
            this.UpdateMucho(value, "singarmonizm");
        }
    }
```

DictionaryTable.cs

```
class DictionaryTable
    {
        private int id;
        private string trWord;
        private string kgWord;
        private int type;
        private SqlConnectionClass connection;
        public int numRecords;

        public DictionaryTable() {
            this.connection = new SqlConnectionClass("translator");
            this.id = 0;
            this.trWord = "";
            this.kgWord = "";
            this.type = 0;
            this.numRecords = 0;
        }

        public void Insert(string trWord, string gkgWord, int type)
        {
            this.connection.QueryStr("insert into
Dictionary(trWord,kgWord,type)values(N'"+trWord+"',N'"+kgWord+"',"+type+")");
            this.connection.Query();
        }
        public void Delete()
        {
            this.connection.QueryStr("delete from Dictionary where id=" +
this.id);
            this.connection.Query();
        }
```

```
        private void Update(string field, int data)
        {
            this.connection.QueryStr("update Dictionary set " + field + "=" +
data);
            this.connection.Query();
        }
        private void Update(string field, string data)
        {
            this.connection.QueryStr("update Dictionary set " + field + "=N'" +
data + "'");
            this.connection.Query();
        }
        private object GetData(int row, string field)
        {
            this.connection.QueryStr("select * from Dictionary where id=" +
this.id);
            this.connection.Data(row, field);
            return this.connection.result;
        }
        public int GetIdValue(int row)
        {
            this.connection.QueryStr("select id from Dictionary");
            this.connection.Data(row,"id");
            this.id = int.Parse((this.connection.result.ToString()));
            return this.id;
        }
        public int GetIdByTrWordValue(int row)
        {
            this.connection.QueryStr("select id from Dictionary where
'trWord'=N'"+this.trWord+"' order by id desc");
            this.connection.Data(row, "id");
            this.id = int.Parse(this.connection.result.ToString());
            return this.id;
        }
        public int GetIdByKgWordValue(int row)
        {
            this.connection.QueryStr("select id from Dictionary where
'kgWord'=N'" + this.kgWord + "' order by id desc");
            this.connection.Data(row, "id");
            this.id = int.Parse((this.connection.result.ToString()));
            return this.id;
        }
        public string GetTrWordValue()
        {
            this.trWord=this.GetData(0,"trWord").ToString();
            return this.trWord;
        }
```

```
    public string GetKgWordValue()
    {
       this.kgWord=this.GetData(0,"kgWord").ToString();
       return this.kgWord;
    }
    public string GetTypeValue()
    {
       this.type=int.Parse(this.GetData(0,"type").ToString());
       return this.trWord;
    }
}
```

3.2.3 Түркчө текстти Кыргызчага которуучу класс

Которуу ишин толугу менен бул класс аткарат. Которуу учуруда бул класс жогоруда берилген класстардын бардыгын колдонот. Проекттин эң негизги кыймылдаткычы десек да жаңылышпаган болобуз. Бул классты колдонуу үчүн эң биринчи бул класстын объекти түзүлөт, объектти түзүүдө Translate() же Translate(string tr_text) конструкторлордун биринин жардамы менен ал объектти биринчи колдонууго даяр абалга алып келинет. Андан кийин эгер объект Translate() конструкторунун жардамы менен колдонууга даярдалган болсо, анда SetInputText(string input_text) методуну чакырып түркчө киргизилген текст которууга берилет, бул метод ала турган аргумент ушул түркчө текст болот. Андан кийин TranslateTrText() методу чакырылат. Бул метод эң биринчи алынган түркчө тексттеги сөздөрдү пунктуациялык жана башка белгилерден ажыратат (арасына бир боштук коюу менен), бул амал SplitPunctuationsAndWordsInString() методуну чакыруу менен аткарылат. Сөздөр пунктуациялык жана башка белгилерден ажыратылгандан кийин SplitInputText() методу чакырылат. Бул методдун жардамы менен сөздөр, пунктуациялык белгилер жана башка белгилер бир массивге салып бөлүнөт, бөлүүдө боштук негиз кылып алынган. Эки боштуктун арасындагы ар бир сөз же пунктуациялык белгилер же башка белгилер бир сөз деп кабыл алынып массивге салына берет. Бөлүү иши аяктагандан кийин сөздөр салынган массивдеги сөздөрдү которууга өтүлөт. Мында эң биринчи сөздүк жана жана түркчө мүчөлөрдү колдонуп сөздүн уңгусу жана мүчөсү аныкталат. Уңгунун жана мүчө кыргызчага которулат. Уңгуну сөздүктүн жардамы менен

которулат. Мүчөнү которууда болсо түркчө жана кыргызча мүчөлөр сакталган таблицалардагы берилиштерге карап которулат. Эң биринчи түркчө мүчөнүн группасы аныкталат, анын группасы бүтүн сан түрүндө берилген болот. Кыргызча мүчөлөр сакталган таблицадан группасы түркчө мүчөнүн группасында болгон мүчө тандап алынат. Кыргызча мүчөнү алганда группасы бирдей болгон бир канча мүчө алдыбызга чыгат. Мындан кийинки маселе алардын ичинен туурасын тандап алуу. Туура болгон мүчөнү тандап алуу үчүн которулган кыргызча сөз жана тамга айкалыштары сакталган таблица колдонулат. Эң биринчи бул кыргызча сөздүн акыркы эки тамгасы алынат жана ал тамга айкалыштары сакталган таблицадан изделет. Жыйынтыкта бир сап берилет. Ал саптан талаа аты мүчөнүн группасынын номериндей болгон талаадагы сандык маани алынат. Алынган ал маани менен подгруппасы бирдей болгон кыргызча мүчө тандап алынат. Андан кийин которулган кыргызча сөз менен кыргызча мүчө бири-бирине уланат, кыргызча сөздөрдү сактаган массиге салынат. Эгер сөздүк сакталган таблицадан бир сөздүн уңгусун таба албаса анда бул сөз сөздүктө жок деп эсептелет да, кандай болсо ошондой кыргызча сөздөрдү сактаган массивге салынат. Пунктуациялык жана башка сөз эмес болгон белгилер да түздөн-түз, которулбай кыргызча сөздөр салынган массивге салынат. Бардык сөз которулуп бүткөндөн кийин массив бириктирилет, ал MergeTextForKGOutput() методунун жардамы менен аткарылат. MergePunctuationsAndWordsInString() методу чакырылат. Бул метод сөздөр менен пунктуациялык жана башка белгилердин арасындагы керексиз бошлуктарды жоготуу үчүн колдонулат. Эң аягында кыргызча котормону алуу үчүн GetOutputText() методу чакырылат. Ал берген жыйынтык түркчө берилген тексттин кыргызчага котормосу болуп эсептелет.

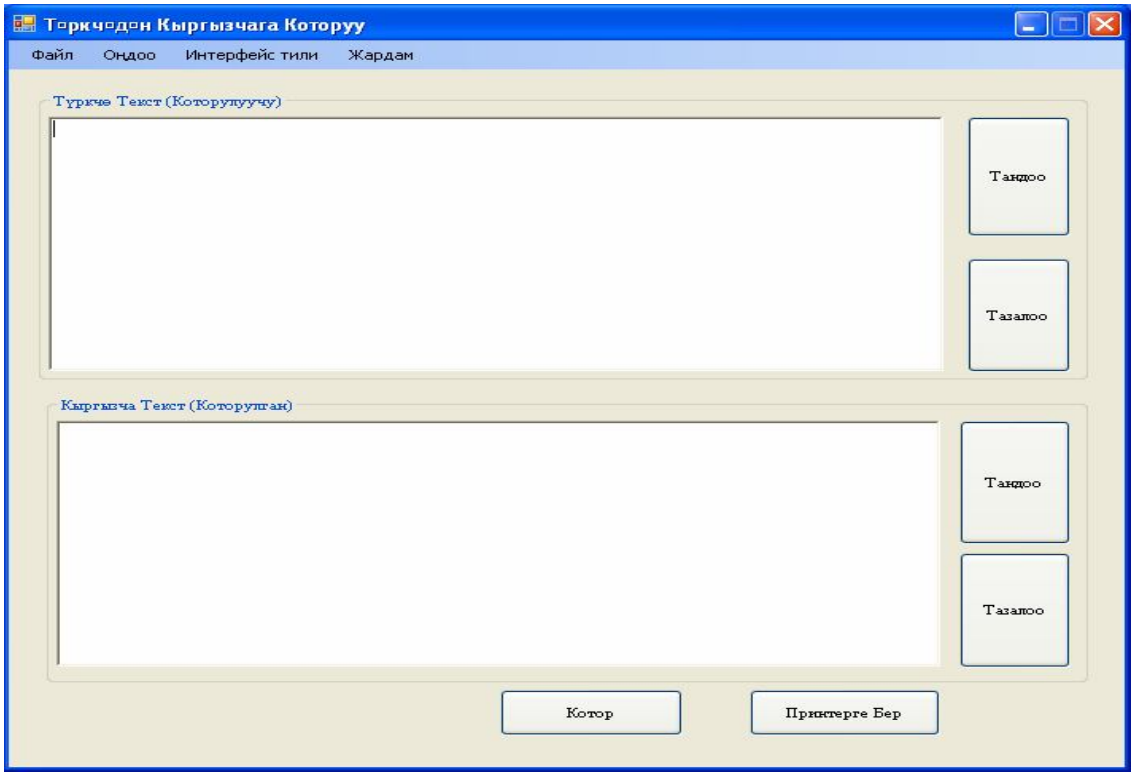
Translate.cs

```
class Translate
 {
    private string input_text;
    private string output_text;
```

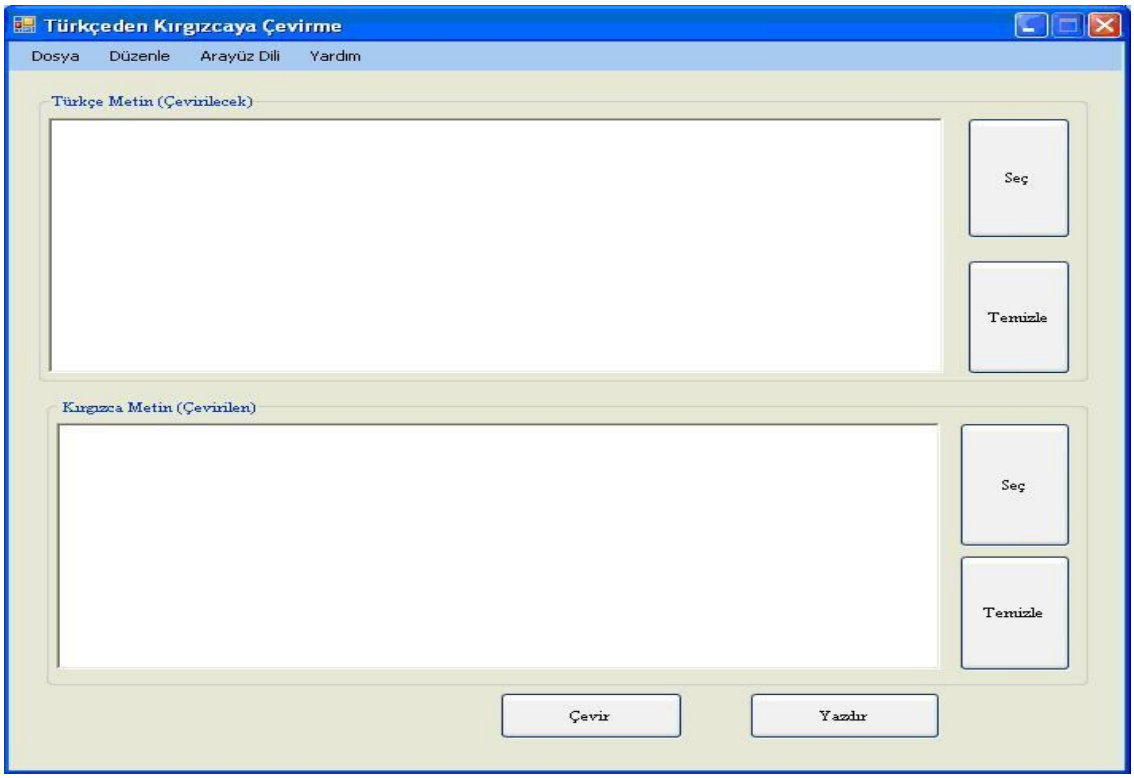
```
    private string[] splitted_tr_words;
    private int splitted_tr_word_num;
    private string[] splitted_kg_words;
    private int splitted_kg_word_num;
    private string suffix_kg;
    private string suffix_tr;

    public Translate() ;
    public Translate(string tr_text);
    public void SetInputText(string input_text);
    public string GetOutputText();
    public void TranslateTrText();
    private void SplitPunctuationsAndWordsInString();
    private void MergePunctuationsAndWordsInString();
    private void SplitInputText();
    private void MergeTextForKGOutput();
}
```

3.3. ПРОГРАМДЫК ЖАБДЫКТЫ КАНТИП КОЛДОНУУ КЕРЕК?

Программдык жабдык интерфейси кыргызча жана түркчө болуп, эки тилде даярдалды. Эки интерфейс тең бирдей эле жумуш аткарат, болгон айырмачылыгы менюлар жана баскычтар биринде кыргызча болсо, биринде түркчө. Кыргызча жана түркчө интерфейстер 3 жана 4 сүрөттөрдө келтирилген.



3-сүрөт. Программдык жабдыктын жалпы көрүнүшү. Кыргызча интерфейс.



4-сүрөт. Программдык жабдыктын жалпы көрүнүшү. Түркчө интерфейс.

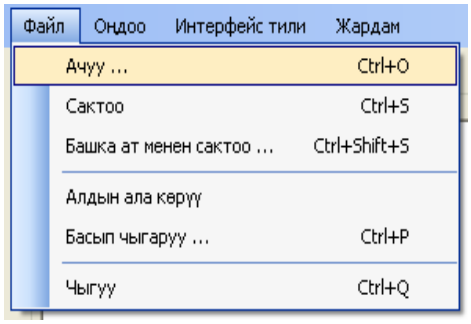
3.3.1 Менюлар

Программдык жабдыкта төрт меню бар. Алар Файл (Dosya), Оңдоо (Düzenle), Интерфейс Тили (Arayüz Dili) жана Жардам (Yardım).

3.3.1.1 Файл (Dosya) менюсу

Бул меню өз ичине Ачуу... (Aç...), Сактоо (Kaydet), Башка ат менен сактоо... (Farklı Kaydet...), Алдын ала көрүү (Önizleme), Басып чыгаруу (Yazdır...) жана Чыгуу (Çık) деген алты команданы камтыйт. Ачуу... (Aç...) командасын колдонуп компьютерде бар болгон бир текстти колдонууга мүмкүнчүлүк алса болот. Бул команда менен алынган файлдын ичиндеги текст автоматтык түрдө которуулуучу жерге салынат. Сактоо (Kaydet) жана Башка ат менен сактоо... (Farklı Kaydet...) командалары которулган текстти компьютерге сактоого мүмкүнчүлүк берет. Алдын ала көрүү (Önizleme) командасы болсо, принтерге басып чыгарууга бериле турган текст бараста кандай көрүнүштө жайгашышын көрүүгө мүмкүнчүлүк берет. Басып чыгаруу (Yazdır...) командасынын жардамы менен которулган текстти принтерге берип басып чыгаруу мүмкүн. Ал үчүн албетте принтер компьютерге уланган жана таанытылган болушу керек. Чыгуу (Çık) командасы болсо, программдык жабдыктан чыгуу мүмкүнчүлүгүн сунат. Бул менюнун кыргызча жана түркчө варианттары 5-сүрөттө келтирилген.

А)



Б)

5-сүрөт. Файл (Dosya) менюсу. А-Кыргызча жана Б-Түркчө интерфейстеги

3.3.1.2 Оңдоо (Düzenle) менюсу

Бул меню өз ичине Артка алуу (Geri), Алдыга алуу (İleri), Кесүү (Kes), Копиялоо (Kopyala), Коюу (Yapıştır) жана Бардыгын тандоо (Tümünü Seç) деген алты команданы камтыйт. Артка алуу (Geri) жана Алдыга алуу (İleri) командалары менен жасалган бир аракет артка же алдыга алынат. Кесүү (Kes) жана Копиялоо (Kopyala) командаларынын жардамы менен тандалган аянт тиешелүү түрдө кесүү же копиялоо үчүн эске убактылуу сактап турулат. Коюу (Yapıştır) командасы болсо, Кесүү (Kes) же Копиялоо (Kopyala) командаларынын жардамы менен эске убактылуу сакталган маалымат курсор турган жерге коюлат. Бардыгын тандоо (Tümünü Seç) командасы курсор турган объекттин ичиндеги тексттин бардыгын тандоо мүмкүнчүлүгүн берет. Бул менюнун кыргызча жана түркчө варианттары 6-сүрөттө келтирилген.

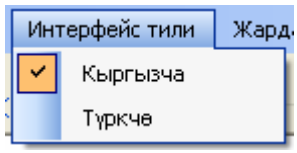
А)

Б)

6-сүрөт. Оңдоо (Düzenle) менюсу. А-Кыргызча жана Б-Түркчө интерфейстеги.

3.3.1.3 Интерфейс Тили (Arayüz Dili) менюсу

Бул меню өз ичине Кыргызча (Kırgızca) жана Түркчө (Türkçe) деген эки команданы камтыйт. Ал командалардын жардамы менен интерфейс тили кыргызча же түркчөгө өзгөртүлөт. Программдык жабдык иштеп баштаганда интерфейс тили кыргызча болот. Бел менюнун кыргызча жана түркчө варианттары 7-сүрөттө келтирилген.

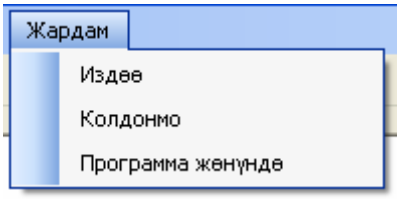
А) 

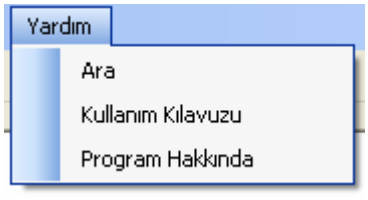
Б) 

7-сүрөт. Интерфейс тили (Arayüz Dili) менюсу. А-Кыргызча жана Б-Түркчө интерфейстеги.

3.3.1.4 Жардам (Yardım) менюсу

Бул меню өз ичине Издөө (Ara), Колдонмо (Kullanım Kılavuzu) жана Программа жөнүндө (Program Hakkında) деген командаларды камтыйт. Издөө (Ara) көмандасынын жардамы менен колдонмо китепчесинин ичинен бир сөз же сөздөр бирикмесин табууга болот. Колдонмо (Kullanım Kılavuzu) командасы чакырылганда, колдонуучуга программдык жабдыкты колдонуу жоболору камтылган китепчени экранга чыгарып берет. Программа жөнүндө (Program Hakkında) командасы болсо Программдык жабдык жөнүндө кыскача маалымат берет. Ал маалымат өз ичине программдык жабдыктын версиясы, жасалган жылы, автору жана байланыш адресин камтыйт. Бул менюнун кыргызча жана түркчө варианттары 8- сүрөттө келтирилген.

А) 

Б) 

8-сүрөт. Жардам (Yardım) менюсу. А-Кыргызча жана Б-Түркчө интерфейстеги.

3.3.2 Текстти которуу

Которула турган түркчө текст “Түркчө Текст (Которулуучу)” (Türkçe Metin (Çevirilecek)) деген жерден киргизилет, андан кийин ылдый жактагы “Котор” (Çevir) баскычына басып, кыргызчага которулган жыйынтыгы “Кыргызча Текст (Которулган)” (Kırgızca Metin (Çevirilen)) деген жерден алынат.

ЖЫЙЫНТЫК

Бул проект түркчөдөн кыргызчага текст которгон программа болуп эсептелет. Проект компьютер болгон бардык жерлерде колдонууга арналган программдык жабдык болуп эсептелет. Бул проекттин аткарылышында максатка жетүү үчүн аткарылган кадамдар, колдонулган программдык жабдыктар, берилиштер базасынын структурасы, иштөө ыкмалары жана методологиялары тууралуу түшүнүктөр аныкталып баяндалды. Проект учурдагы программдык жабдыктардын мүмкүнчүлүктөрүн колдонуу менен түзүлдү. Колдонуучуларга киргизилген түркчө тексттерди кыргызчага которуп берүү башкы максат катары алынды.

Программдык жадбыктын интерфейси колдонуу үчүн маанилүү болуп эсептелет. Программдык жабдыктын интерфейси колдонууга татаал кылынып жасалса бат эле колдонуудан чыгып калышы мүмкүн экендиги эске алынып колдонууга оңой жана жөнөкөй кылып жасалды.

Жасалган программдык жабдык жөнүндө маалымат берилди. Программдык жабдык кандай бөлүктөрдөн тураары, иштөө принциби, берилиштер базасыдагы таблицалар, берилиштер базасы менен кандай иштер жүргүзүлүшү жана бул берилиштер базасында кандай маалыматтар сакталышы каралды.

Бул проект C# тилин жана берилиштер базасы катары MSSQL берилиштер базасын колдонуп жасалды.

ÖZET

Yüksek lisans tezi projesi Türkçeden Kırgızcaya metin çevirisi yapan yazılımdır. Bu bilgisayarın olduğu tüm yerde kullanılması düşünülen yazılımdır. Bu yüksek lisans tezi projesi yapılırken amaca ulaşmak için gerçekleştirilen işler, kullanılam yazılımlar, veritabanı yapıları, çalışma prensipleri ve yöntemleri hakkındaki düşünceler belirlenip açıklandı. Proje günümüzdeki yazılımların imkanlarından faydalanılarak gerçekleştirişdi. Kullanıcıya verilen Türkçe metni Kırgızcaya çevirerek sunmak en önemli amac olarak belirlendi.

Yazılım arayüzü kullanım için tasarımı kullanım için çok önemlidir. Eğer arayüz kullanım için zor olarak tasarlanmışsa bu yazılım çok çabuk kullanımdan çıkar, kullanıcı kitlesini kaybeder. Bu yüzden yazılım arayüzü kullanım için çok basit bir şekilde hazırlandı.

Yapılan yazılım hakkında bilgi verildi. Yazılımın hangi parçalardan oluşması, çalışma şekli, veritabanındaki tablolar, varitabanıyla ne tür işlemler gerçekleştirilmasi ve bu veritabanı ne tür verileri içermesi üzerinde çalışıldı.

Yüksek lisans projesi C# programlama dili ve veritabanı için MSSQL veritabanını kullanarak gerçekleştirildi.

КОЛДОНУЛГАН АДАБИЯТТАРДЫН ТИЗМЕСИ

1. Machine Translation – MT. http://www.sametran.com/index.php?content=genelbakis

2. Bilgisayarlı Çeviri. http://www.tdk.gov.tr/TR/dosyagoster.aspx?DIL=1& BELGEANAH=2908& DOSYAISIM=sunum11.ppt

3. Microsoft SQL Server. http://ru.wikipedia.org/wiki/Microsoft_SQL_Server

4. Microsoft SQL Server. http://en.wikipedia.org/wiki/Microsoft_SQL_Server

5. C# programlama dili, tarihi, doğuşu. http://progenc.blogcu.com/c-programlama-dili-tarihi-dogusu/2606425

6. C# (C Sharp)'a Genel Bakış. http://www.internetdergisi.com/index.php?Part=Article&id=259

АВТОБИОГРАФИЯ

Туулган жери жана жылы :	Өзбекстан		1984
Билим алган мекемелер :	**Баштоо**	**Бүтүрүү**	**Мекеменин аты**
Мектеп :	1991	1993	Мелибоев атындагы 3-орто мектеп. Өзбекстан, Жызак обл, Зарбдар рай.
Мектеп :	1993	1997	Токтогул атындагы 44-орто мектеп. Өзбекстан, Жызак обл., Заамин рай.
Лицей :	1997	2000	Өзбек-Түрк лицейи, Өзбекстан, Жызак ш.
Мектеп :	2000	2001	Токтогул атындагы 44-орто мектеп. Өзбекстан, Жызак обл., Заамин рай.
Бакалавр :	2004	2008	Кыргыз-Түрк Манас Университети
Магистратура :	2008	-	Кыргыз-Түрк Манас Университети
Үй-бүлөлүк абалы :	Бойдок		
Билген чет тилдери: :	Түркчө		Эң жакшы
	Англисче		Жашкы
	Орусча		Жакшы
	Өзбекче		Эң жакшы
Иштеген мекеме (лер) :	**Ишке баштоо жана кетүү даталары**		**Мекеменин аты**
1.	05.2003	06.2004	Жызак Мамлекеттик Педагогикалык Институту, Өзбекстан, Жызак ш.
2.	11.2008	-	Кыргыз-Турк Манас Университети
Катышкан илимий конференциялар, семинарлар :	Исхак Раззаковдун 100 жылдыгына арналган жаш окумуштуулардын жана студенттердин 52-илимий техникалык талкуусу. Студенттик илим: Жаштардын көз карашы		

18.05.2010

Бахориддин Душабаев

КТМУ Т.И.И.
КОМПЬЮТЕР ИНЖЕНЕРИЯСЫ БАГЫТЫ

ТҮРК ТИЛИНЕН КЫРГЫЗ ТИЛИНЕ ТЕКСТ КОТОРУУ ПРОГРАММАСЫ
(МАГИСТРДИК ДИССЕРТАЦИЯ)

Бахориддин ДУШАБАЕВ

БИШКЕК
2010

КЫРГЫЗ-ТҮРК МАНАС УНИВЕРСИТЕТИ
ТАБИГЫЙ ИЛИМДЕР ИНСТИТУТУ
КОМПЬЮТЕР ИНЖЕНЕРИЯСЫ БАГЫТЫ

ТҮРК ТИЛИНЕН КЫРГЫЗ ТИЛИНЕ ТЕКСТ КОТОРУУ ПРОГРАММАСЫ

(МАГИСТРДИК ДИССЕРТАЦИЯ)

Бахориддин ДУШАБАЕВ

БИШКЕК 2010