Софийски университет "СВ.Климент Охридски"

Факултет по Математика и Информатика

Катедра: Софтуерни технологии

ДИПЛОМНА РАБОТА

ТЕМА:

Проектиране на система за защита на Бази данни за ресурсно планиране чрез пароли и различни нива на достъп.

на
Ася Огнянова Григорова

Направление: Информатика

Магистърска програма: Защита на информацията в компютърните системи и мрежи

Ф.№ 22167

Ръководител: Проф.д-р Нина Синягина

София
2008

Използвани съкращения

А

Автентикация (authentication) – процес на проверка, дали даден потребител е този, за който се представя.

Авторизация (authorization)- процес на проверка, дали даден потребител има право да извърши дадено действие.

Б

База Данни (БД) - организирана колекция от данни, използвана с цел моделиране на някакъв вид организация или организационен процес.

Д

Деловодство – служба, която следи движението на документите в компания. Извършва справки за докумнети, в зависимост от постъпилите данни.

Документооборот – движение на входящите и изходящите документи, проследяване на техния статус от въвеждане до архив.

Доставчик на данни (Data Providers) – съвкупност от класове, които осъществяват връзка с различни бази данни. Осигуряват възможност да се изпълняват команди и да се получават резултатите по начин, който е независим от източника на данни и неговата специфична функционалност.

И

Информационна система (ИС)- система, предназначена за съхраняване, предаване и обработка на данни, с цел получаване на необходимата за потребителя информация.

М

Моделиране на данни – процес на прехвърляне на нещата от реалния в компютърния свят.

О

Операционна система (ОС) – комплекс от управляващи и обслужващи програми, осигуряващи максимална ефективност на работата на компютъра , чрез автоматично управление на информационните процеси и ресурси.

П

Предметна област (ПО) – част от реалния свят , данните за която е необходимо да се съхраняват в базата данни.

С

Система за управление на Бази данни (СУБД) – програмно осигуряване, което създава базата данни и осигурява достъпа до данните се нарича система за управление на бази данни.

Неформални понятия	**Формални понятия**
Table/Таблица	Relation/Релация
Row/Ред	N-орка/Ред
Column/Колона	Attribute/Атрибут
МДЗ (Множество от допустими значения)	Domain/домен
Релация	
ADD	CustomerAddress

ART	Article
ACF	ArticleClaimFor
AD	ArticleDocument
CAAC	CustomerAgentAddressContact
ORD	Order
OD	OrderDetails
OO	OrderOther
OOSM	OrderOtherShipMethod
OCD	OrderCustomerShipMethodDetailsOrderOther
OASM	OrderOtherArticleShipMethod
OESU	OrderDetailsEmployeeSalesSpecialOfferUnitPrice
SSO	SalesSpecialOffer
SUP	SalesUnitPrice
SAL	Sales
HIST	History
CUST	Customer
DOC	Document
CC	CustomerContact
AS	ArticleStorehouse
RAIT	ArticleReviewRaiting
AM	ArticleModule
AME	ArticleModuleExec
AMD	ArticleModuleDefect
CA	CustomerAgent
ARC	ArticleResultClaim
ACOI	ArticleClaimOutsideInspection
AOA	ArticleClaimForOutsideInspectionArticleResultClaim
AOE	ArticleStorehouseExModRevClOrderEmployee
EMP	Employee
AE	ArticleExecution
AR	ArticleReview
TYPE	Type

Атрибут	
FirstN	FirstName
LastN	LastName
PosAgent	PositionAgent
Ph	Phone
IncN	IncomingNumber
InvNum	InvoiceNumber
PersCust	PersonallyCustomer
DatePers	DatePersonally
Art_id	SerialNumberArticle
DateMod	DateModification
OpinionRec	OpinionRecomendation
ResInsp	ResultInspection
ClaimRec	ClaimRecognize
MechConstr	MechanicalConstruction
StickingPl	StickingPlaster
CardIdent	CardIdentificator

Съдържание

Приложения стр.111

Въведение

Целта на дипломната работа е анализ и разработка на модул от система за автоматизация на документооборота за предприятие, работещо в сферата на информационните технологии.

Задачата за събирането, обработката и разпространението на информация стои пред човечеството във всеки един етап от неговото развитие. В продължение на дълги години основен инструмент за решението й били човешкият ум, език и слух.

Първото важно изменение става с появата на писмеността, а след това и с появата на книгопечатането. В епохата на книгопечатането основен носител на информация става хартията.

Положението коренно се променя с появата на компютрите. Принципно нова крачка е направена, когато от използването им за решение на отделни задачи се преминава към използването им за комплексна автоматизация на едни или други завършени части от дейността на човек за преработка на информацията.

Един от първите примери за подобно системно използване на компютрите в световната практика били така наречените административни системи за обработка на данни: автоматизация на банковите операции, счетоводните отчети, резервиране и оформление на билети и т.н. Решаващо значение за ефективността на системи от подобен род има обстоятелството, че те са базирани на автоматизирани информационни бази данни.

При решаване на поредната задача системата се нуждае от въвеждане на малко допълнителна информация, останалото се взима от информационната база. Всяка нова информация, която се въвежда, изменя съществуващата база на системата. Тази база (информационна, или база данни) по този начин се намира в състояние на непрекъснато обновяване, отразявайки всички изменения, произтичащи реално в обекта, с който има работа системата.

Особено място заемат базите данни и другото програмно обезпечение, свързано с тяхното използване в качеството на инструмент за деловодство и рационализиране на труда. Тяхното използване позволява да се съкрати времето, което е необходимо за подготовка на конкретните маркетингови и производствени проекти, да се намалят загубите при тяхната реализация, да се изключи възможността за поява на грешки в подготовката на счетоводна, технологична или друг вид документация, която дава на организацията непосредствен икономически ефект.

За разкриването на всички потенциални възможности, които използват бази данни, е необходимо в работата им да се използват комплекс от програмни и апаратни средства, максимално съответстващи на поставените задачи. Затова в днешно време фирмите имат огромни изисквания към компютърните програми, които поддържат и съгласуват работата на управленческото и финансовото звено на дадена компания, а също и за оптималното използване на компютърното оборудване.

Широкото внедряване на информационните технологии в живота на съвременното общество довежда до необходимост от решения на редица информационни проблеми, свързани с информационната безопасност. Изискванията за гарантиране на безопасността в различните ИС могат да се различават съществено, но те винаги са насочени към това да се достигнат три основни свойства:

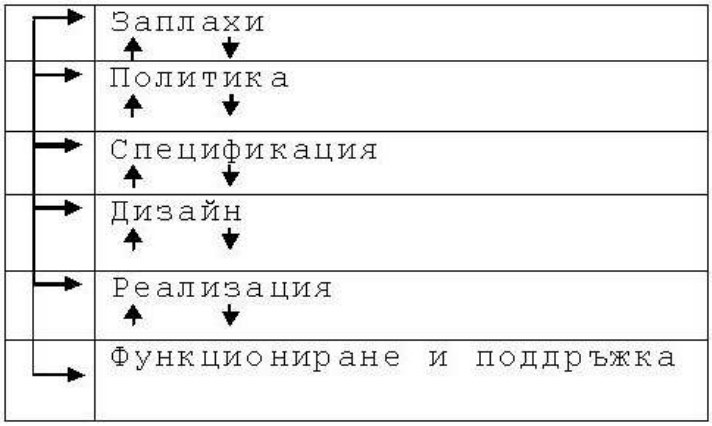
1. Цялостност – информация, на основата на която се приемат решения, трябва да бъде достоверна и точна, защитена от възможно неумишлено и умишлено изопачаване.
2. Достъпност - информацията и съответните автоматизирани служби трябва да бъдат достъпни, готови за работа винаги, когато възникне необходимост.
3. Конфиденциалност – засекретената информация трябва да бъде достъпна само за този, за когото е предназначена.

За решения на проблемите от информационната безопасност е необходимо да се съчетаят законодателните, организационните и програмно-технологичните средства.

Основно внимание в теорията и практиката за гарантиране на безопасността в използването на информационните технологии и системи е съсредоточено в защитата от умишлено разваляне на програмните средства и изопачаването на информацията в базата данни. Затова са разработени и се развиват проблемно-ориентирани методи и средства за защита от:

1. несанкциониран достъп;
2. различни типове вируси;
3. изтичане на информация;

Най-често сигурността се разглежда като: ”необходимите действия за защита на компютъра и информацията, която той съдържа”. В глобалната мрежа по принцип комуникациите са отворени и неконтролирани. Тази идея за отвореност влиза в противоречие с изискването за конфиденциалност и цялостност на транзакциите. Тези две изисквания могат да се реализират, ако бъде осигурена системна сигурност, добре конфигуриран софтуер и внимателно подбран персонал от специалисти. Цикълът по изграждане и поддържане на сигурността на една система се представя схематично чрез Фиг.1 по следния начин[1]:



Фиг.1

Дейността на всяко едно предприятие е свързана с оформяне на голямо количество документи. Преминавайки през всяка инстанция, документът трябва да бъде проверен, заверен и отправен до следващото местоназначение. Това предполага грешки,закъснения,загуби. Решението за всеки директор:

автоматизация на процесите, на подготовка и съгласуване на документите.

Целта на настоящата дипломна работа е да бъде проектирана и създадена единна база данни, в/от която да се съхранява/извлича обработваната информация за клиенти, поръчки, рекламации и служители на компанията, работили по тях. Да се осигурят правдоподобни, непротиворечиви данни в базата данни, която да бъде защитена от външни, злонамерени атаки. Да се разгледат етапите на жизнения цикъл на проектирането й, и да се спазят основните принципи за такава база от данни, а именно:

- Структуриране на информацията
- Осигуряване на бърз достъп до БД
- Осигуряване на възможност за разширение на БД
- Осигуряване цялостност на данните в проектираната база от данни.
- Предотвратяване на несанкциониран достъп
- Предоставяне на ограничен достъп

Тази база от данни да е част от информационна система за управление на производството, продажбите, маркетинга и човешките ресурси разработена за целите на Мултипроцесорни системи ООД.

Описание на структурата на дипломната работа:

Въведение – представя проблема, който трябва да бъде решен, като са формулирани целите и задачите, които се решават в настоящата дипломна работа.

Организиране на деловодството в дейността на стопанските субекти – описва замисъла и целите на разработката на системата, целите и задачите на документооборота и се дава кратко описание на методите и средствата за защита на информацията.

Advanced Encryption Standard (AES) – представя историята и описанието на алгоритъма за криптиране AES.

Анализ на предметната област – дава описание на предметната област и построяването на модел на последната, чрез диаграми на потоците от данни (DFD).

Анализ на представянето на моделите на данните – прави се обосновка на избора на логически и концептуален модел на данните; избор на конкретна СУБД и език за манипулиране на данните.

Архитектура на разработваната система – дава описание на технологиите, използвани в настоящата работа, както и описание на модела на сигурност, който използваме.

Описание на реализацията - описва проектиране на база данни (създаване на ER-диаграма) създаване на приложение за Windows, което съдържа форми, свързани с базата данни; дизайн на формите и определяне действията и взаимодействията между отделните интерфейси на модула.

Глава 1

Организиране на деловодството в дейността на стопанските субекти

1.1 История на развитието на информационните системи

Първите информационни системи се появяват през 1950 година. В тези години те били предназначени за обработка и пресмятане на заплати. Това довежда до намаляване на загубите и времето за подготовка на хартиените документи.

През 1960 година се изменя отношението към информационните системи. Информацията от тях започва да се използва за периодични отчети с много параметри. За тези цели на организациите било нужно компютърно оборудване с широко предназначение, способно да извършва множество функции.

През 70-те и началото на 80-те години информационните системи започват да се използват в качеството на средство за управление и контрол, което поддържа и ускорява процеса за вземане на решения.

Към края на 1980 година концепцията за използването на информационните системи отново се изменя. Те стават стратегически източник на информация и се използват във всички нива на дадена организация с различен профил.

Период	Концепция за използването на информацията	Вид информационни системи	Цел на използване
1950-1960г.	Хартиен поток на ведомостта за заплатите.	Информационни системи за обработка на документи, служещи за	Увеличаване скоростта за обработка на документите. Опростява се процедурата за

		пресмятане...	обработка на сметките и пресмятането на заплатите.
1960-1970г.	Основна помощ в подготовката на отчети.	Управленски информационни системи за производствена информация.	Ускоряване процеса за подготовка на отчети.
1970-1980г.	Управленски контрол на реализация (продажби).	Системи за поддръжка приемането на решения. Системи за по-висшите звена.	Създаване на най-рационалното решение.
1980-2000г.	Информация-стратегически ресурс, гарантиращ конкурсно преимущество.	Стратегически информационни системи. Автоматизирани офиси.	Оцеляване и процъфтяване на организацията.

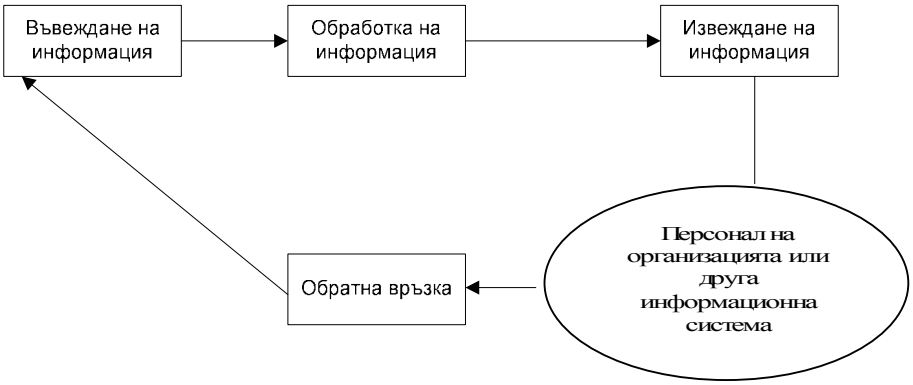
Фиг.2 История на информационните системи

1.2 Процеси в информационната система

Процесите, които гарантират работата на информационните системи във всяко едно направление (Фиг 3.) са следните:

1. Въвеждане на информация от външни или вътрешни източници;
2. Обработка на входната информация и представянето й в приемлив вид;
3. Извеждане на информацията за потребителите или предаването й в друга система;

4. Обратна връзка- това е информация, обработена от хората на дадена организация за корекция на входната информация.



Фиг 3. Процеси в информационната система

Необходимо е да се разбере същността на проблема, който информационната система решава, а също и организационните процеси, в които е включена. Например, при определени възможности компютърната информационна система за поддръжка на приемане на решения, трябва да се вземе под внимание:

1. Структуриране на решаваните управленски задачи;
2. Йерархичното ниво на управление в дадена организация, на което трябва да се вземе решение;
3. Принадлежност на решаваната задача към една или друга функционална сфера в бизнеса;
4. Вид на използваните информационни технологии;

Технологията на работа в компютърната информационна система е достъпна за разбиране от специалисти,а така също и от други специалисти, извън областта на компютрите и може да бъде използвана успешно за контрол на процесите от професионалната дейност и тяхното управление.

Внедряването на информационните системи може да способства за:

1. Получаване на повече рационални варианти за решение на управленски задачи, за сметка на внедряването на математическите методи и интелектуални системи и т.н;
2. Освобождаване на служителя в дадената организация от рутинната работа за сметка на автоматизацията;
3. Гарантиране достоверност на информацията;
4. Замяна на хартиения „носител" на данни на диск, което води до по-рационална организация за обработка на информацията на компютър и намаляване обема на документите на хартя;
5. Усъвършенстване на структурата на потоците от информация и системите за документооборот в организацията;
6. Намаляване на загубите за производство на продукти и услуги;
7. Предоставяне на потребителите уникални услуги;
8. Намиране на нови пазарни ниши;

1.3 Замисъл и цели на разработваната на система

В съвременните условия важна област става информационното осигуряване, което се състои в събиране и обработка на информация, необходима за приемане на обосновани управленчески решения. Предаване на информация за положението и дейността на дадена фирма на високо ниво на управление и взаимният обмен на информация между всички подразделения на организацията се осъществява на базата на компютри и други технически средства за връзка.

В дейността на държавните структури, които представляват комплекс от огромно число всекидневно свързани и взаимодействащи подразделения, предаването на информация се явява първостепенен и задължителен фактор за нормалното функциониране на дадена структура. Тук голямо значение се дава на осигуряването на достоверност на информацията. За много

организации вътрешната система за информация решава задачи за организацията на технологичния процес и има производствен характер. Информацията играе важна роля в предоставянето на сведения за приемането на управленчески решения и се явява един от факторите, които гарантират намаляването на производствените разходи. Информационните системи, предоставяйки навреме нужната информация, помагат на оганизациите да достигнат успехи в своята дейност, да създават нови стоки и услуги , да намират нови пазари , да си гарантират достойни партньори, да се пуска продукция на ниска цена и други.

Информацията служи като основа за подготовка на доклади, отчети, предложения за изработване и приемане на решения.

Съдържанието на всяка конкретна информация се определя от потребностите на управленческите звена и изработваните управленчески решения. Към информацията има следните изисквания:

1. кратки и ясни формулировки, своевременно постъпване;
2. целенасоченост- да удовлетвори конкретнните потребности;
3. за точност и достоверност- правилен подбор на първичните сведения;

1.4 Общи изисквания и задачи,които се решават от системите за документооборот

Разглеждат се общите изисквания към системите за документооборот.

1. *Мащабируемост.* За да може системата за документооборот да поддържа както пет, така и пет хиляди потребители и нейните способности да растат, мощността й се определя само от мощността на апаратното оборудване, на което е инсталирана.

2. *Разпределение.* Основните проблеми при работа с документи възникват в териториално-разпределените организации, затова архитектурата на системата за документооборот е длъжна да поддържа взаимодействие между отделните звена. Те могат да бъдат обединени чрез най-разнообразни по скорост и качество канали за връзка. Също така архитектурата на системата е длъжна да гарантира взаимодействие с отдалечени потребители.
3. *Модулност.* Съществува възможност, клиент да не поиска веднага внедряване на всички компоненти на системата за документооборот и понякога кръга на решаваните от клиента задачи е по-малък от спектъра задачи за документооборота. Затова системата трябва да се състои от отделни модули, които са интегрирани помежду си.
4. *Откритост.* Системата за документооборот не може и не е длъжна да съществува отделно от другите приложения, например често е необходимо да се интегрира система с приложна счетоводна програма. Следователно, системата за документооборот е длъжна да има открити интерфейси за възможно доработване и интеграция.

Ще се разгледа общия спектър от задачи за електронния документооборот. Задачите, и съответно необходимата система за автоматизация се определят от стадиите на жизнения цикъл на документа, който е необходимо да поддържат. Жизненият цикъл се състои от два основни стадия:

1. Разработка на документа, която може да включва собствена разработка на съдържанието на документа, оформяне и утвърждаване на документа. В такъв случай, ако документа се намира в стадий на разработка, той се счита за непубикуван и правата над него се определят от правата за достъп на конкретния потребител.
2. Стадий на публикуваният документ, който може да съдържа: активен достъп, архивен документ за краткосрочно или дългосрочно съхранение, унищожение на документа.

Когато документът преминава във втори стадий, той се счита за публикуван, и върху него има едно право- право на четене. Освен права за достъп, за четене, може да съществуват права за превод на публикуван документ в стадия на разработка.

В зависимост от конкретния стадий на жизнениея цикъл на документа, с които работят архивните системи, те се разделят на следните типове:

1. Статични архиви на документите(или прости архиви)- системи, които обработват само публикувани документи.
2. Динамични архиви (или системи за управление на документи)- работят както с публикувани документи, така и с тези, които се намират още в разработка.

1.5 Информационна безопасност

В съвременния свят информационният ресурс става един от най-мощните за икономическо развитие. Владеенето на информацията с необходимите качества в нужното време и на нужното място се явява залог за успех във всяка една стопанска дейност. Монополното притежание на определена информация се оказва преимущество в конкурентната борба и с това определя високата цена на “информационния фактор”.

В днешни дни е трудно да си представим предпиятие или организация(с изключение на най-малките фирми), което да не е снабдено със съвременни средства за предаване и обработка на информацията. В компютрите се събира значителен обем информация, която често има конфиденциален характер или е важна за нейния “притежател”.

В общият случай под понятието информационна безопстност се разбира защитеност на информацията и поддържащата инфраструктура от случайни или преднамерени въздействия от „естествен или изкуствен” характер, способни да нанесат вреди на „притежателят” и/или на потребителите на автоматизираната система и на инфраструктура й.

Информационната безопасност в съвременните информационни системи съгласно общоприетия подход включва три взаимосвързани аспекта:

1. Цялостност – актуалност и непротиворечивост на информацията, нейната защитеност от разрушения и несанкционирани изменения;
2. Конфиденциалност – защита от несанкционирано „запознаване" с информацията.
3. Достъпност – възможност за приемливо време, в което да се получи изисканата информация, информационна услуга.

1.5.1 Класификация на методите и средствата за защита на информацията

Разработените методи и средства за защита могат да се разделят на четири основни групи:

1. *Апаратни*- използват се на различни нива.
2. *Програмни*- по- надеждни и продължителността на гарантираното им използване без препрограмиране е по-голяма, отколкото при апаратните средства. Функции на специализирани програми, осигуряващи програмна защита са:
    a. Идентификация на техническите средства и файловете и разпознаване на правоимащите потребители.
    b. Регистриране и контрол на работата на апаратните средства и потребителите.
3. *Организационни*- допълват и разширяват функциите на апаратните и програмните методи за защита на данните. Тъй като несанкционираният достъп и изтичането на информация най-често са в резултат от злонамерени действия, небрежност или неквалифицирани действия от страна на персонала и потребителите, влиянието на тези и на други подобни фактори трудно може да бъде ограничено с помощта на апаратни и програмни средства, както и чрез

криптографски методи. При проектирането и изграждането на БД и при нейната последваща реализация ролята на организационните методи се състои в:

   a. Изключване на възможността за непозволено проникване в работните помещения.
   b. Осигуряване на строг контрол за подбор и подготовка на персонала.
   c. Определяне на правила за въвеждане, обработка и съхраняване на постъпващите информационни източници.

4. *Криптографски*- намират широко приложение в системите, работещи със секретни данни или с данни за ограничено ползване. Предаването на информация по каналите за връзка в мрежа от компютри с използването на криптографските методи е едно от най-достъпните и рационални средства за защита на БД.

Най-общо криптографията може да бъде определена като система за защита на данните чрез използване на методи за шифриране и дешифриране, при което предназначената за определен потребител/ потребители информация се подава в такъв вид, че да бъде недостъпна за останалите потребители. Следователно, главната задача, стояща пред криптографията, може да се формулира така:

1. *Осигуряване на секретност*- предотвратяване на неавторизиран достъп до информацията, предавана по незащитен канал. Съответните системи за засекретяване гарантират на подателя на съобщението, че то ще бъде прочетено само от получателя, за когото е предназначено.
2. *Установяване на достоверност*- системите, създадени на този принцип предотвратяват неавторизирано подаване на съобщения по каналите за връзка, като по този начин се гарантира на получателя на съобщението достоверността на неговия подател.

1.5.2 Основни задачи на криптографията

Задачата на криптографията, тоест тайното предаване на данни възниква само за информацията, която се нуждае от защита. В такива случаи се казва ,че информацията съдържа тайна или се явява защитена, частна, конфиденциална, секретна. За най-типичните, често срещани ситуации от такъв тип, са въведени специални понятия, като:

1. държавна тайна;
2. военна тайна;
3. юридическа тайна;
4. лекарска тайна;
5. търговска тайна и др.;

Когато се говори за защитена информация, ще се има предвид следните нейни признаци:

1. Има определен кръг законни потребители, които имат право да използват информацията.
2. Има незаконни потребители, които се стремят да овладеят тази информация и да се възползват от нея за облагодетелстване, и да нанесат вреди на законните потребители.

Криптография- това е набор от методи за защита на информацията от злоумишлени действия на различни субекти, чрез методи, които се базират на секретни алгоритми за преобразуване на информацията, включващи алгоритми, които сами по себе си не се явяват секретни, но използват секретни параметри.

Исторически погледнато, първата задача на криптографията била защитата на предаваните текстови събщения, от несанкциониран достъп, което намира и отражение в самото название на тази дисциплина. Тази защита се базира на използването на “секретен език” , известен само на подателя и

получателя, като всички идеи за шифриране възникват само с развитието на тази философска идея.

Криптографски методи за защита се наричат средства и методи на преобразуване на информацията, в резултат на което нейното съдържание остава скрито.

Алгоритмите за шифриране с използването на ключ предполагат, че данните никой не може да ги прочете, освен този, който има ключ за тяхното дешифриране.

1.5.3 Симетрични алгоритми

Ще дадем описание на няколко типа симетрични алгоритми. При тях за шифриране и дешифриране се използват едни и същи алгоритми. Един и същ секретен ключ се използва за шифриране и дешифриране. Този тип алгоритми се използват както от симетричните, така и от асиметричните криптосистеми.

Тип	Описание	Предимства и недостатъци
DES	Популярен алгоритъм за шифриране. Използва се като стандарт за шифриране на данни от правителството на САЩ. Шифрирането на блока от данни зависи от резултата от шифрирането на предишните блокове данни.	Шифрира се блок от 64- бита, използва се 64-битов ключ(от които са необходими само 56-бита). Може да работи в 4 режима: ECB(Electronic Code Book)- Електронна кодова книга- обикновен DES, използва два различни алгоритъма. CBC(Cipher Block Chaining),)- Режим на блоковата верига, в който шифрирането на блока от данни зависи от резултатите на шифрирането на предишните блокове от данни. OFB(Output Feedback)- режим с вътрешна обратна връзка, близък до CFB, ключовата разлика е, че блокът който се прилага с "изключващо или" към открития текст се генерира независимо както от открития, така и от шифрования текст. Използва се като генератор на случайни числа.

		CFB(Cipher Feedback)- режим с обратна връзка – използва се за получаване на кодове за аутентификация на съобщението. Използва се "изключващо или", но при този режим криптирането е на порции по-малки от размера на блока.
TripeleDES	Симетричен блоков шифър, който е създаден на основата на DES, с цел да бъде премахнат неговият основен недостатък – малката дължина на ключа(56 бита), който може да бъде "разбит" чрез метода "разбиване не ключа". При TripeleDES е избран прост начин за увеличаване на дължината на ключа, без да се преминава към нов алгоритъм- в него над 64 битовия блок от данни няколко пъти се извършва шифриране с помощта на DES(с различен ключ). В най-прост вариант: DES(k3;DES(k2;DES(k1;M))), където M-блок от началните данни; k1, k2 и k3 — ключовете при DES.	По - усъвършенстван вариант на DES. При този алгоритъм открития текст се пропуска през алгоритъма DES три пъти (втората операция се провежда в режим на дешифриране). Реализацията на трите операции се извършва чрез различни 56 – битови ключа, което е равносилно на 168 - битов ключ. Скоростта за шифриране и дешифриране на данните се понижава. Тук проблемът с малката дължина на ключа е решен, но основен недостатък се явява: нарастване на сложността и времето за шифриране и дешифриране – удачен за приложения, при които времето за реализация не е критично.

AES	Описанието следва в глава 2 - Advanced Encryption Standard	Основан на алгоритъма Rijndael. Нов стандарт за симетричен алгоритъм , приет през 2000 г., използва ключове с дължина 128, 192, 256 бита. Дължина на блока – 128 бита. AES се основава на permutations(пермутация-изменение на реда на данните) и substitutions(заместване на един блок от данни с друг). В AES се използват няколко ключа вместо 1 (новите ключове се наричат итеративни (round keys), за да се подчертае различието им от изходния ключ). Общо мнение:най-устойчив алгоритъм за шифриране от съществуващите. Единствения ефективен начин е чрез груба сила- да се преодолеят всички възможни ключове.
RC2	Блоков шифър с дължина на блока 64-бита, с променлива дължина на ключа, разработен от Рон Ривест. Устойчивостта може да бъде по-голяма или по-малка в сравнение с DES, в зависимост от дължината на ключа. Алгоритъмът RC2 е собственост на компанията RSA Security Inc. като за неговото използване трябва лиценз. В САЩ дължината на ключа за използване вътре в страната е 128-бита, но споразумението сключено между Software Publishers Association (SPA) и правителството на	RC2 и RC4 са блокови шифри, с ключ с променлива дължина. Приблизително два пъти по-бърз в сравнение с DES. Специално разработен за да замени ("drop-in" replacement) DES. RC2 може да се използва там където се използва и DES. RC2 и RC4 с ключове от по 128 бита осигуряват такова ниво на безопастност както IDEA и TripleDES. Конфиденциален алгоритъм – принадлежи на RSA Data Security.

	САЩ дава на RC2 специален статус, който означава че е разрешено използването на шифър с дължина на ключа до 40 бит. 56-битови ключове е разрешено да се използват в задграничните представителства на американските компании.	
Rijndael	AES не е тъждествен на Rijndael, тъй като Rijndael поддържа широк диапазон за дължината на ключовете и блоковете. В AES размерът на блока е фиксиран и е равен на 128 бита, размера на ключа може да бъде 128, 192, 256 бита, а в Rijndael се поддържат различна дължина на ключовете и блоковете- от 128 до 256 бита, със стъпка 32 бита. Ключът за шифриране може да бъде разширен, с помощта на процедурата Rijndael's key schedule.	Блоков алгоритъм за шифриране с променлива дължина на блока и променлива дължина на ключа. Дължината на блока и на ключа могат да бъдат независимо установени от по 128, 192 или 256 бита. Rijndael може да се изпълнява по-бързо, отколкото обикновен блоков алгоритъм за шифриране. Порменлива дължина на блока- от 192 до 256 бита позволява да се създава хеш-функция без колизии (дължина на блока от 128 бита за тази цел днес се счита за недостатъчна). Може да се използва в качеството на итерационна хеш-функция. Съществува 1 възможна реализация. Препоръчва се дължината на блока и на ключа да е равна на 256 бита. Съществуват много способи , с помощта на които Rijndael може да се използва в качеството на генератор на псевдослучайни числа
IDEA	Блоков алгоритъм за шифриране на данни, патентован от швейцарската фирма Ascom. Първоначално се наричал IPES (Improved PES), тъй като се явявал развитие на	IDEA – нов и недостатъчно изследван алгоритъм. Използва се 128 – битов ключ за криптиране на 64 битови блокове от данни. Няма официално документирани недостатъци в сигурността (до момента). За шифриране и дешифриране се

	стандарта PES (Proposed Encryption Standard). Лицензът позволява да се използва алгоритъма в приложения, с не търговска цел.	използва един и същи алгоритъм.

Фиг.4 Видове симетрични алгоритми

1.5.4 Асимтерични алгоритми

Асиметричните алгоритми се използват в асиметричните криптосистеми за шифриране на сесийни ключове (които се използват за шифриране на самите данни).

Използват се два различни ключа- единят известен на всички, а другият се пази в тайна. Обикновено за шифриране и дешифриране се използват и двата ключа. Но данните шифрирани с единя ключ може да се дешифрират само с помощта на другия ключ.

Тип	Описание	Предимства и недостатъци
DSA	Тайното създаване на хеш-стойността и нейната публична проверка- само един човек може да създаде хеш-значение(стойност) на съобщението, но всеки може да провери нейната коректност.	Алгоритъм с открит ключ за създаване на електронен подпис, но не и за шифриране.
RSA	Автори на алгоритъма- Rivest, Shamir, Adleman. Използват сравнимост на числата. Това е зависимост между две цели числа a и b , изпълнена точно тогава, когато a-b се дели на дадена цяло число m(начин на записване: a=b(mod m)). Алгоритъмът се основава	Сигурността зависи от дължината на ключовете – в момента 384 битови ключове са съвсем лесно пробиваеми с директни атаки. Смята се, че 1024 битови ще бъдат сигурни векове (а с програмата PGP масово се използват 2048 битови ключове). Тези предположения ще са

	на факта, че в съвременните компютри изчисляването на съставните числа като произведение от големи прости числа е свързано с извършване на сложни и продължителни изчисления. Експериментални данни показват, че този проблем е трудно решим.	верни , само ако не се измисли начин , по който може да бъде изведен е-ти корен по модул n, но не е доказано че не могат да съществуват подобни прости аналитични атаки. Подобна е ситуацията и при другите два алгоритъма – DSA и Diffie-Hellman На практика няма "лесен" начин за разбиване на ключове при тези алгоритми което гарантира, че в близко бъдеще те ще покриват нужното ниво на сигурност.
ECDSA	Алгоритъм с открит ключ за създаване на цифров подпис, аналогичен по начин на създаване на DSA.	
Diffie-Hellman	Алгоритъм, който позволява на двете страни да получат общ секретен ключ, използвайки незащитен канал за връзка. Този ключ може да бъде използван за шифриране на по-нататъшния обмен на данни, с помощта на алгоритъм за симетрично шифриране. За първи път този алгоритъм е публикуван от Whitfield Diffie и Мартин Хелман през 1976 година. През 2002 година Хелман предлага да нарекат този алгоритъм "Диффие — Хелман — Меркле".	Дифие и Хелман предполагат много голямо първоначално число за криптиране на информацията, може да извлечете от него второ също така много голямо число. Второто, въпреки че е различно от криптиращото първо, ще декриптира информацията. Формулата на двамата учени е основното уравнение, което е заложено във всички модулни криптиращи техники. Ключовият протокол Дифие-Хелман позволява две страни (примерно Ларс и Хапи) да комуникират през незащитен канал с определен ключ (който може да бъде както потребителски частен ключ, така и поделен

		частен ключ за реални комуникации) по такъв начин, че трета страна), който подслушва този канал, да не може да разгадае ключа.

Фиг.5 Видове асиметрични алгоритми

1.5.5 Хеш-функции

Хеш-функциите се явяват едни от най-важните елементи на криптосистемите на основата на ключове. Лесно се изчисляват, но почти е невъзможно да се разшифрират. Хеш-функцията има изходни данни с променлива дължина и връща ред с фиксирана дължина(понякога се нарича дайджест съобщение- MD).Обикновено 128 бита. Хеш-функцията се използва за намиране на модификация на събщението.

Тип	Описание	Предимства и недостатъци
MD2	Най- бавната хеш-функция, оптимизирана за 8-битови машини.	
MD4	Еднопосочна хеш-функция, разработена от Рон Ривест. За всяко входящо съобщение функцията връща 128- разрядно хеш-значение(стойност), наричано още "дайджест" на съобщението. Този алгоритъм се използва по-точно за създаване на NT- хеш на паролите в системите Windows NT, 2000, XP и Vista.	Известни уязвимости – съществуват.

MD5	128- битов алгоритъм за хеширане, разработен от професор Роналд. Л. Ривест през 1991 година. Най-разпространена от семейството MD-функции. Обезпечава цялостта на данните. Предназначен за създаване на "отпечатъци" или "дайджести" на съобщение с произволна дължина. Заменя MD4.	Добавя се четвърти етап , в сравнение с MD4. Във всяко действие се използва уникална константа, която се добавя. Известни уязвимости-теоретично има. През 2004 година за малко време китайските изследователи Сяоюнь Ван (Xiaoyun Wang), Денгуо Фен (Dengguo Feng), Сюецзя Лай (Xuejia Lai) и Хонбо Ю (Hongbo Yu) обявяват, че са намерили уязвимост в алгоритъма, позволяващ да се намери колизия. Авторите не откриват своята тайна пред широката публика.
SHA	Създава 160-битова стойност(значение) от изходни данни с променлив размер. Предложена от NIST и приета като стандарт от правителството на САЩ. Предназначена за използване в стандарта DSS.	Подобен много на MD4 , по-скоро вариант на MD4. Позволява да се генерират електронни подписи с дължина от 160 бита за съобщение с произволна дължина. Добавя се четвърти етап. Счита се че е по-защитен от MD5.(особености в етапите)
RIPEMD-160	Един от алгоритмите за криптографско хеширане. Дължината на резултатния "дайджест" е 160 бита. Разработен от Hans Dobbertin, Antoon Bosselaers и Bart Preneel. Заради проблеми, свързани с MD4, MD5, европейската организация RIPE(RACE Integrity Primitives	Вариант на MD4 Особености в етапите

	Evaluation) разработва стандарта RIPEMD-160, на основата на MD4, но генерира "дайджест" с дължина 160 бита. Заедно с SHA-1, този алгоритъм е по-малко ефективен, отколкото MD5.	
HAVAL	Еднопосочна хеш-функция с променлива дължина. Тази функция е модификация на функцията MD5. . Функцията HAVAL може да даде хеш-значение с размери 128, 160, 192, 224 или 256 разряда.	Хеш-функция с променлива дължина Явява се модификация на MD5. Обработва съобщение на блокове от по 1024 бита, два пъти по-големи, в сравнение с MD5. Особености в етапите

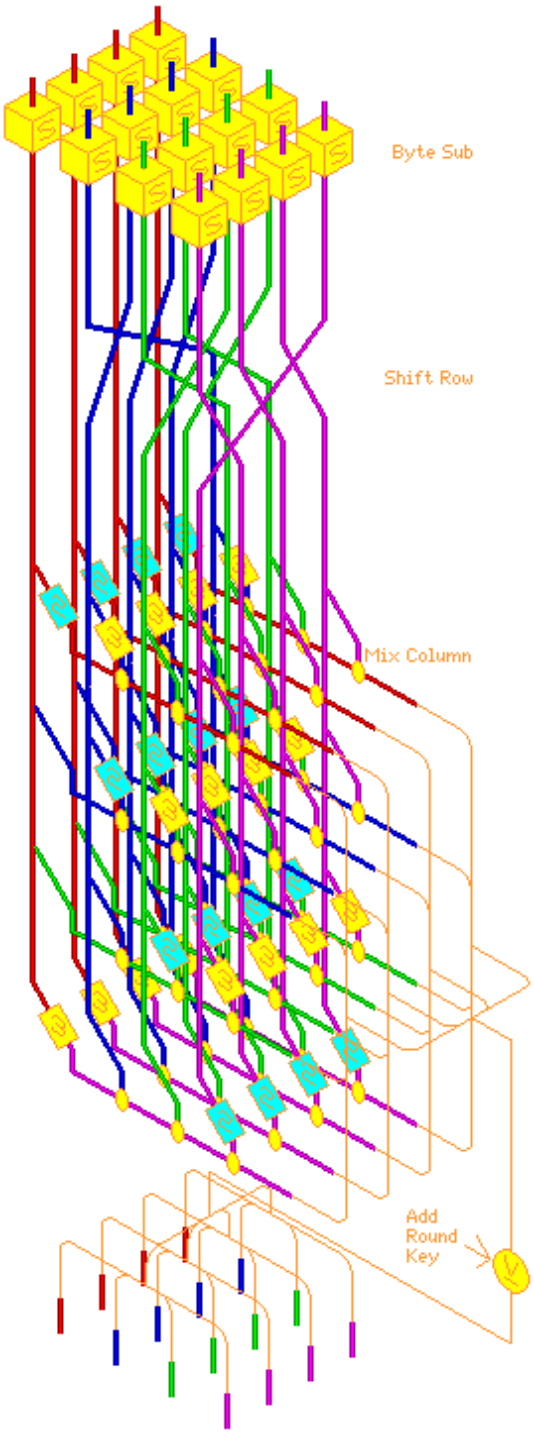
Фиг.6 Хеш-Функции - видове

Алгоритъм	Дължина на хеш – стойност	Скорост на шифриране (kB /sec)
MD2	128	23
MD4	128	236
MD5	128	174
SHA	160	75
RIPE-MD	128	182
HAVAL (3 прехода)	Променлива	168
HAVAL (4 прехода)	Променлива	118
HAVAL (5 прехода)	Променлива	95
Други:		
Хеш-функция ГОСТ	256	11
Davies-Meyer(с DES)	64	9
Още...		

Фиг.7 Характеристики на алгоритми

Глава 2

Advanced Encryption Standart (AES)



С развитието и внедряването на компютрите в различни сфери на човешката дейност(например: научни изследвания, банково дело, електронна търговия и др.) все повече се обръща внимание на въпроса за безопасния обмен на данни през незащитени канали. Единственото безопасно средство за защита на информацията остава нейното криптографско преобразуване.

След няколко години публично обсъждане на 2 октомври 2000 година Националният институт по стандарти и технологии на Съединените щати (NIST) утвърждава новият разширен стандарт за шифриране AES (Advanced Encryption Standard), който заменя DES (Data Encryption Standard), като последният повече от 20 години се явява общоизвестен стандарт за шифриране.

Необходимостта от приемане на нов стандарт се появява поради недостстъчната дължина на ключа на DES (56 бита). Също така, програмната реализация на алгоритъма DES върху платформи с ограничени ресурси няма исканото бързодействие.

За да може алгоритъмът AES да стане достоен "заместник" на DES, неговата архитектура трябва да отговаря на няколко критерия:

1. да предлага висока степен на защита;
2. да притежава проста структура;
3. да притежава висока производителност;

С най-висок приоритет за алгоритъмът AES се явява защитата. На ниво вътрешна архитектура трябва да притежава надеждност (сигурност), достатъчна за да се противопостави на бъдещи опити за разбиване.

Едновременно с това структурата на алгоритъма, трябва да бъде достатъчно проста, за да гарантира ефективна процедура на шифриране.

Следващото изискване към AES – висока производителност. Тя предполага висока скорост за работа при шифриране и дешифриране, а също и при реализацията на ключа.

На 2 януари 1997 година NIST обявява своето намерение да се избере "наследник" на DES, който се явява американски стандарт

от 1977 година. NIST приема различни предложения от заинтересованите страни, за това по какъв начин трябва да бъде избран алгоритъма.

За да бъде утвърден в качеството на стандарт, алгоритъмът трябва да отговаря на редица критерии и препоръки.

След първият тур на конкурса, в NIST постъпват 21 предложения, от които 15 удовлетворяват зададените критерии. След допълнителни изследвания на решенията (включително дешифриране и проверка на производителността) през август 1999 година NIST обявява 5 финалиста, сред които и алгоритъмът Rijndael.

Rijndael - бърз и компактен алгоритъм, с проста математическа структура, благодарение на която той се оказва лесен за анализ при оценката на нивата на защита, за което специалистите от NIST не предявяват никакви претенции. Заради тази простота хакерите ще трябва да изучат ограничен математически апарат, и ако някъде в Rijndael има скрити проблеми, рано или късно те ще бъдат намерени.

Алгоритъмът демонстрира много добра устойчивост на атаки, при които хакерът се опитва да декодира шифрираното съобщение, като анализира външните му проявления, включително и време за изпълнение. Rijndael може лесно да се защити от такива атаки, тъй като той се базира основно на булеви операции.

Общата производителност на програмната реализация на Rijndael се оказва най-добра. Алгоритъмът отлично преминава през всички тестове. Увеличаването на ключа отчасти забавя неговата работа, тъй като при обработката на ключове с по-голяма дължина алгоритъмът предвижда изпънение на допълнителни рундове на шифрирарне.

NIST спира избора си на Rijndael, тъй като той съчетава в себе си простота и висока производителност.

На 2 октомври 2000 година е обявено, че победител в конкурса е алгоритъмът Rijndael. Започва се процедура по стандартизация, като той е регистриран като официален

федерален стандарт – FIPS 197 (Federal Information Processing Standart) на 26 ноември 2001 година.

NIST потвърждава голямата сигурност на AES чрез астрономически числа. 128- битов ключ осигурява $340*10^{36}$ възможни комбинации, а 256- битовия ключ увеличава това число до $11*10^{76}$. За сравнение, алгоритъмът DES дава общо число на комбинациите $72*10^{15}$. За тяхното разбиване на специално построената машина "DES Cracker" й трябват няколко часа. Дори да можеше да го извърши за 1 секунда, то за разбиването на 128- битов ключ машината би изразходвала 149 трилиона години.

2.1 Описание на криптоалгоритъма

RIJNDAEL – е итерационен блоков шифър, който има архитектура "Квадрат". Шифърърт има променлива дължина на блоковете и различна дължина на ключовете. Дължината на ключа и дължината на блока мога да бъдат равни независимо един от друг, 128, 192 или 256 бита. В стандарта AES е определена дължината на блока от данни, равна на 128.

Промеждутъчните (междинните) резултати от преобразуванията, които се изпълняват в рамките на криптоалгоритъма се наричат *състояния (State)*. Състоянието може да се представи във вид на правоъгълен масив от байтове (Фиг.8):

State

a_0	a_4	a_8	a_{12}
a_1	a_5	a_9	a_{13}
a_2	a_6	a_{10}	a_{14}
a_3	a_7	a_{11}	a_{15}

Фиг.8

На Фиг.1 е показан пример за представяне на 128- разряден блок от данни във вид на масив *State*, където a_i- байт от блока от данни, а всеки стълб- една 32- разрядна дума.

При размер на блока равен на 128 бита, този 16-байтов масив (Фиг.9 а):

s_{00}	s_{01}	s_{02}	s_{03}
s_{10}	s_{11}	s_{12}	s_{13}
s_{20}	s_{21}	s_{22}	s_{23}
s_{30}	s_{31}	s_{32}	s_{33}

Фиг.9 а) Формат на данните:Пример за представяне на блок(Nb=4)

Има 4 реда и 4 стълба(всеки ред и всеки стълб в този случай могат да се разглеждат като 32- разрядни думи). Входните данни за шифъра се обозначават като байтове на състоянието в реда: s_{00}, s_{10}, s_{20}, s_{30}, s_{01}, s_{11}, s_{21}, s_{31}…

След завършването на действията на шифъра, входните данни се получават от байтове на състоянието в същия ред (порядок). В общия случай, числото на стълбовете N_b е равно на дължината на блока, разделено на 32.

Ключът за шифриране също е представен във вид на правоъгълен масив с четири реда(Фиг.2 б)):

k_{00}	k_{01}	k_{02}	k_{03}
k_{10}	k_{11}	k_{12}	k_{13}
k_{20}	k_{21}	k_{22}	k_{23}
k_{30}	k_{31}	k_{32}	k_{33}

б)

Фиг.9 б) Формат на данните:Ключ за шифриране(N_k=4), където s_{ij} и k_{ij}- съответно са байтовете на масива State и ключ, който се намира при пресичането на i- ред и j-ти стълб.

Числото на стълбовете N_k на този масив е равно на дължината на ключа, разделен на 32. В стандарта са определени ключове с размери – 128 бита, 192 бита и 256 бита, със съответно 4,6 и 8

32- разрядни думи(или стълбове- в таблично представяне). В някои случаи ключът за шифриране се разглежда като линеен масив от 4-байтови думи. Думите се състоят от 4 байта, които се намират в един стълб(при представянето във вид на правоъгълен масив). Числото на рундовете N_r в алгоритъма *RIJNDAEL* зависи от стойностите(значения) на N_b и N_k (Фиг.3):

N_r	N_b=4	N_b=6	N_b=8
N_k=4	10	12	14
N_k=6	12	12	14
N_k=8	14	14	14

Фиг.10 Числото на рундовете Nr като функция от дължината на ключа Nk и дължината на блока Nb.

В стандарта AES е определено съответствието между размера на ключа, размера на блока от данни и числото рундове за шифриране(Фиг.11):

Стандарт	Дължина на ключа (N_k думи)	Размер на блока от данни (N_b думи)	Число рундове (N_r)
AES-128	10	12	14
AES-192	12	12	14
AES-256	14	14	14

Фиг.11

2.2 Рундово преобразуване

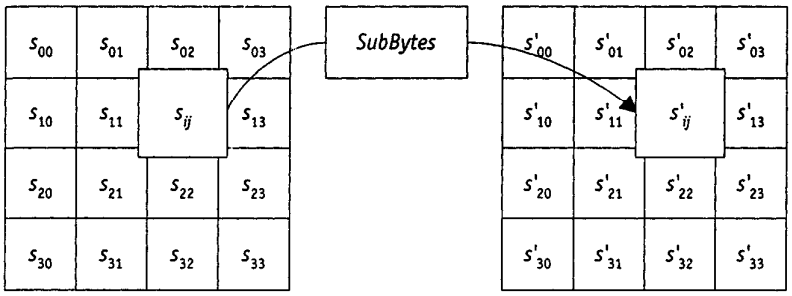
Рундът се състои от четири различни преобразувания:

1. замяна на байтове *SubBytes()*
2. преместване(отместване) на редовете *ShiftRows()*
3. разбъркване(смесване) на стълбовете *MixColumns()*
4. събиране с рундовия ключ *AddRoundKey()*

Замяна на байтовете *(SubBytes())* – преобразуването *SubBytes()* представлява нелинейна замяна на байтове, която се изпълнява независимо с всеки байт на състоянието.

Използването на описания S- блок към всички байтове на съсоянието се означава като SubBytes(State).

Фиг.13 ни показва използването на преобразуването SubBytes() към състояние:



Фиг.13 SubBytes() действа на всеки байт състояния

Логиката на работа на S- блока при преобразуване на байта {xy} е показана на Фиг.14:

x	y															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Фиг.14 Таблица на замените S-блок

Например, резултат {ed} от преобразуването на байта {53} се намира на при пресичането на 5-ти ред и 3-ти стълб.

Преобразуване отместване на редове *(ShiftRows)*

Последните 3 реда на състоянието циклично отместват наляво на различно число байтове. Ред 1 се придвижва на C_1 байт, ред 2- на C_2 байта, ред 3- на C_3 байта. Стойностите на придвижването в RIJNDAEL зависят от дължината на блока Nb. Техните величини са приведени във Фиг.15:

N_b	C_1	C_2	C_3
4	1	2	3
6	1	2	3
8	1	3	4

Фиг.15

В стндарта AES, където е определен единствен размер на блока, равен на 128 бита, C_1=1, C_2=2, C_3=3.
Операцията отместване на последните три реда на състоянието е обозначена като *ShiftRows(State)*.

State					**State'**			
S_{00}	S_{01}	S_{02}	**S_{03}**	Без отместване ⇨	S_{00}	S_{01}	S_{02}	**S_{03}**
S_{10}	S_{11}	S_{12}	**S_{13}**	Циклично отместване на C_1 ⇨	S_{11}	S_{12}	**S_{13}**	S_{10}
S_{20}	S_{21}	S_{22}	**S_{23}**	Циклично отместване на C_2 ⇨	S_{22}	**S_{23}**	S_{20}	S_{21}
S_{30}	S_{31}	S_{32}	**S_{33}**	Циклично отместване на C_3 ⇨	**S_{33}**	S_{30}	S_{31}	S_{32}

Фиг.16 *ShiftRows()* действа на редовете на състоянието

Преобразуването разбъркване на стълбовете (MixColumns)

При това преобразуване стълбовете на състоянието се разглеждат като многочлени над $GF(2^8)$ и се умножават по модул x^4+1 на многочлен g(x), който изглежда по следния начин:

$$g(x)=\{03\}x^3+\{01\}x^2+\{01\}x+\{02\}.$$

Може да бъде представено и в матричен вид по следния начин:

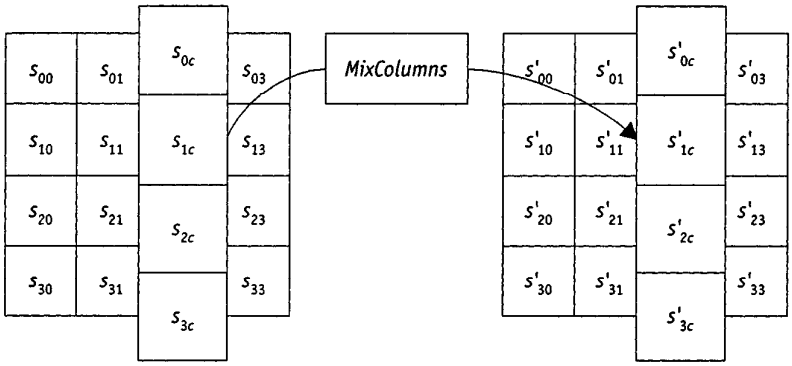
$$\begin{bmatrix} s'_{0c} \\ s'_{1c} \\ s'_{2c} \\ s'_{3c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{bmatrix}, 0 \le c \le 3,$$

Фиг.17

където *c*- номер на стълба в масива State.
В резултат на такова умножение, байтовете на стълба s_{0c}, s_{1c}, s_{2c}, s_{3c} се заменят съответно на байтовете:

$$s'_{0c} = (\{02\} \bullet s_{0c}) \oplus (\{03\} \bullet s_{1c}) \oplus s_{2c} \oplus s_{3c},$$
$$s'_{1c} = s_{0c} \oplus (\{02\} \bullet s_{1c}) \oplus (\{03\} \bullet s_{2c}) \oplus s_{3c},$$
$$s'_{2c} = s_{0c} \oplus s_{1c} \oplus (\{02\} \bullet s_{2c}) \oplus (\{03\} \bullet s_{3c}),$$
$$s'_{3c} = (\{03\} \bullet s_{0c}) \oplus s_{1c} \oplus s_{2c} \oplus (\{02\} \bullet s_{3c}).$$

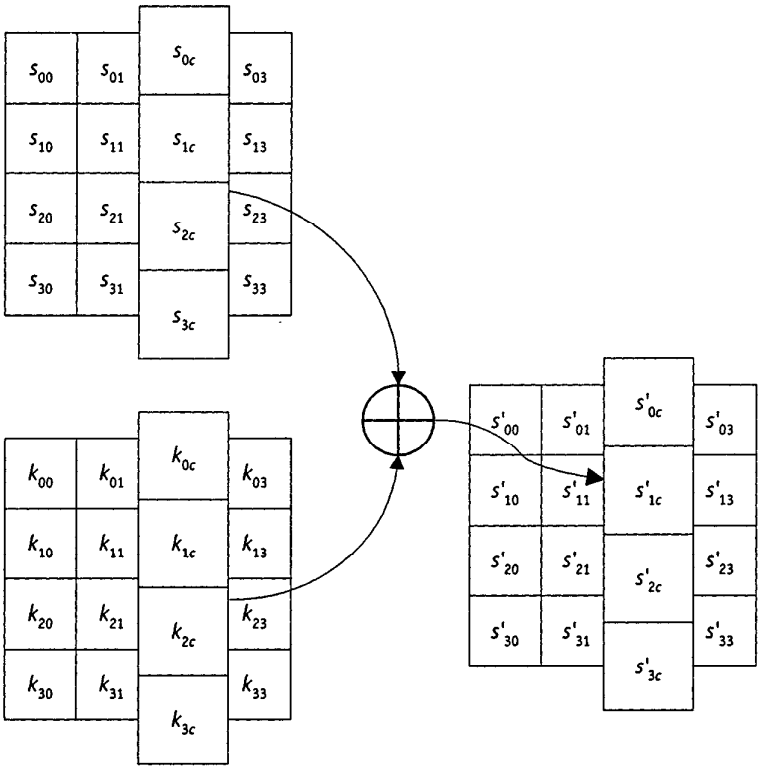
Прилагането на тази операция към всички четири стълба на състоянието е обозначено като *MixColumns(State)*. Фиг.18 показва използването на преобразуването *MixColumns()* към стълбовете на състоянието:



Фиг.18

Добавяне на рундов ключ (*AddRoundKey*)

В дадената операция рундовият ключ се добавя към състоянието чрез просто използване на XOR. Рундовият ключ се изработва от ключа за шифриране чрез алгоритъм за изработка на ключове(key schedule). Дължината на рундовият ключ (в 32- разрядни думи) е равна на дължината на блока Nb. Преобразувание, което съдържа добавяне чрез (посредством) XOR на рундов ключ към състоянието(Фиг. 19), се обозначава като *AddRoundKey(State, RoundKey):*



Фиг.19 При добавяне на ключа, рундовия ключ се добавя чрез операция XOR към състоянието.

Алгоритъм за изработка на ключове (Key Schedule**)**

Рундовите ключове се получават от ключа за шифриране чрез алгоритъма за изработка на ключове. Съдържа два компонента: разширение на ключа (Key Expansion) и избор на рундов

ключ(Round Key Selection). Основополагащите принципи на алгоритъма изглеждат по следния начин:

1. общото число битове на рундовите ключове е равно на дължината на блока, умножено на числото рундове, плюс 1 (например, за блок с дължина 128 бита и 10 рунда се изискват 1408 бита рундови ключове);
2. ключът за шифриране се разширява в разширения ключ (Expanded Key);
3. рундовите ключове се взимат от разширението на ключа по следния начин: първият рундов ключ съдържа първите Nb думи, вторият- следващите Nb думи и т.н.

Разширение на ключа(Key Expansion)

Разширения ключ в RIJNDAEL представлява линеен масив *w[i]* от $N_b*(N_r+1)$ 4- байтови думи, i=1,…, $N_b*(N_r+1)$. В AES масивът *w[i]* се състои от $4*(N_r+1)$ 4- байтови думи, i=0,…, $4*(N_r+1)$.

//--

Псевдокод на функцията *Key Expansion()*

//--

```
Key Expansion (byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
    begin
     word temp
       i=0
         while(i<Nk)
            w[i]=word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
            i=i+1
         end while
            i=Nk
         while(i<Nb*(Nr+1))
            temp=w[i-1]
            if(i mod Nk=0)
                temp=SubWord(RotWord(temp)) xor Rcon[i/Nk]
            else if (Nk>6 and i mod Nk=4)
                temp=SubWord(temp)
            end if
```

```
                w[i]= w[i-Nk] xor temp
                i=i+1
            end while
    end
//-----------------------------------------------------------------
```

Първите N_k думи съдържат ключа за шифриране. Всички останали думи се определят рекурсивно от думите с по-малки индекси. Алгоритъмът за изработка на ключове зависи от величината N_k. А Фиг.13 а) можем да забележим, че пъвите N_k думи се запълват с ключа за шифриране. Всяка следваща дума w[i] се получва чрез XOR с предишната дума w[i-1] и думата на N_k позиция по-рано w[i-N_k]:

w[i]= w[i-1] XOR w[i-N_k].

За думите, позицията на които е кратна на N_k, преди XOR се прилага преобразуване към w[i-1], а след това се прибавя и рундовата константа *Rcon*. Преобразуването (преобразование) се реализира с помощта на две допълнителни функции: *RotWord()*, която осъществява побайтова отместване на 32- разрядната дума по формулата $\{a_0,a_1,a_2,a_3\}$-> $\{a_1,a_2,a_3,a_0\}$, и *SubWord()*, осъществяваща побайтова замяна с използване на S- блока от функцията *SubBytes()*. Значенето (стойността) на Rcon[j] е равно на 2^{j-1} . Стойността на w[i] в този случай е равно на:

w[i]= SubWord(RotWord(w[i-1])) XOR Rcon[*i/Nk*] XOR w[i-*Nk*].

W_0	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}	…

а)

Рундов ключ 0	Рундов ключ 1	Рундов ключ 2			

б)

Фиг.13 Процедури: а)разширяване на ключа б)избор на рундов ключ за Nk=4.

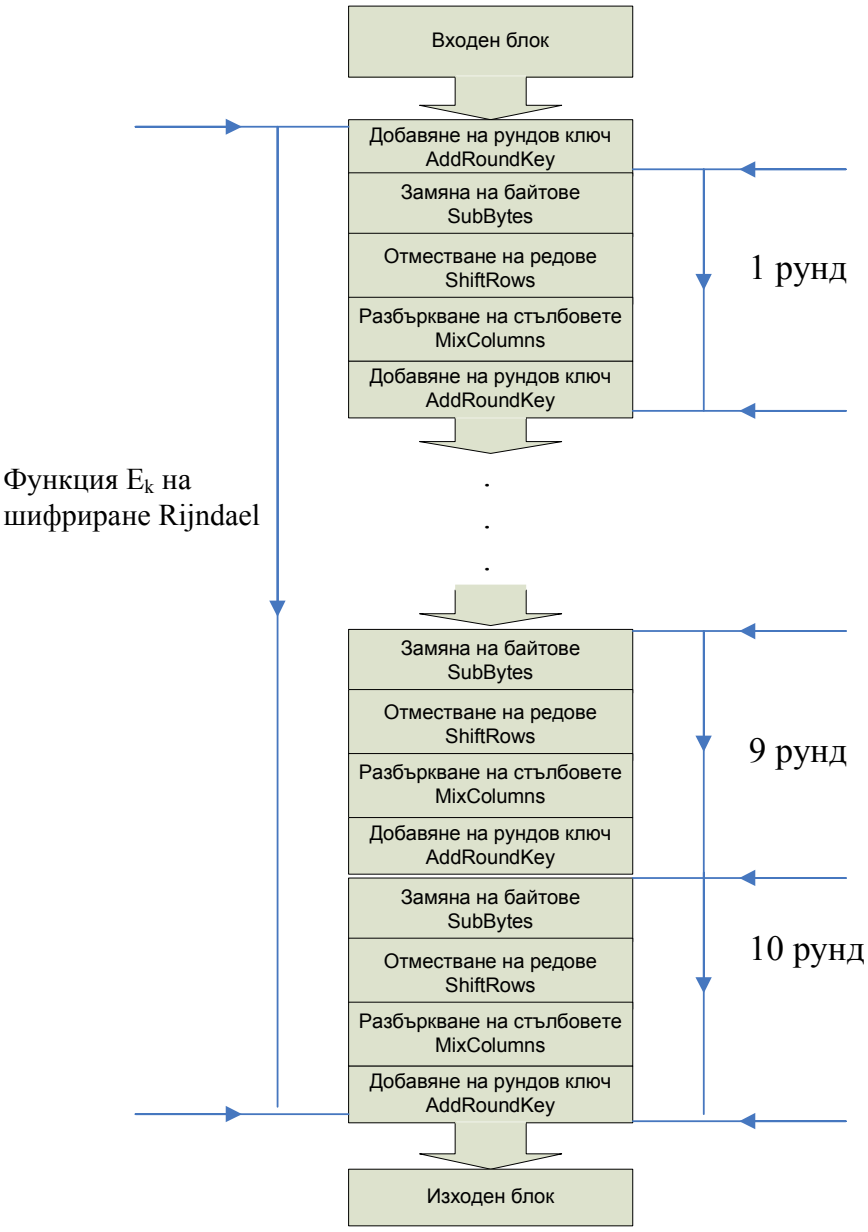
Избор на рундов ключ(Round Key Selection)

Забележка: Алгоритъмът за изработка на ключове може да се осъществи и без използването на масива w[i]. За реализациите, в които има съществено изискване към паметта, която се заема, рундовите ключове могат да се изчислят непосредствено чрез използване на буфер от N_k думи. Разширеният ключ винаги се получава от ключа за шифриране и никога не се указва направо. Няма никакви ограничения за избор на ключ за шифриране.

Фунция за шифриране

Шифър RIJNDAEL се състои от(Фиг.20):

1. първоначално добавяне на рундов ключ;
2. (Nr-1)рунда;
3. заключителен рунд, в който отсъства операцията *MixColumns()*;



Фиг.20 Схема на функцията E_K – за шифриране на криптоалгоритъма RIJNDAEL при N_k=N_b=4

На входа на алгоритъма се подават блокове от данни State, като в хода на преобразуването, съдържанието на блока се изменя и на изхода получаваме шифротекст, който е организиран пак във вид на блокове State (Фиг.21), където N_b=4, in_m и out_m - m са байтовете съответно на входния и изходния блок, m = 0,…,15, s_{ij} - байт, който се намира при пресичането на i-ти ред и j-ти стълб в масива State, i= 0,...,3 и j = 0,...,3.

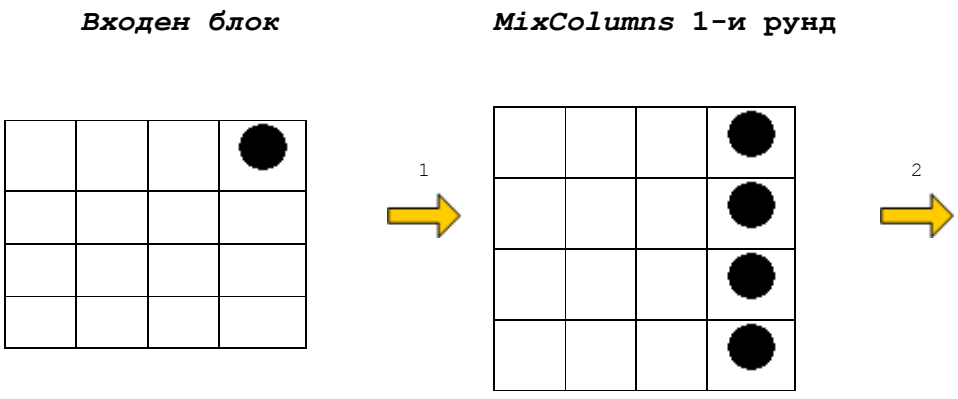
Фиг.21 Ход на преобразуването на данните, организирани във вид на блокове State

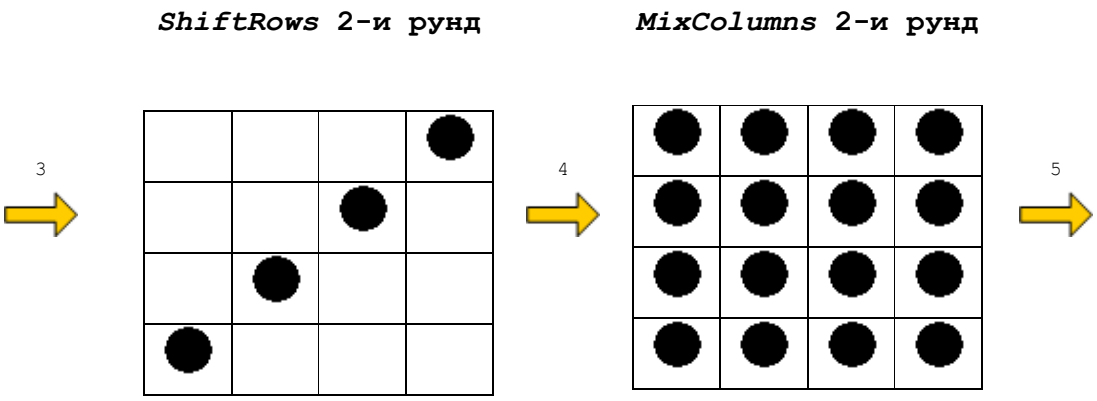
Преди началото на първи рунд се извършва сумиране по mod 2 с началния ключ за шифриране, след това- преобразуване на масива от байтове State в продължение на 10,12 или 14 рунда в зависимост от дължината на ключа. Последният рунд се отличава от останалите по това, че не се изпълнява функцията *MixColumns()* – разбъркване на байтовете в стълбовете.

```
//-------------------------------------------------------
                        Процедура за шифриране
//-------------------------------------------------------

Cipher ( byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
    begin
        byte state[4,Nb]
        state=in
        AddRoundKey(state, w[0, Nb-1])
            for round=1 step 1 to Nr-1
            SubBytes(state)
            ShiftRows(state)
            MixColumns(state)
            AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
            end for
            SubBytes(state)
        ShiftRows(state)
        AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
            out=state
    end
    //-------------------------------------------------------
```

Фиг.22 ни показва разсейващите(разпръскващите) и разбъркващите свойствата на шифъра. Вижда се, че 2 рунда са достатъчни за пълното разсейване и разбъркване.





Фиг.22 Принцип на действие на криптоалгоритъма RIJNDAEL.

◙ - изменен байт

Функция за разшифриране

Ако вместо *SubBytes()*, *ShiftRows()*, *MixColumns()* и *AddRoundKey()* в обратен ред изпълним инверсните им преобразувания, може да се построи функция за разшифриране. В този порядък(ред) използването на рундови ключове се явява обратен по отношение на този, използван за шифриране.

```
//-------------------------------------------------------------
                        Процедура за разшифриране
//-------------------------------------------------------------
InvCipher (byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
    begin
        byte state[4,Nb]
        state=in
            AddRoundkey(state, w[Nr*Nb, (Nr+1)*Nb-1])
            for round Nr-1 step -1 downto 1
                InvShiftRows(state)
                InvSubBytes(state)
                AddRoundkey(state, w[round*Nb, (round+1)*Nb-1])
                InvMixColumns(state)
            end for
        InvShiftRows(state)
        InvSubBytes(state)
        AddRoundkey(state, w[0, Nb-1])
```

```
            out=state
    end
//-------------------------------------------------------------
```

Ще се направи кратко описание на функциите, обратни на тези, които използваме при шифриране.

Фукцията *AddroundKey()* е обратна сама на себе си, отчитайки използваната в нея операция XOR.

Преобразуване *InvSubBytes*

Логиката на работа на инверсния S-блок при преобразуване на байта {xy} ще я покажем на Фиг.23 .

x	y															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Фиг.23

Преобразуване *InvShiftRows*

Последните 3 реда на дадено състояние циклично се отместват надясно на различно число байтове. Ред 1 се премества на C_1 байта, ред 2- на C_2 байта, ред 3- на C_3 байта. Стойностите(значенията) на C_1, C_2, C_3 зависят от дължината на блока N_b.

Преобразуване *InvMixColumns*

В това преобразуване стълбовете на състоянието се разглеждат като многочлени над $GF(2^8)$ и се умножават по mod x^4+1 на мнгочлена $g^{-1}(x)$, който изглежда по следния начин:

$$g^{-1}(x)= \{0b\}x^3+\{0d\}x^2+\{09\}x+\{0e\}.$$

Може да бъде представен и в матричен вид:

$$\begin{bmatrix} s'_{0c} \\ s'_{1c} \\ s'_{2c} \\ s'_{3c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{bmatrix}, 0 \le c \le 3.$$

В резултат на това, на изхода получаваме следните байтове:

$$\begin{aligned} s'_{0c} &= (\{0e\} \bullet s_{0c}) \oplus (\{0b\} \bullet s_{1c}) \oplus (\{0d\} \bullet s_{2c}) \oplus (\{09\} \bullet s_{3c}), \\ s'_{1c} &= (\{09\} \bullet s_{0c}) \oplus (\{0e\} \bullet s_{1c}) \oplus (\{0b\} \bullet s_{2c}) \oplus (\{0d\} \bullet s_{3c}), \\ s'_{2c} &= (\{0d\} \bullet s_{0c}) \oplus (\{09\} \bullet s_{1c}) \oplus (\{0e\} \bullet s_{2c}) \oplus (\{0b\} \bullet s_{3c}), \\ s'_{3c} &= (\{0b\} \bullet s_{0c}) \oplus (\{0d\} \bullet s_{1c}) \oplus (\{09\} \bullet s_{2c}) \oplus (\{0e\} \bullet s_{3c}). \end{aligned}$$

Функция за разшифриране

Алгоритъмът за разшифриране, описан по-горе има ред за прилагане на операциите, обратен на реда на операциите в алгоритъма за шифриране, но използва същите параметри. За операцията *MixColumns()* се описва:

InvMixColumns = (State XOR RoundKey)=

InvMixColumns(State) XOR InvMixColumns(RoundKey).

При формиране на ключ за шифриране в процедурата за "развиване" на ключа трябва да се добави следващия код:

```
//------------------------------------------------
for i=0 step 1 to (Nr+1)* Nb-1
```

```
 dw[i]=w[i]
end for
for round=1 step 1 to Nr-1
    InvMixColumns(dw[round*Nb, (round+1)* Nb-1])
end for
//-----------------------------------------------
```

Забележка: В последния оператор (в функцията *InvMixColumns()*) има преобразуване на тип, тъй като ключа се пази във вид на линеен масив от по 32-разрядни думи, като в същото време входния параметър на функцията е двумерен масив от байтове.

2.3 Основни особености на AES

1. Новата архитектура "Квадрат" , осигурява бързо разпръскване (разсейване) и разбъркване на информацията, като при това за един рунд на преобразуване се подлага целия входен блок.
2. Байт- ориентирана структура, която е удобна за реализация на 8-разрядни микроконтролери.

Акуалността на избраното решение се основава на постоянно повишаващият се интерес на обществото по въпросите за защита на информацията, и по-конкретно за актуалността на алгоритъма, приет неотдавна за стандарт AES (2001).

2.4 Обобщение

Изследванията, проведени в различни страни показват високо бързодействие на RIJNDAEL на различни платформи.

Основно свойство за този шифър се явява неговата байт-ориентираната структура, която обещава добри перспективи при неговата реализация в бъдеще на процесори или специални схеми. За недостатък може да се счита, че режимът на разшифриране се различава от режима на шифриране не само заради

последователността на функциите, но и самите функции се различават със своите параметри от използваните в режима на шифриране. Този факт се отразява на ефективността на апаратната реализация на шифъра. Такъв недостатък нямат шифрите на Фейстел(DES, ГОСТ 281147-89),в които при разшифриране се променя само реда на изпълнение на рундовите ключове. Но RIJNDAEL има по-цялостна структура, за един рунд в него се преобразуват всички битове на входния блок, за разлика от шифрите на Фейстел, при които за един рунд се изменят като по правило само половината битове на входния блок. Поради това RIJNDAEL може да си позволи по-малко число рундови преобразувания, което може да се разглежда като компенсация за загуби при реализация на режима за обратно шифриране. Шифърът е напълно самостоятелен – той не използва никакви части заимствани от други шифри, има отчетлива и ясна структура, тоест неговата здравина(устойчивост) не се основава на някакви сложни и не напълно разбираеми преобразувания. Последното свойство също намалява вероятността за съществуване на нйкакви “тайни проходи”. Гъвкавостта, заложена в архитектурата на RIJNDAEL, позволява да се варира не само за дължината на ключа, който се използва в стандарта AES, но и размера на блока на данните, които се преобразуват, макар и не е намерило приложение в стандарта, авторите го разглеждат като нормално разширение на този шифър.

Глава 3

Анализ на предметната област

Описание на предметната област

Системата, в зависимост от нейното предназначение е свързана с определена част от реалния свят, който се нарича *предметна област*.

Анализът на предметната област е необходим за построяване на *модел на предметната област* – формално описание на последната, в която ще бъдат отделени основните класове от обекти и типовете връзки между екземплярите на отделните класове.

Ще се определи използваната терминология:

Предприятието осъществява покупка на стоки (модули) от доставчик и продажба на стоки (изделия) на клиенти. От гледна точка на документооборота предприятието получава от доставчика приходен документ (приходен ордер) и дава (връща) на клиентите заедно със стоката разходен документ(разходен ордер). В отношенията си с клиентите и доставчиците(контрагентите) ще се вземе под внимание не само стоковите, но и паричните движения (заплащане към доставчиците и заплащане от клиентите). В самото предприятие също има стокови движения, поради наличие на складове: склад готова продукция, склад елементи.

В рамките на стопанската дейност предприятието има разходи от вида: цена на закупените стоки (модули), тяхната доставка, а също и разходи, които не са свързани с търговската дейност (пример: наем на помещения, заплата на служители и др.). Разходите, които не са свързани с търговската дейност ще се наричат *вътрешно-финансови разходи*.

Целият обем от реализирани стоки като сума ще представлява приход на предприятието. Данните за прихода и направените

разходи ще служат като показатели за формирането на финансовия резултат от дейността на предприятието.

3.2 Построяване на модел на предметната област

За построяване на модел на предметната област ще бъдат използвани диаграми на потоците от данни - DFD(Data Flow Diagrams). Диаграмите на потоците от данни описват смисълът на операциите в дадената предметна област. DFD може да се разглежда като граф, чрез който е показано движението на данните от техния източник, чрез преобразуващите ги процеси, до техните потребители в други обекти.
Ще се представят условните обозначения в диаграмите на потоците от данни- Фиг.24:

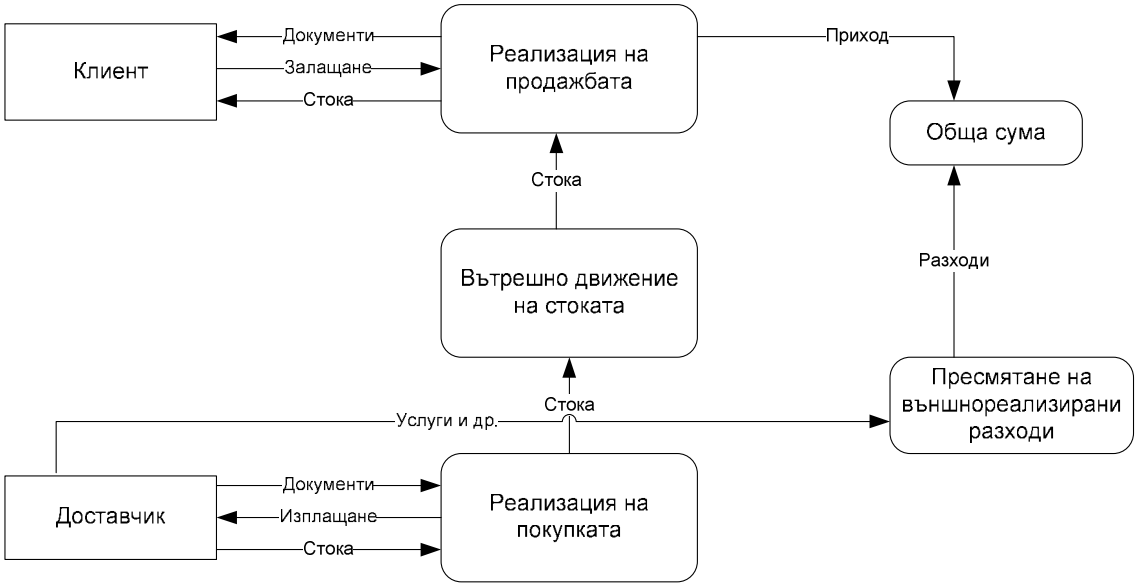
Фиг.24

DFD се състоят от следните елементи:

1. Външна същност- това са логическите класове от предмети или физически лица, които представляват източникът или приемникът на информация(пример: склад, клиент).
2. Логически процес, това е "механизъм", който приема входни потоци и ги преобразува в изходни, в съответствие със своята вътрешна логика. На всеки процес се дава име, което отразява неговата функция.

3. Информационни потоци. Разглежда се информационен поток , който представлява пътят, по който информацията от източникът се предава на приемникът.

Принцип на построяването на модел на предметната област: построяване на основната DFD диаграма на потоците от данни- първо ниво, което отразява общите моменти от работата на предприятието. След това се построява диаграмата на потоците от данни – диаграми от второ ниво, които уточняват процесите DFD от първо ниво. Този процес се продължава, докато достигнем определено ниво на опростеност.



Фиг.25 Основна диаграма на потоците от данни на предметната област

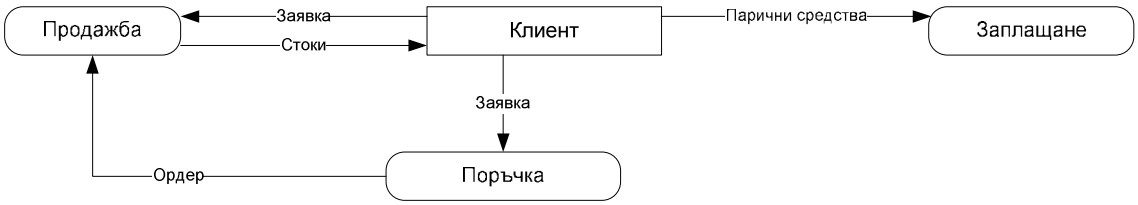
Основната DFD диаграма съдържа следните процесите:

1. *Реализация на продажбата*- процес, който взаимодейства с външната същност *Клиент* и е предназначен за оформяне на операциите, свързани с отпускане на стоки от склад.

2. *Реализация на покупката*- процес който взаимодейства с външния обект *Доставчик* и е предназначен за оформяне на операциите, свързани със закупуване на стока.
3. *Вътрешно движение на стоката- процес, който* реализира преместването на стоката до склад.
4. *Пресмятане на вътрешно-финансови разходи* – процес, който служи за ръководене на ревизията за разходи, които не са свързани с търговска дейност.
5. *Обща сума(финансова)*- процес, който формира вътрешната счетоводна отчетност на организацията.

3.2.1 Процес реализация на продажбите

На Фиг.26 е представена диаграма на потоците от данни второ ниво, детайлизираща процес ***Реализация на продажбите.***

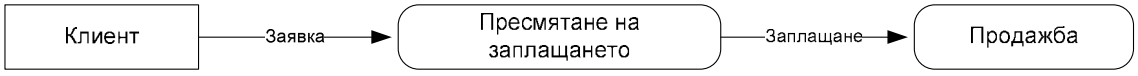


Фиг.26

Той е предназначен за оформяне на операциите, свързани с отпускане на стоки от склад- заявка на клиент, изпращане на стока, постъпване на финансови средства от клиент.

Процес Поръчка

На Фиг.27 е изобразена диаграма на потоците от данни трето ниво, което детайлизира процес ***Поръчка.***



Фиг.27

Процесът на поръчка е предназначен за фиксиране на предварителната договореност за купуване на стока от клиента, на чието основание може да се извършва заплащане и изпращане на стоката.

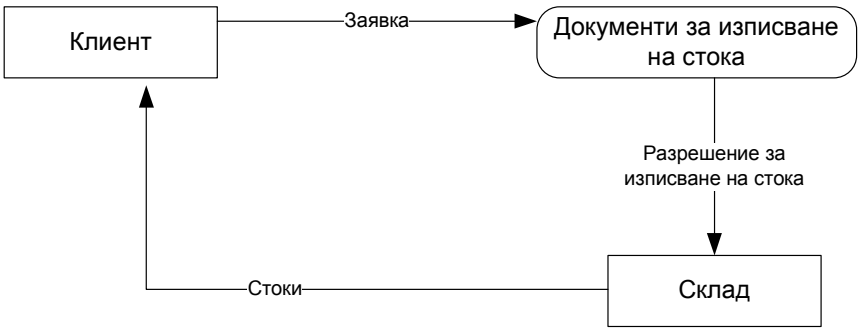
Процес ***Пресмятане на заплащането***:

Фази на процеса:

- Попълване на документ, на чието основание се извършва заплащане и изпращане на стоката (при изпращане автоматично се премахва "резервирането" на стоката).
- Фиксиране на предварително споразумение за получаване на стоката от купувача.

Процес Продажба

Диагарама на потоците от данни трето ниво, детайлизираща процес ***Продажба***(Фиг.28).



Фиг.28

Процес на *Продажба*- регистрира факт за изпращане на стока на клиент по заявка.

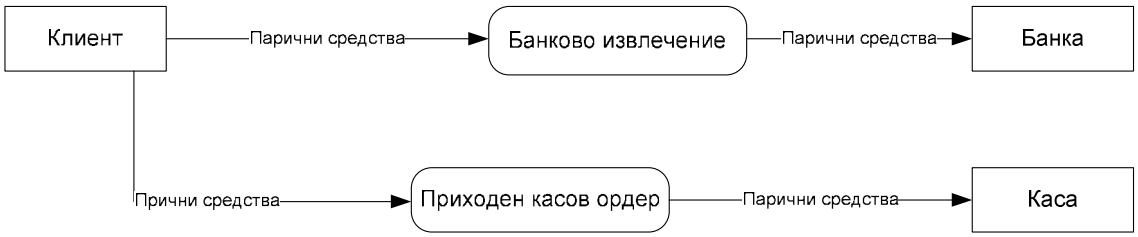
Процес ***Документи за изписване на стока***:

Фази на процеса:

- попълване на документите "Складова разписка" и "Фактура".
- оформяне на операциите, свързани с изписване(отпускане) на стоките от склад, при търговска дейност, свързана с продажбата на стоки.

Процес Заплащане

На Фиг.29 е изобразена диаграма на потоците от данни трето ниво, която описва детайлите на процес ***Заплащане.***



Фиг.29

Процесът на *Заплащане* се използва за оформяне на паричните постъпления от клиентите.

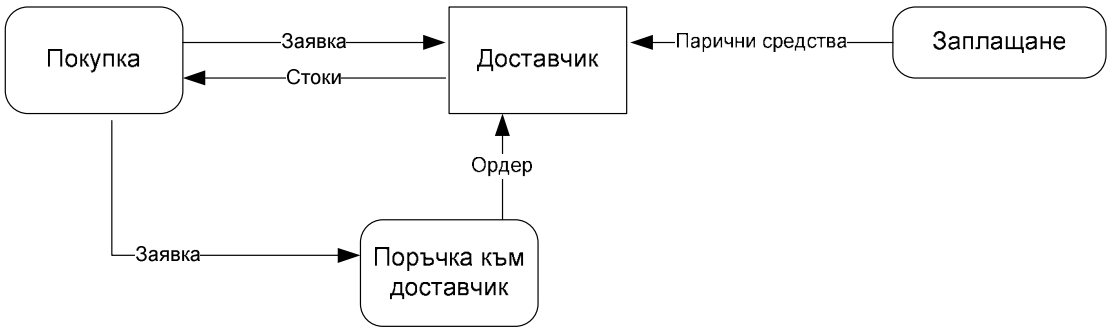
Процес ***Приходен касов ордер***:

Фази на процеса:

- попълване на документа „Приходен касов ордер”.
- пресмятане на постъпилите парични средства от клиент

3.2.2 Процес Реализация на покупката

На Фиг.30 е представена диаграма на потоците от данни второ ниво, която описва процес ***Реализация на покупката.***



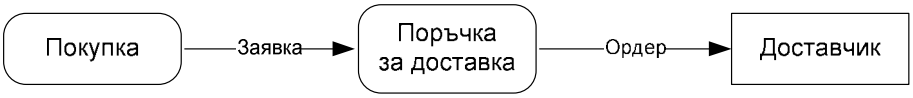
Фиг.30

Процес *Реализация на покупките* е предназначен за оформяне на операциите, свързани с покупката на стоки:

1. поръчка за доставка към доставчика;
2. вписване (заприходяване) на стоките;
3. заплащане на доставчика;

Процес Поръчка към доставчик

На Фиг.31 е изобразена диаграма на потоците от данни трето ниво, детайлизираща процес ***Поръчка към доставчик*** (Фиг.15).

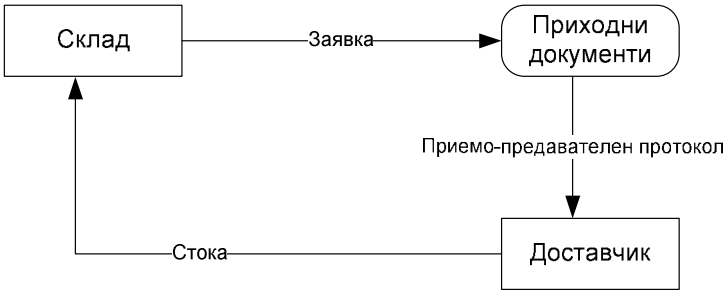


Фиг.31

Даденият процес е необходим за фиксиране на поръчаните от доставчика стоки.

Процес Покупка

На Фиг.16 е изобразена диаграма на потоците от данни трето ниво, описваща процес **Покупка.**

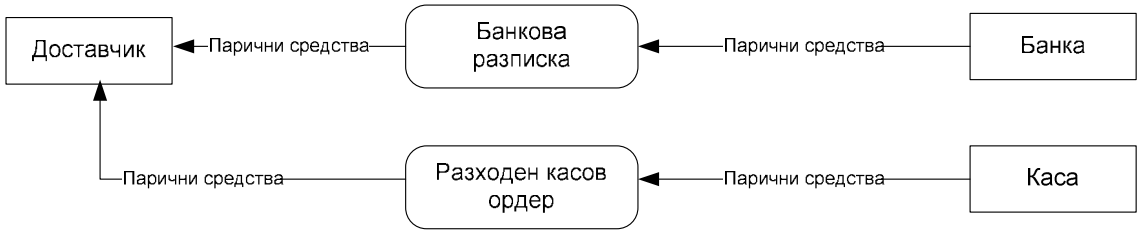


Фиг.31

Процес *Покупка* регистрира операциите, свирзани със заприходяване на стоки от доставчици.

Процес Заплащане

Диаграма на потоците от данни трето ниво, детайлизираща процес ***Заплащане***(Фиг.32).



Фиг.32

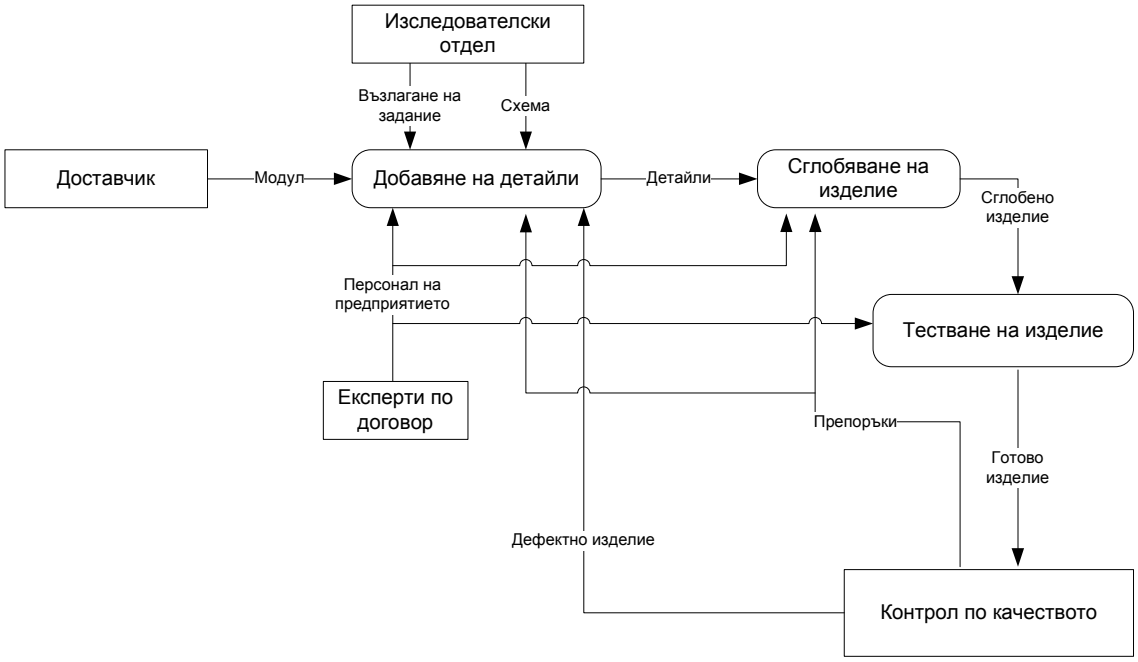
Процес *Заплащане* се използва за офомяне на "отпусканите" парични средства за заплащане на доставчика.

Процес ***Разходен касов ордер:***

Фази на процеса:

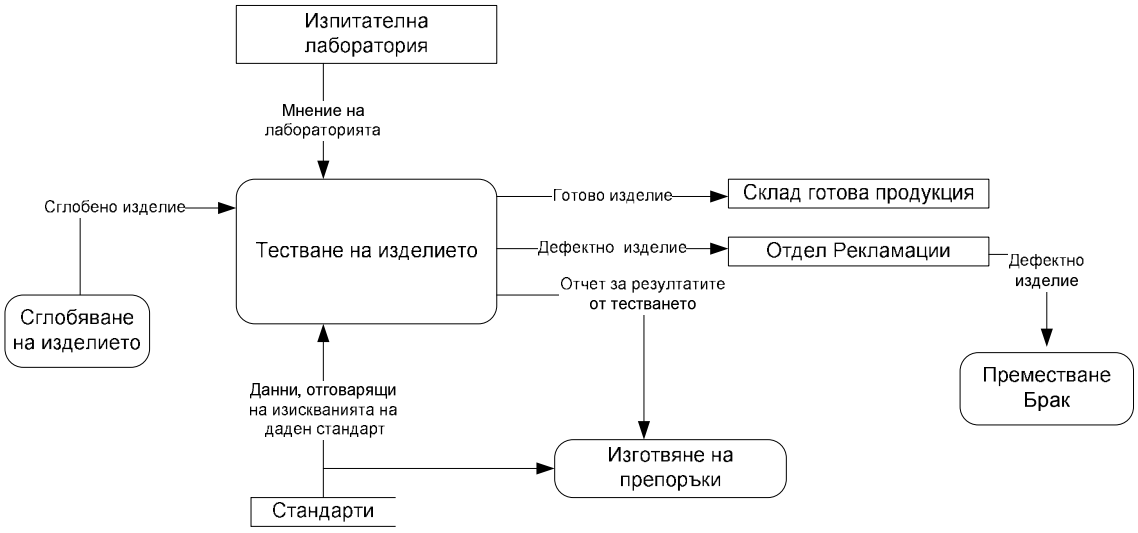
- Попълване на документа "Разходен касов ордер".
- Пресмятане на изплатените парични средства на доставчика, в качеството на заплащане за изпратената стока и предоставените услуги.

Процес Вътрешно движение на стоката (изделие)



Фиг.33

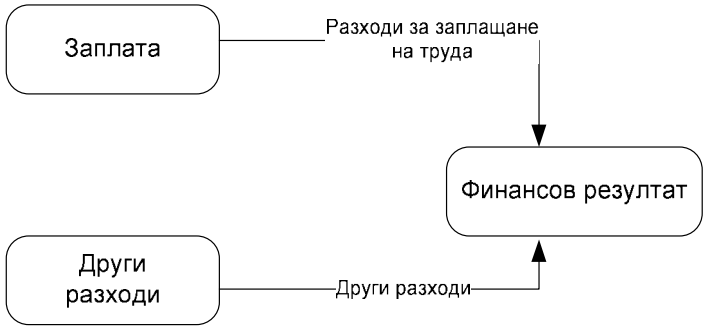
Процес Тестване на готово изделие



Фиг.34

Процес Пресмятане на вътрешно-финансови разходи

На Фиг.35 е изобразена диаграма на потоците от данни второ ниво, описващ процеса на пресмятане на ***Вътрешно-финансови разходи***.



Фиг.35

Процес *Пресмятане на вътрешно-финансови разходи* показва вътрешната счетоводна дейност на предприятието. Обединява в себе си операционни и аналитични отчети. Към операционните отчети се отнасят, например, отчет за движението на паричните средства в банката, отчет по разплащането с клиенти. Към основните аналитични отчети се отнасят отчети за продажбите, отчет за доходност на стоките, анализ на цените.

Процес ***Заплата***:

- попълване на документа "Разходен касов ордер" (за изплащане на заработеното възнаграждение).
- фиксиране на разход на парични средства за изплащане на заработено възнаграждение.

Процес ***Други разходи***:

- разход на парични средства за стопански и други нужди.

Глава 4

Анализ на представянето на моделите на данните

4.2 Избор на логически модел

4.2.1 Йерархичен модел на данните

При този модел данните се съхраняват в предварително опеределена йерархия. Йерархичното дърво е обърнато с корена нагоре. Едно от предимствата на този модел е бързият достъп до данните, но за да се създаде една пълна и работеща БД и да се осигурят връзките между отделните данни, може да се наложи съхраняване на излишна информация.

Релацията в една йерархична БД се базира на термини като родител/наследник. При тази релация всяка родителска таблица може да бъде асоциирана с повече от една дъщерни таблици, но една дъщерна таблица може да бъде асоциирана само с една родителска таблица. Тези таблици са изрично свързани чрез указател или чрез физическата подредба на записите в таблиците.

Потребителят може да осъществи достъп до данните в рамките на модела, като започва от таблицата-корен и обхожда дървото надолу, докато достигне до исканите(търсените) данни.

Предимства: бърз достъп до данни- съществуването на изрични връзки между структурите на таблицата. Цялостта на връзките е вградена и се прилага автоматично- запис в дъщерната таблица трябва да е свързан със съществуващ запис в родителската таблица; изтрит запис в родителската таблица, ще доведе до изтриване на асоциирани с него записи в дъщерната таблица.

Недостатъци: за да се създаде една пълна и работеща БД и да се осигурят връзките между отделните данни, може да се наложи съхраняване на излишна информация. На логическо ниво това

опростява достъпа, но на физическо ниво се съхранява излишна информация[2].

4.2.2 Мрежов модел на данните

Основните компоненти в този модел са записи и съвкупност от записи. Обектите от една предметна област са обединени в "мрежа" (множество). Базата данни се състои от няколко множества, а всяко множество се състои от записи. Една съвкупност от записи може да принадлежи на няколко мрежи.

При този модел данните и връзките между тях се представят чрез граф, върховете на който са типове записи, дъгите- типове връзки. Графичното изображение на графа се нарича структурна диаграма на данните. Моделът се представя с помощта на *възли и свързващи структури*. Всеки възел представя колекция от записи, а всяка свързваща структура създава и представя релация в мрежова база данни. Чрез структурната диаграма се свързват два възела, като единият възел е *собственик*, а другият възел е *член*. Всяка свързваща структура поддържа релация от тип едно към много, което означава, че един запис във възела собственик може да бъде свързан с един или повече записи във възела-член, но един единствен запис във възела-член може да бъде свързан само с един запис във възела –собственик.

Предимства: бърз достъп до данните- потребителите могат да създават заявки, които са по-сложни от тези при йерархичните БД.

Недостатъци: работата в този модел се утежнява от сложната логическа структура на данните, за да се осъществи достъп до всички записи; потребителят трябва да бъде много добре запознат със структурата на базата данни, за да може да обхожда свързващите структури; структурата не може да се променя лесно, тъй като това влияе на приложните програми, които взаимодействат с нея [2].

4.2.3 Релационен модел на данните

Релационният модел на данните е предложен от Е.Код през 1970 година. Този модел има две важнии характеристики:

1. Прост и естествен начин за представяне на данните- таблици;
2. Строга математическа основа на модела- теория на множествата и предикатната логика;

Всяка таблица се състои от редове и колони, като значенията във всяка колона са сравними- принадлежат на едно множество от допустими значения. Всяка колона има наименование, което показва какъв е смисъла на значенията в колоната.

От дефиницията на релацията могат да се определят свойствата:

1. в релациите няма еднакви редове;
2. редовете в една релация са ненаредени;
3. атрибутите в една релация са ненаредени;
4. стойността на всеки атрибут е атомарна(неделима);

Релацията, която притежава последното свойство се нарича *нормализирана*.

Често таблицата се използва като изображение на абстрактното понятие релация. При такова представяне се внасят някои свойства, които не са верни според формалната дефиниция. В таблицата редовете са наредени отгоре надолу, също както и колоните- отляво надясно. Когато се говори неформално понятията релация и таблица са синоними, както и понятията атрибут и колона. Това е едно от най-големите преимущества на релационния модел- едно формално понятие има просто неформално представяне. В езика SQL се използват неформални понятия- таблица, колона, ред.

Релационна база данни(РБД)- това е съвкупност от изменящи се във времето релации от различна степен. РБД, използваща релационен модел се създава от атрибути на таблици. Тези таблици (релации) са прости плоски файлове, в които редовете съответстват на записи, а колоните са полета (атрибути). Всяка

таблица има първичен ключ, който се използва за достъп до данните.

Предимствата на релационния модел може да се разделят на две групи:

I. Предимства за потребителя:
   1. Релационната БД представлява набор от взаимосвързани таблици, с които потребителят е привикнал да работи;
   2. Не е нужно да запомня пътя за достъп до данните и да измисля алгоритми и процедури, за да обработи своята заявка;
   3. Релационните езици са прости (леки) за изучаване и усвояване, като в същото време езиците за общуване с йерархичния модел и мрежовия модел са по-малко пригодени за потребители;

II. Предимства при обработка на данните на релационната БД:
   1. Свързаност- релационното представяне на данните дава ясна картина на взаимната връзка на атрибутите на различните отношения.
   2. Независимост на данните – БД трябва да допуска възможност за разширяване, тоест да се добавят нови атрибути и отношения.
   3. Други

Йерархичният и мрежовият модел са по-ефективни при процеси с голям обем данни, а релационният- при по-сложни и разнообразни данни. Релационните модели не изискват поддръжка на указатели, както при останалите два модела. Но релационните модели, които не са съгласувани с нормализационни правила могат да дублират данни в различни таблици. За да се направят данните по-ефективни много такива модели не са съгласувани напълно с тези правила.

Поради това, че от изброените логически модели на данните релационният притежава предимства и малко недостатъци, тя ще бъде избрана за решението на задачите, поставени в началото на настоящата дипломна работа.

4.3 Избор на концептуален модел

За избора на концептуален модел, се разглеждат следните разновидности:

1. Семантичен модел- основава се на построяване на семантична мрежа. Под семантична мрежа ще разбираме ориентиран граф, който се състои от възли и дъги, като се задават обектите и отношенията на предметната област. Семантичните мрежи имат редица преимущества:
    a. Описанието на обектите на предметната област става посредством естествен език.
    b. Всички записи, които постъпват в БД се натрупват(събират) в относително еднородна структура.

Въпреки изброените преимуществата, семантичния модел на данните притежава и редица недостатъци, като най- съществен от тях е, че построяване на релационен модел на данните на основата на семантичните мрежи е по-сложно.

2. Фреймов модел- фреймът е структура за представяне на знания. Фреймът съдържа информация за компонентите на описваното понятие, връзки с други подобни понятия, начините за достъп до самия него и как се променя във времето.
3. Модел "същност-връзка" (ER модел)- неформален модел – използва неформални, интуитивни понятия. Този модел се базира на основни понятия, като:
    a. обект (entity- същност)- нещо, което съществува реално, представлява интерес за ПО и за което ще се съхранява информация в БД. Обект може да е нещо конкретно(например: клиент, доставчик и др.) или абстрактно понятие.

b. тип обекти (entity set- клас обекти)- множество от всички подобни обекти в ПО, т.е обекти притежаващи общи свойства.
c. свойство (attribute- свойство)- обектите се характеризират със свойства- свойството е някакъв факт за обекта(например:име, пол, длъжност и др.).
d. връзка (relationship)-свързва или асоциира обекти (2,3 или повече) от различни класове или от един и същи клас. Най-често всяка връзка свързва два обекта.
e. тип връзки (relationship set)- множество от всички подобни връзки, които съществуват между два или повече класа обекти и имат еднаква семантика.
f. Видове връзки- съществуват няколко вида, в зависимост от това, обект от единя клас с колко обекта от другия може да бъде свързан в определен момент и обратно. Връзките се разделят на: 1:1(едно към едно), 1:M(едно към много), M:M (много към много). **ER- диаграма**- графично представяне на връзките между същностите(обектите).

Тъй като ER моделът е по-близък по принципи на организацията до релационния модел, и реализацията на последния на базата на първия е най-удобна, то в качеството на концептуален модел се избира модел "същност- връзка".

4.3 Избор на конкретна СУБД

Проектирането на БД на основа на релационния модел има редица преимущества пред останалите модели:

a. независимост на логическата структура от физическото й потребителското представяне.
b. гъвкавост на структурата на БД- конструктивните решения не ограничават възможностите на разработчика на БД да изпълнява в бъдеще най-разнообразни заявки.

MySQL

Сравнявайки СУБД MySql с други бази данни, трябва да се вземе под внимание кой фактор се явява най-важен за нас - производителност, поддръжка, възможности, ограничения, цена. Взимайки под внимание всички тези съображения, и най-вече производителност и възможности, СУБД MySql има предица преимущества:

4. Бързодействие- достатъчно бързодействаща СУБД. Разработчиците са на мнение, че MySQL се явява една от най-бързите БД на днешния пазар.
5. Простота на използване- СУБД MySql се явява високопроизводителна и относително проста в използването си СУБД, която е лесна за инсталиране и администриране, в сравнение с други системи.
6. Поддръжка на език за заявки- MySql "разбира" командите на езика SQL, език за заявки, който намира приложение във всички съвременни СУБД.
7. Възможности- Сървърът позволява едновременно да се включват неограничен брой потребители. Достъп до сървъра на СУБД MySql може да се осъществи в интерактивен режим, с помощта на различни интерфейси, позволяващи да се въвеждат заявки и да се разглеждат получените резултати(програми-клиенти, работещи с command line,Web-браузери и др.). Достъпът до БД е възможен и с помощта на приложения, които поддържат

ODBC (открито вазимодействие с базите данни) протокола за връзка с БД , разработен от компанията Microsoft. По този начин може да се използва както готово клиентско програмно осигуряване, така и да си създадем наше собствено.

8. Взаимодействие и безопасност- MySql е предназначена за работа в мрежа и може да бъде достъпна чрез Internet. По този начин с данните може да се работи, от която и да е точка на земното кълбо. Затова СУБД MySql е снабдена с развита система за защита от несанкциониран достъп.
9. Малък размер- СУБД MySql има ограничен размер, особено в сравнение с огромните дискови пространства, необходими за повечето търговски СУБД.
10. Работоспособност и цена- MySql се явява Open Source, достъпна при условията на GNU General Public License (GPL). Това означава, че СУБД MySql се разпространява безплатно за домашни потребители.
11. Свободно разпространение

4.5 Избор на език за манипулиране на данните

Релационните езици за работа с бази от даннни могат да се разделят на няколко типа:

1. Релационна алгебра- ISBL (Information System Base Language) в PRTV на IBM.
2. Релационно смятане- QUEL в INGRES на университета в Беркли.
3. Език SQL (Structured Query Language)- с елементи на релационното смятане и релационната алгебра.

Всички релационни езици имат една обща черта. Всеки оператор действа не върху един, а върху много редове и резултатът е множество редове (често релация). Всички релационни езици са еднакво мощни- всяка операция, която може

да се изрази на един език, ноже да бъде изразена и в другите, което се нарича релационна пълнота.

4.5.1 Език SQL

Езикът SQL притежава елементи от релционното смятане и релационната алгебра. Създаден в IBM, където през 1974-75 година се разработва една от първите релационни СУБД System/R. Езикът, създаден за тази система, е наречен SEQUEL(Structured English Query Language). След експлоатация, тестване и развитие на езика, неговото име е променено- SQL и проекта System/R завършва. Така започва историята на езика SQL, който сега е стандарт за релационните СУБД.

През 1982 година към ANSI е сформиран комитет за стандартизация на SQL, а през 1986 г. е приет първия стандарт SQL X3.135, който през 1987 година е приет от ISO. Тъй като езикът се развива, стандартизацията му продължава. По- късно са приети следващите стандарти – SQL/89 или SQL1, SQL/92 или SQL2 и SQL3. Въпреки многото стандарти, съществуват и много промишлени диалекти, които се различават един от друг. Повечето съвременни промишлени СУБД реализират езика в съответсвие с SQL2, като го разширяват с нови възможности.

Глава 5

Архитектура на разработваната система

За целите на настоящата дипломна работа, ще се използва модел на сигурност, базиран на роли, при който първо се установява идентичността на потребителя и след това се дават права за достъп до даден ресурс, в зависимост от съответната идентичност.

Взето е решение за използване на MySQL Server 5.0, базирайки се на редицата преимущества, изброени в глава 4. *"Анализ на представянето на моделите на данните"*. За достъп до базата данни ще използваме *MySQLDirect NET Data Provider (доставчик на данни)*.

Поради поставените изисквания, за бърз достъп до базата данни към проекта ще бъдат използвани: .NET Framework, ADO.NET и Windows Forms.

ADO.NET е технология, която .NET Framework ни предоставя за достъп до данни. Това е набор от библиотеки за работа с данни, които включват класове, интерфейси, структури и други типове, които са предназначени за достъп до различни източници на данни. Тя се състои от компоненти, които могат да бъдат имплементирани за различни видове бази данни – както релационни, така и нерелационни. За достъп до различните видове бази данни се използват така наречените доставчици на данни (data providers). Те са специфични за съответната база, но спазват програмния модел на ADO.NET като имплементират дефинираните в него интерфейси[5].

Всеки доставчик на данни съдържа четири основни класа, чрез които се осъществява достъпа до съответната база. Те ни осигуряват:

1. **Връзка към базата данни** (Connection)

Класът Connection осъществява връзката към базата данни. Този клас ни предлага методите Open() и Close(), с помощта на които ще отваряме и затваряме връзката.

2. **Изпълнение на команди** (Command)
   Класът Command представя заявка към базата данни. Command има свойството CommandText, което съдържа SQL заявка. Ако се очаква заявката да не върне резултат (например: при INSERT, DELETE),то тя ще бъде изпълнена с метода ExecuteNonQuery(). Ако трябва да върне само една стойност – ще се използва метода ExecuteScalar(). Ако се очаква резултатът да бъде таблица (един или повече редове), заявката ще бъде изпълнена чрез метода ExecuteReader(), който връща DataReader обект.
3. **Поточно извличане на данни** (DataReader)
   Класът DataReader се използва за извличане на данни. Той осигурява последователен достъп до данните от една или повече таблици.
4. Други

В настоящата работа се използва *MySQLDirect*, разработка на фирмата CoreLab, Украйна. Основните класове на използаният от нас *MySQLDirect NET Data Provider са : MySqlConnection, MySqlCommand, MySqlDataReader.*

5.1 Слоеве на приложението

В настоящата дипломна работа се използва класическата трислойна архитектура,тоест:

1. **Database Layer** – достъп до базата данни;
   Капсулира се достъпа до базата данни. Чрез него се изпълняват всички заявки към сървъра на базата данни. Тук единствено отваряме и затваряме връзките към базата. За да създадем връзка към нея е необходимо да се зададат параметри(адрес на сървър, име на базата данни, парола, порт). За тази цел използваме свойството

ConnectionString на класа на връзката, като го съхраняваме в конфигурационен файл app.config по следния начин:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <configSections>
    </configSections>
        <connectionStrings>
                        <add
                        name="SysDoc.Properties.Settings.ConnectionString"
                        connectionString="Provider= CoreLab.MySql;
                        Database=asydb;User ID=root; Password=123456;
                        Host=localhost;Port=3306;"
                        providerName="CoreLab.MySql" />
        </connectionStrings>
</configuration>
```

Връзката към базата данни се затваря, когато работата с нея приключи. За да сме сигурни, че при възникване на изключение тя ще бъде затворена, се използва конструкцията try, catch, finally.

2. **Business Layer** – тук данните се обработват по специфичен за приложението начин. В него е съсредоточена програмна логика за специфично поставен проблем.
   Тук се валидират данните, тоест дали са тези, които очакваме,и дали са в нужния формат.
3. **Presentation Layer** – служи за визуално представяне на данните, предоставяйки възможност на потребителя да оперира с тях по удобен и лесен за него начин.
   Съдържа класове и функционалност, с помощта на която данните се показват коректно на потребителя.
   Интерфейсът е проектиран да бъде максимално опростен и интуитивен.

5.2 Сигурност, базирана на роли

Системата за сигурност на .NET Framework ни предлага вграден модел за сигурност:

1. **Сигурност, базирана на роли** - потребителите се категоризират в групи, на основата на общи бизнес функции или отговорности. На тези групи от потребители се дават набори от права, така че всички потребители от дадена група да могат да изпълняват само необходимите им функции. Съвкупност от потребители , които имат едни и същи права ще наричаме роля. Правата, които се дават на ролите могат да бъдат използвани за управление достъпа до обектите и методите. Ако на определена роля се даде достъп до обект или до метод на този обект, означава че този достъп е даден на всички членове на тази роля.

   Ролите могат да заменят даването на определени права на отделните потребители.

Тук ще уточним понятията:

1. **Идентификация** – процес, чрез който се установява идентичността на даден потребител. Първо се сверяват данните, въведени от потребителя, с тези от сигурен източник (например: информация за потребител в Active Directory, база данни или Microsoft Passport Account). Ако данните не се потвърдят, то процесът на идентификация е неуспешен и потребителят има идентичност на анонимен потребител. Този процес е единственият възможен начин за определяне идентичността на потребителя – проверка с помощта на въведени от него данни, които могат да са под формата на потребителско име и парола.
2. **Оторизация.** – получаване на права за достъп до заявен ресурс; следва след идентификацията.

Метод на оторизация, поддържан от NET Framework – файлова оторизация – съпоставя искания ресурс с физически файл. NTFS използва списъци за контрол на достъпа (ACLs, AccessControlLists), за да защити ресурсите на файловата система. Този списък указва кои потребители имат достъп до ресурси и права, като четене, писане промяна, триене. Ако идентифицираният потребител има право на достъп до заявения ресурс, казваме че той е оторизиран.

В .NET Framework потребителите са представени от идентичности, а групите от роли. Тогава участник ще наричаме идентичността на потребител и ролята, към която той принадлежи. Тези идентичности съдържат свойства, които дават възможност за достъп до информация за потребител (например: потребителско име).

Глава 6

Описание на реализацията

Реализацията описва процедура за достъп до база данни от приложение за Windows. Тук ще бъде направен анализ и дизайн на приложението.

6.1 Постановка на задачата

Трябва да бъде разработено приложение, което позволява да се взимат подробности за служителите, клиентите, изделията и поръчките от разработената база данни и да се показват подробности за тях в контрол DataGridView.

В настоящата работа ще бъдат представени множество контроли като : DataGridView, Button, ToolStrip. Тук ще бъде показан начина, по който ще бъде прочетена базата данни от XML файл, опитна постановка за шифриране на информация чрез алгоритъма за криптиране AES, както и опитна постановка за сигурност, базирана на роли.

6.2 Жизнен цикъл на проекта

След подробно анализиране на изискванията, при което се определят и документират нуждите на клиента и ограниченията, които той поставя се създава план за разработване на приложението. Събира се цялата информация от клиента и се преценяват неговите нужди. В края на този етап се изготвя описание на проблема на клиента и начини за неговото решаване.

В конкретният случай, при анализиране на проблема се съставя следния списък от проблеми на компанията:

1. Компанията има нужда от създаване на база данни за съхраняване и организиране на информацията.
2. Информацията за всеки служител, клиент, както и за всяка поръчка и изделие трябва да е лесно достъпна.

3. Компанията трябва да анализира данните.
4. На базата на анализираните данни компанията трябва да намери начин да обслужва по- добре клиентите си.

Като решение на поставените проблеми се предлага да се създаде приложение за Windows, което да комуникира с базата данни и да показва данните от нея във форми.

6.3 Проектиране на базата данни

6.3.1 Необходимост от база данни

Базата данни е хранилище за данни, мястото където/от което може да се съхраняват/ извличат данни, когато е необходимо. Съществуват много начини за съхранение и извличане на данни. Един от начините, използван в настоящата дипломна работа е използването на SQL заявки.

6.3.2 SQL заявки

Ще разгледаме основните заявки, които се използват:

1. **Заявка Select** – прави се "запитване" към базата.

Пример: Заявката да извлича всички поръчки от един клиент

```
SELECT cust_company, ord_incn,ord_date
FROM cust, ord, orcu
WHERE orcu_ord_id = ord_id
    AND cust_id = orcu_cust_id
    AND orcu_cust_id = ( SELECT distinct( cust_id )
                         FROM cust, od, odoo, orcu
                         WHERE od_id = odoo_od_id
                             AND odoo_id = orcu_id
                             AND orcu_cust_id = cust_id
                             AND cust_company like '%Mir%' );
```

2. **Заявка Update –** променя данните в таблица(актуализиране на данни).
3. **Заявка Insert –** добавя се ред в таблица.

4. **Заявка Delete –** изтрива се ред от таблица/и.

6.3.3 Първични и външни ключове

За да има достъп до данните, съхранени в таблици, трябва да може да се различава уникално всеки от редовете на таблицата. Затова идентификаторът трябва уникално да определя всички данни в таблицата. В конкретния случай за таблицата Employee се създава допълнителна колона, която има уникална стойност за всеки от редовете в таблицата – (пример: EMP_id). Този уникален идентификатор се нарича първичен ключ. Аналогично се определят първичните ключове за останалите таблици.

Външен ключ – се нарича колона или няколко колони, които задават връзка между двете таблици. Добавянето на колоната с първичните ключове на едната таблица към друга таблица задава връзката между двете таблици (например: в таблица EMP първичният ключ е EMP_id, докато в таблица EMPD този атрибут е с наименование: EMPD_EMP_id).

6.3.4 Нормализация

Нормализацията е процес на намаляване на излишъка от данни. Представлява разделяне на данните в две или повече таблици, докато повтарящите се групи от данни се позиционират в две отделни таблици.

В случая постигнатите резултати са:

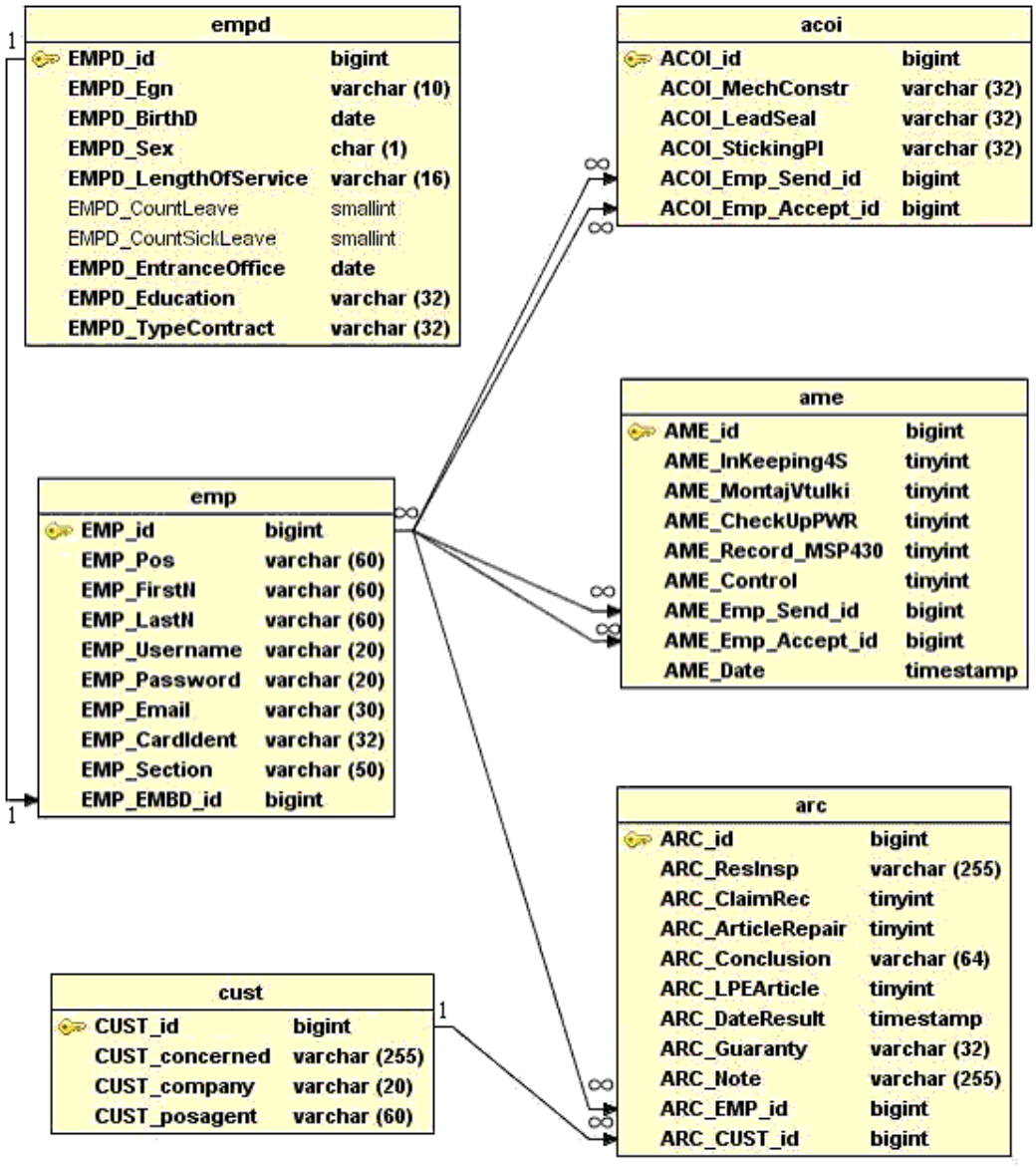
1. Всички таблици имат уникален идентификатор.
2. Таблиците не съдържат повтарящи се стойности или колони.
3. Всички таблици съдържат данни за еднотипни обекти.
4. За всички таблици се избягват колони, позволяващи празни стойности.

6.3.5 Дизайн на базата данни

След прилагането на всички концепции описани дотук, се получава структурата на базата данни във вида, показан на фиг. 36:

6.3.5.1 ER - диаграма на базата данни

Означение	Вид връзка
1 : 1	Едно към едно
1 : ∞	Едно към много
∞ : ∞	Много към много



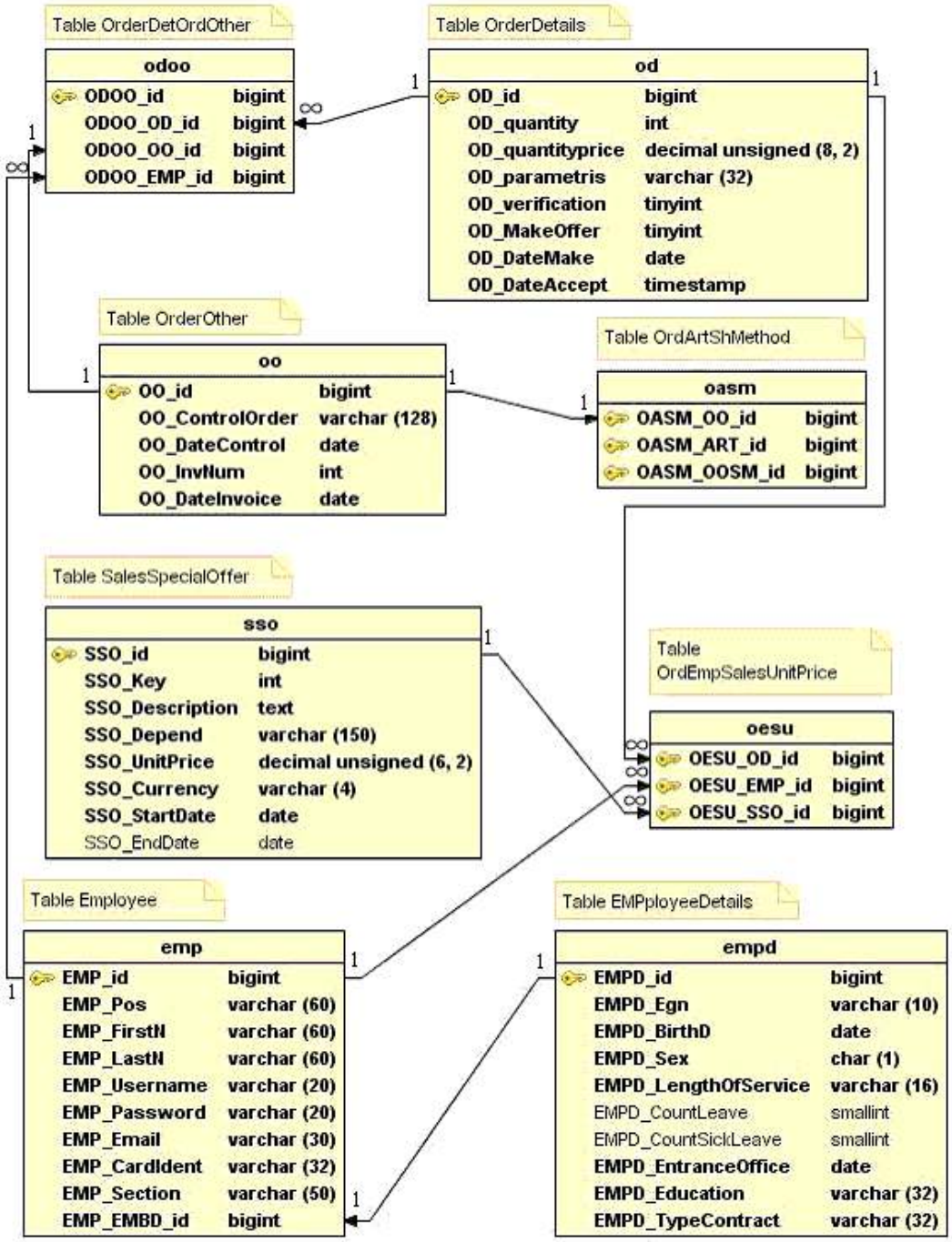


Table OrderDetOrdOther
odoo
ODOO_id bigint
ODOO_OD_id bigint
ODOO_OO_id bigint
ODOO_EMP_id bigint
Table OrderDetails
od
OD_id bigint
OD_quantity int
OD_quantityprice decimal unsigned (8, 2)
OD_parametris varchar (32)
OD_verification tinyint
OD_MakeOffer tinyint
OD_DateMake date
OD_DateAccept timestamp
Table OrderOther
oo
OO_id bigint
OO_ControlOrder varchar (128)
OO_DateControl date
OO_InvNum int
OO_DateInvoice date
Table OrdArtShMethod
oasm
OASM_OO_id bigint
OASM_ART_id bigint
OASM_OOSM_id bigint
Table SalesSpecialOffer
sso
SSO_id bigint
SSO_Key int
SSO_Description text
SSO_Depend varchar (150)
SSO_UnitPrice decimal unsigned (6, 2)
SSO_Currency varchar (4)
SSO_StartDate date
SSO_EndDate date
Table OrdEmpSalesUnitPrice
oesu
OESU_OD_id bigint
OESU_EMP_id bigint
OESU_SSO_id bigint
Table Employee
emp
EMP_id bigint
EMP_Pos varchar (60)
EMP_FirstN varchar (60)
EMP_LastN varchar (60)
EMP_Username varchar (20)
EMP_Password varchar (20)
EMP_Email varchar (30)
EMP_CardIdent varchar (32)
EMP_Section varchar (50)
EMP_EMBD_id bigint
Table EMPployeeDetails
empd
EMPD_id bigint
EMPD_Egn varchar (10)
EMPD_BirthD date
EMPD_Sex char (1)
EMPD_LengthOfService varchar (16)
EMPD_CountLeave smallint
EMPD_CountSickLeave smallint
EMPD_EntranceOffice date
EMPD_Education varchar (32)
EMPD_TypeContract varchar (32)

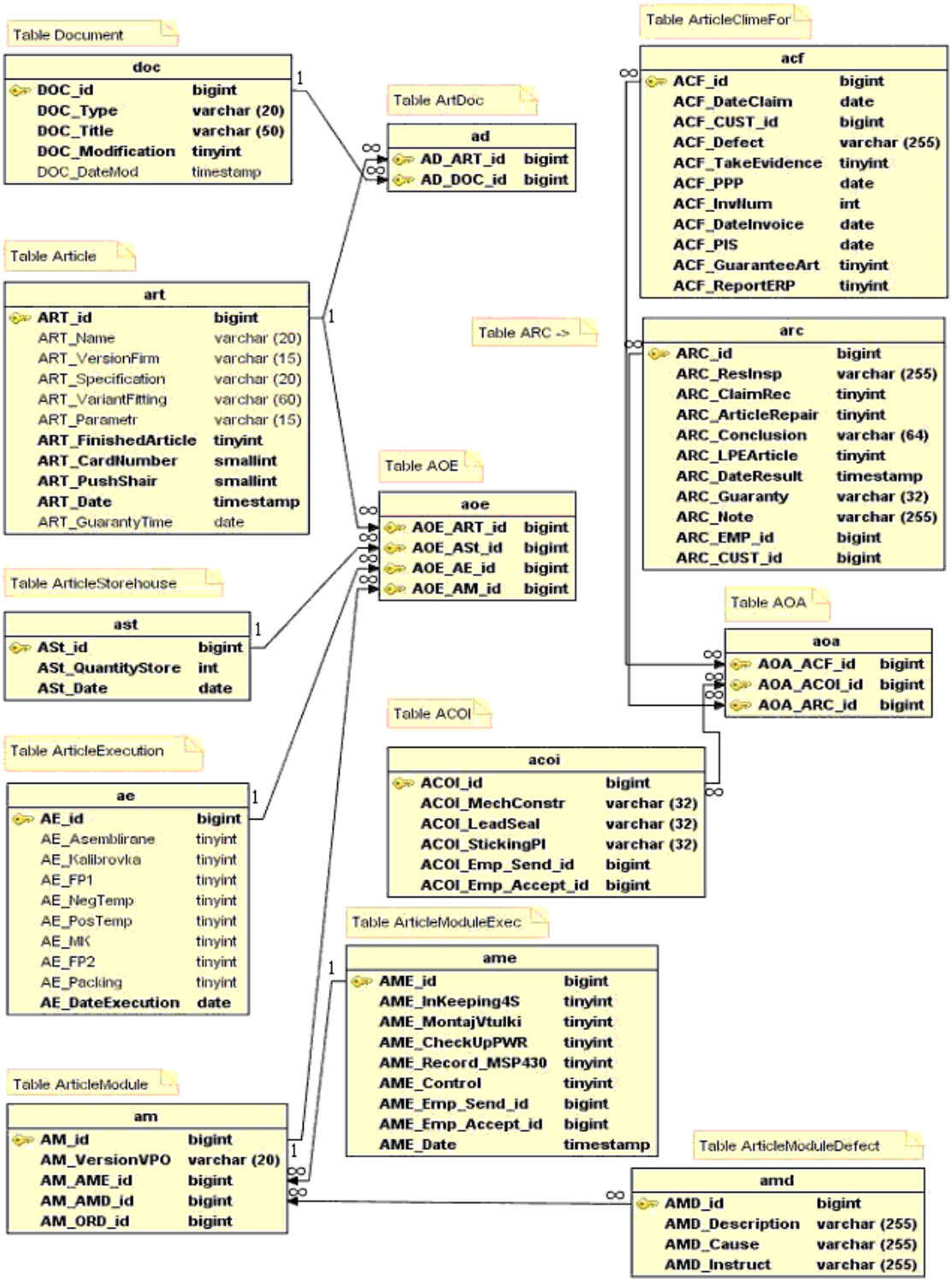
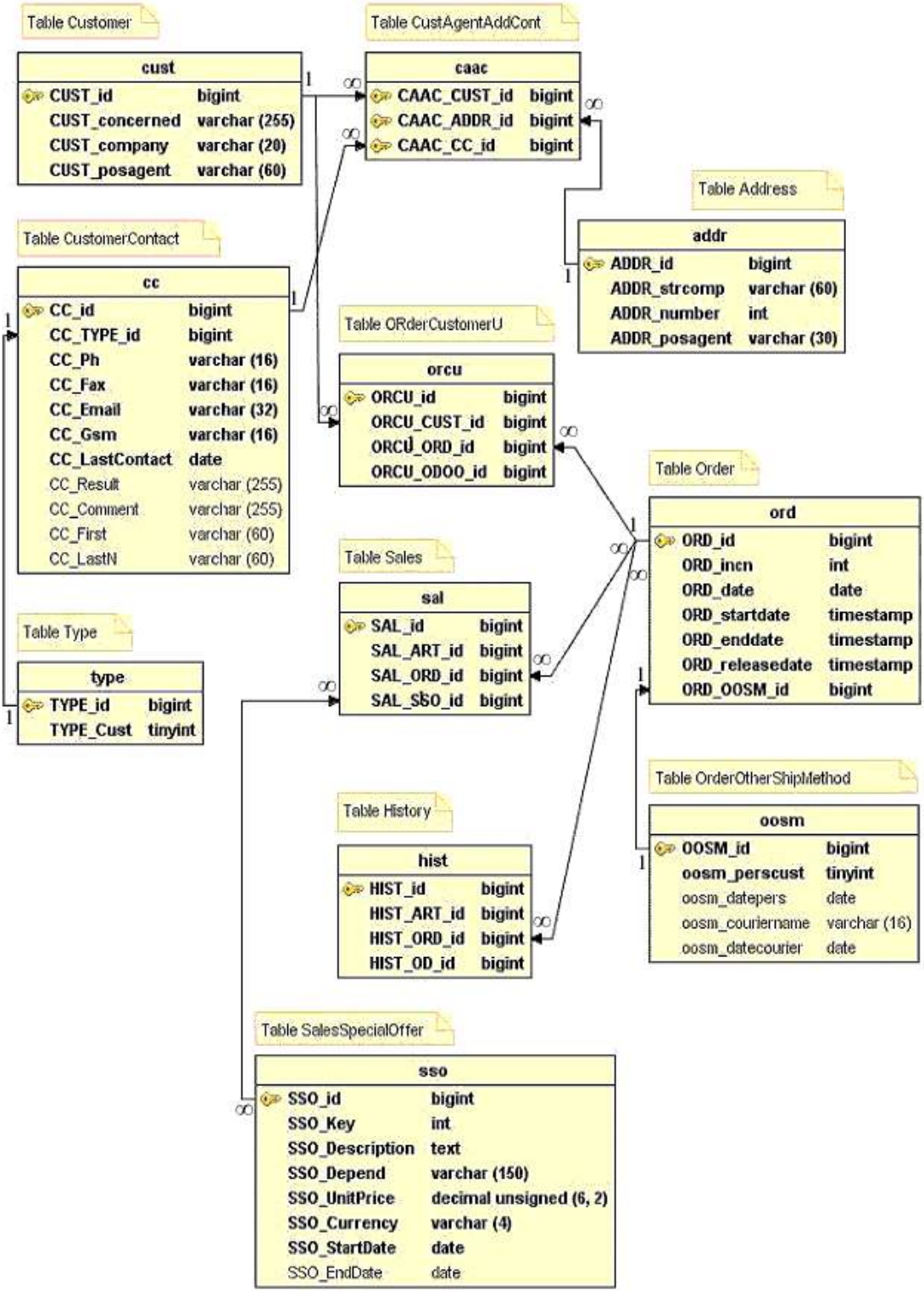


Table Document
doc
DOC_id bigint
DOC_Type varchar (20)
DOC_Title varchar (50)
DOC_Modification tinyint
DOC_DateMod timestamp
Table ArtDoc
ad
AD_ART_id bigint
AD_DOC_id bigint
Table ArticleClimeFor
acf
ACF_id bigint
ACF_DateClaim date
ACF_CUST_id bigint
ACF_Defect varchar (255)
ACF_TakeEvidence tinyint
ACF_PPP date
ACF_InvNum int
ACF_DateInvoice date
ACF_PIS date
ACF_GuaranteeArt tinyint
ACF_ReportERP tinyint
Table Article
art
ART_id bigint
ART_Name varchar (20)
ART_VersionFirm varchar (15)
ART_Specification varchar (20)
ART_VariantFitting varchar (60)
ART_Parametr varchar (15)
ART_FinishedArticle tinyint
ART_CardNumber smallint
ART_PushShair smallint
ART_Date timestamp
ART_GuarantyTime date
Table ARC ->
arc
ARC_id bigint
ARC_ResInsp varchar (255)
ARC_ClaimRec tinyint
ARC_ArticleRepair tinyint
ARC_Conclusion varchar (64)
ARC_LPEArticle tinyint
ARC_DateResult timestamp
ARC_Guaranty varchar (32)
ARC_Note varchar (255)
ARC_EMP_id bigint
ARC_CUST_id bigint
Table AOE
aoe
AOE_ART_id bigint
AOE_ASt_id bigint
AOE_AE_id bigint
AOE_AM_id bigint
Table ArticleStorehouse
ast
ASt_id bigint
ASt_QuantityStore int
ASt_Date date
Table AOA
aoa
AOA_ACF_id bigint
AOA_ACOI_id bigint
AOA_ARC_id bigint
Table ACOI
acoi
ACOI_id bigint
ACOI_MechConstr varchar (32)
ACOI_LeadSeal varchar (32)
ACOI_StickingPl varchar (32)
ACOI_Emp_Send_id bigint
ACOI_Emp_Accept_id bigint
Table ArticleExecution
ae
AE_id bigint
AE_Asemblirane tinyint
AE_Kalibrovka tinyint
AE_FP1 tinyint
AE_NegTemp tinyint
AE_PosTemp tinyint
AE_MK tinyint
AE_FP2 tinyint
AE_Packing tinyint
AE_DateExecution date
Table ArticleModuleExec
ame
AME_id bigint
AME_InKeeping4S tinyint
AME_MontajVtulki tinyint
AME_CheckUpPWR tinyint
AME_Record_MSP430 tinyint
AME_Control tinyint
AME_Emp_Send_id bigint
AME_Emp_Accept_id bigint
AME_Date timestamp
Table ArticleModule
am
AM_id bigint
AM_VersionVPO varchar (20)
AM_AME_id bigint
AM_AMD_id bigint
AM_ORD_id bigint
Table ArticleModuleDefect
amd
AMD_id bigint
AMD_Description varchar (255)
AMD_Cause varchar (255)
AMD_Instruct varchar (255)



Фиг.36 ER - диаграма на базата данни

За построяване на ER- диаграмата се използва SqlEdge средство [2], което позволява да се проектира релационният модел на данните на логическо ниво(логическият модел описва данните чрез средствата на конкретната СУБД)

6.3.5.2 Описание на таблиците от базата данни

В **Приложение 1** подробно са представени всички същности, техните атрибути, ключове и ограничения за цялостност.

Създаване на базата данни чрез SQL скрипт - **Приложение 2 .**

След дизайна на базата данни, ще бъде разгледан дизайна на формите, които показват данните от тази база.

6.4 Дизайн на формите

Разработваният модул включва формите: Authorization, Password, MainForm, Customer, Employee, Order. Тези форми позволяват достъп до данните в съответстващите им таблици.

Форма Authorization

Фиг.37

При стартиране на програмата потребителите въвеждат потребителско име и парола. Програмата проверява потребителското име, паролата на потребителя и неговата роля в базата данни и определя необходимият интерфейс.

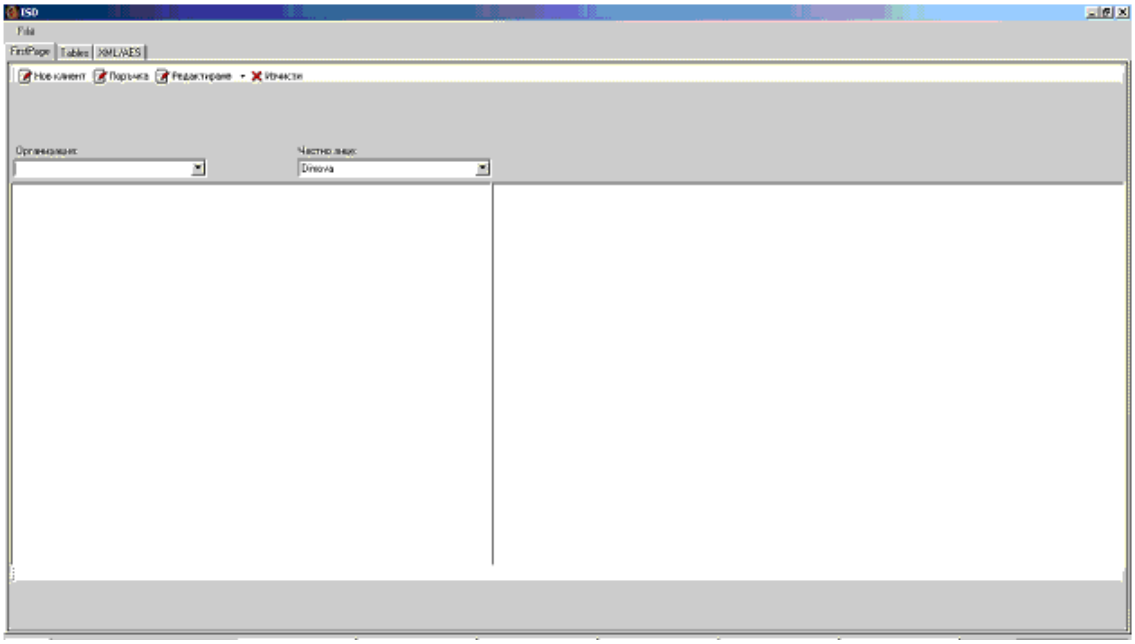
Форма MainForm

Формата frmMainForm - главна работна форма. Използва се контролът TabControl, за да може да се създаде множество страници в прозорец или диалог. Този контрол има характеристика TabPages, която се използва за да се добавят страници към контрола.

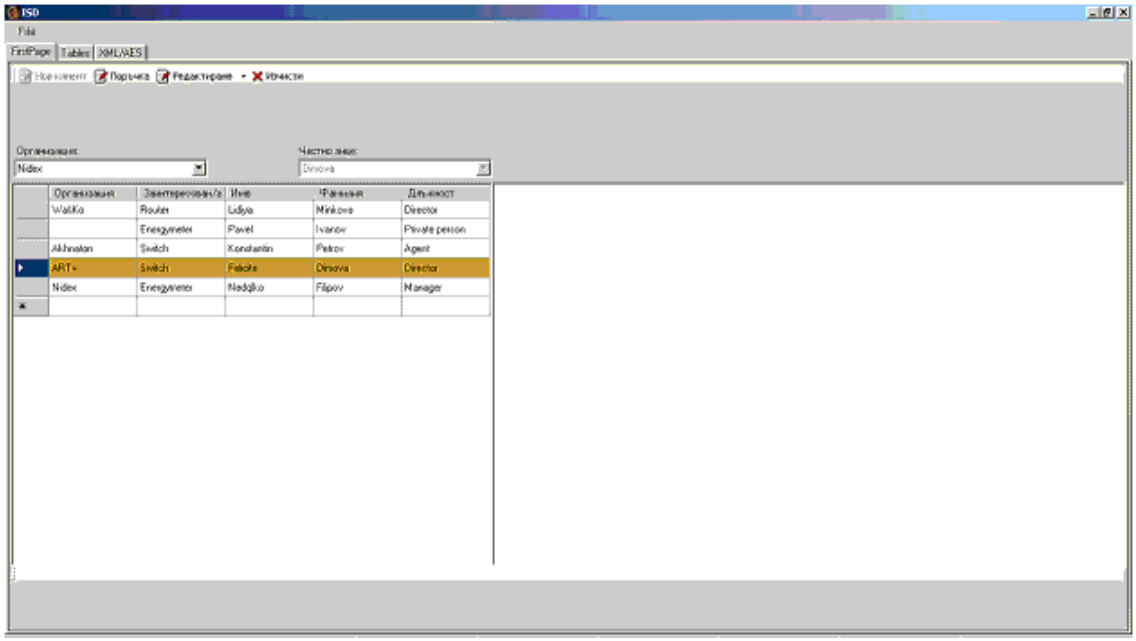
Страница FirstPage – дава информация за организация/частно лице, която/което е закупила/о изделия от компанията.

Страница Tables – извежда информацията от таблиците Преференции, Складова наличност, История и Типове документи от базата данни.

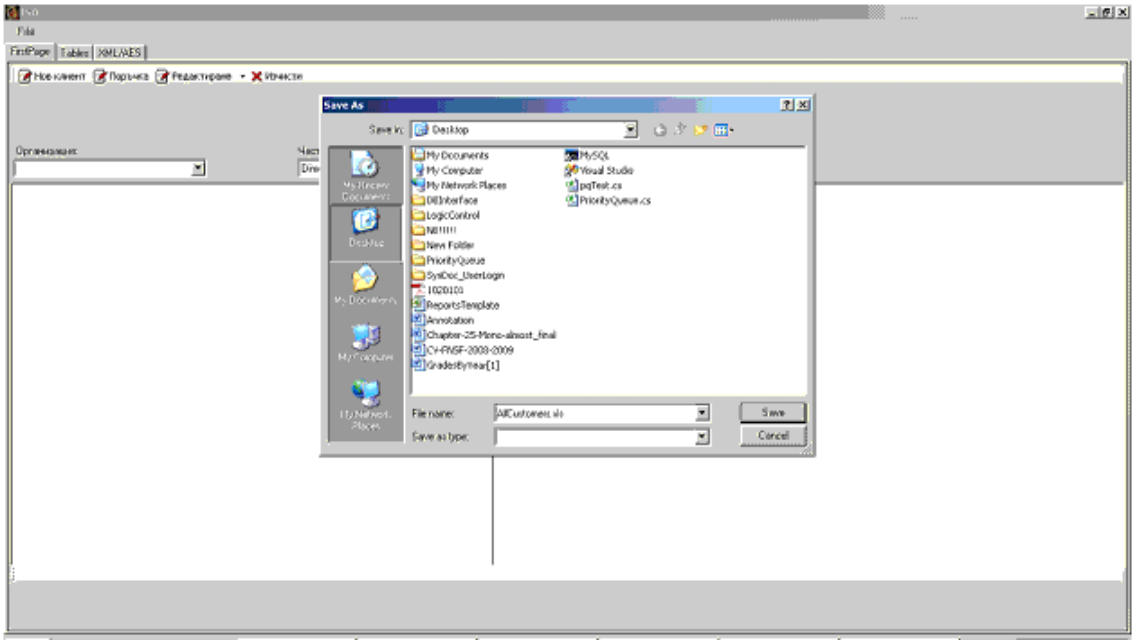
Страница XML/AES – показва опитна постановка за криптиране на информация чрез AES, прочитане на базата данни от xml файл, както и сигурност, базирана на роли.



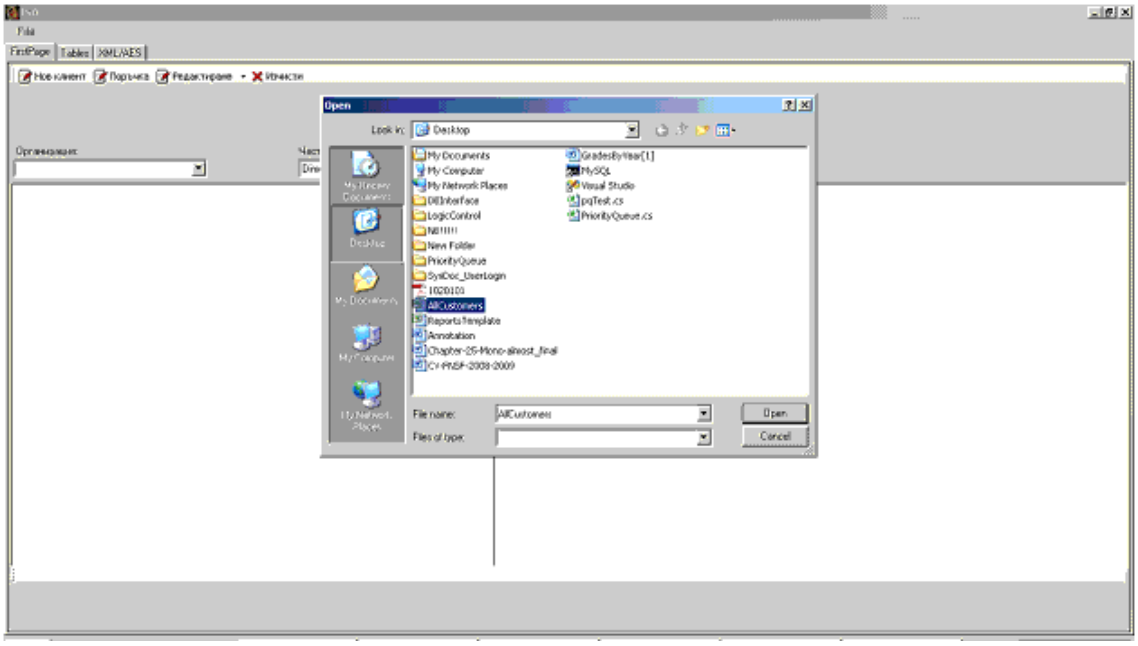
Фиг.38 Страница FirstPage



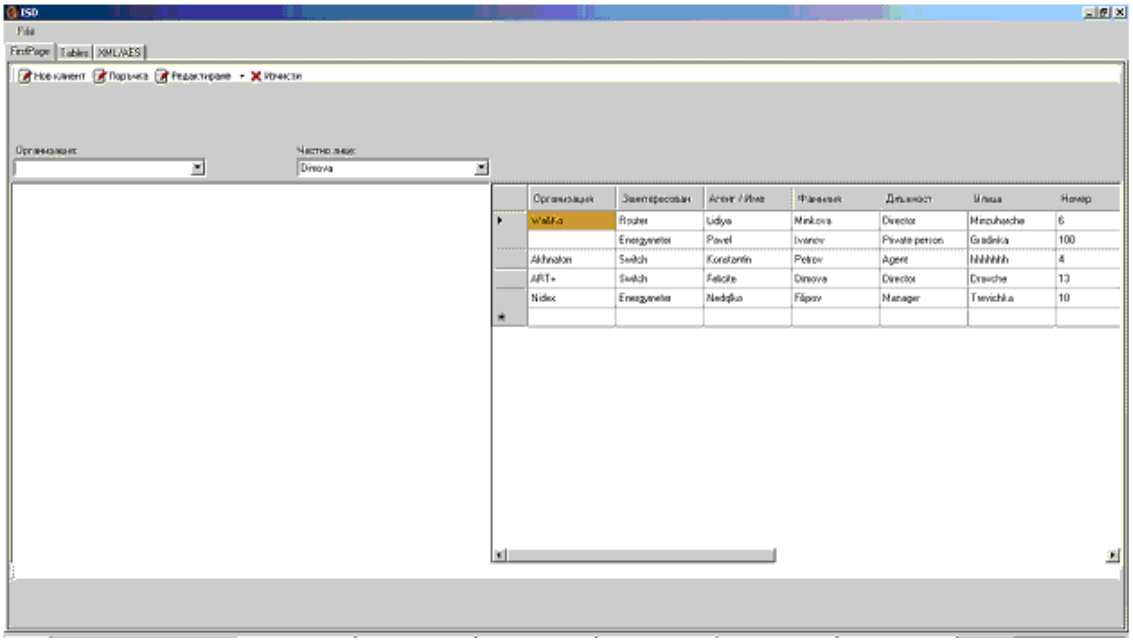
Фиг. 39 Страница FirstPage с изведени записи за клиент



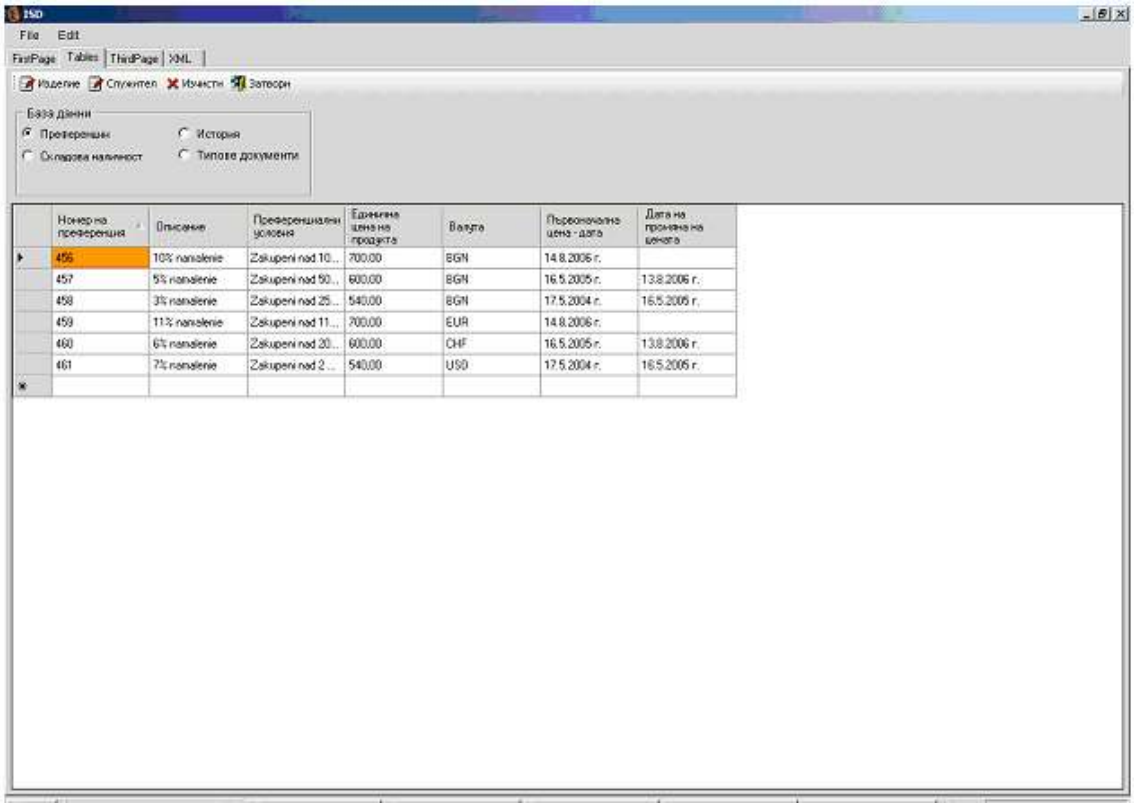
Фиг.40 Запис на всички данни за клиентите в базата данни в файл



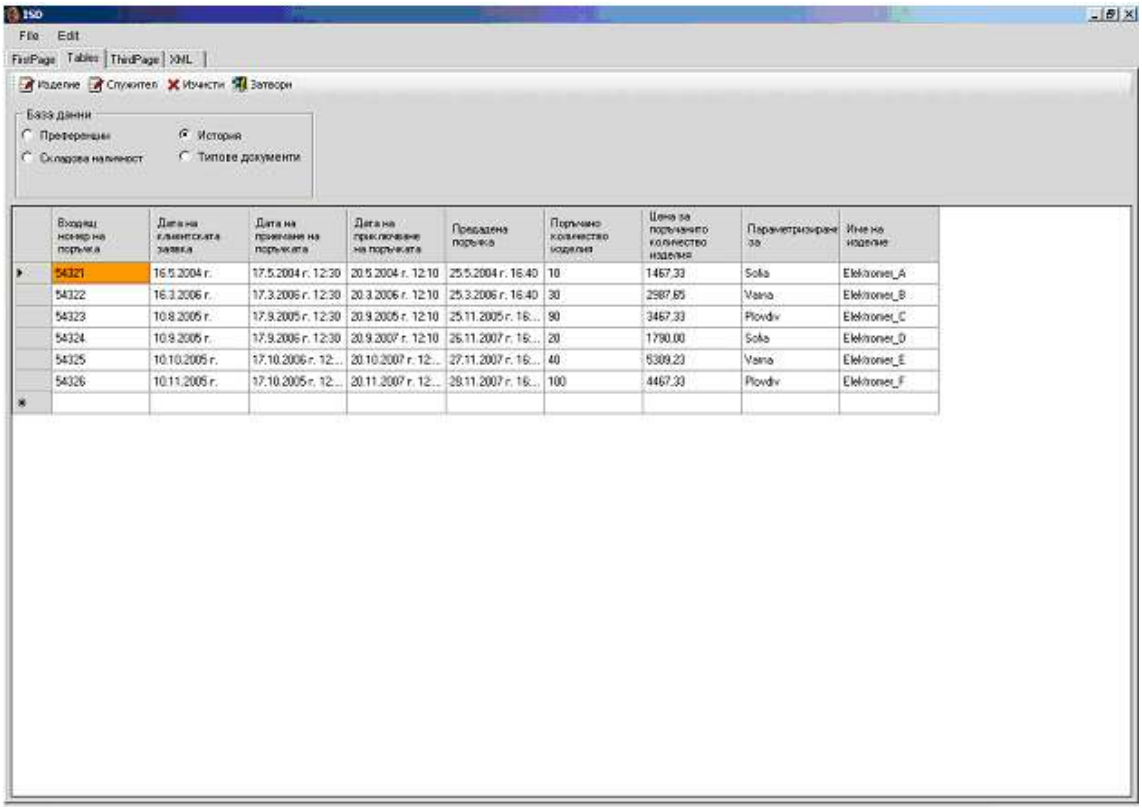
Фиг.41 Отваряне на файл с всички данни за клиентите от базата



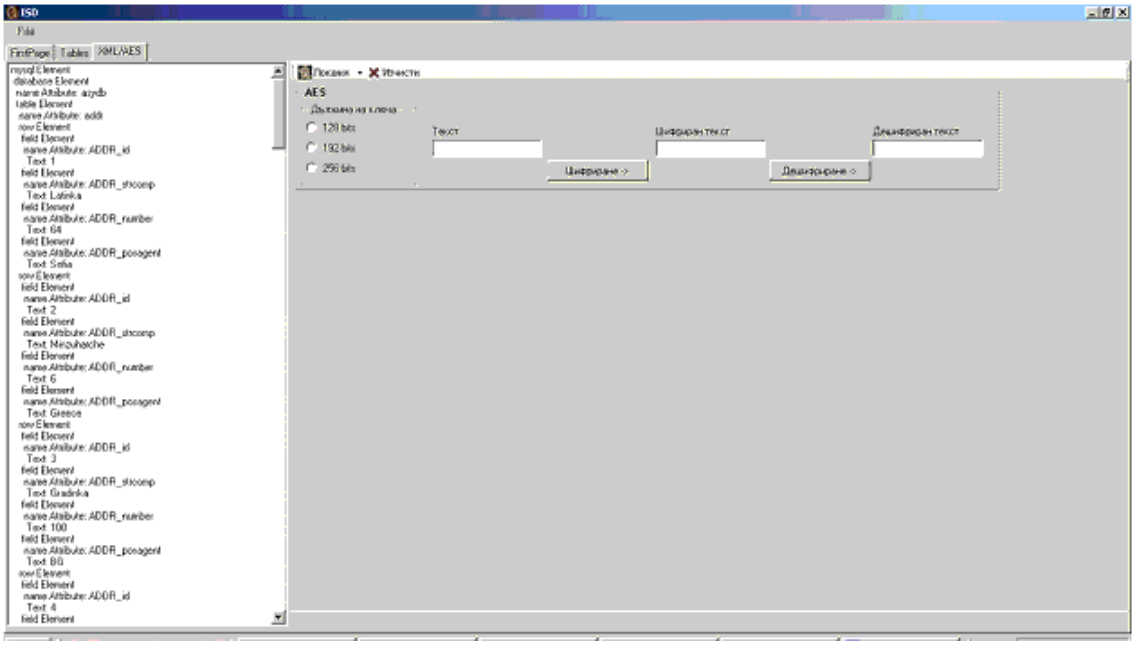
Фиг.42 След отваряне на файла с всички данни за клиентите от базата - визуализиране на информацията



Фиг.43 Страница Tables с информация за преференции

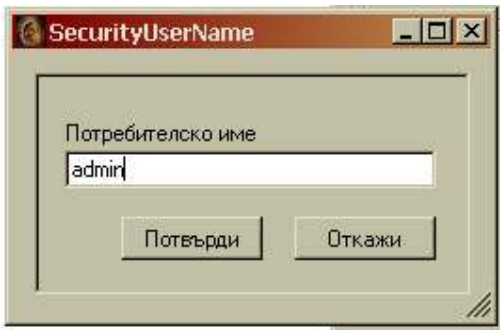


Фиг.44 Страница Tables с информация за историята на дадена поръчка

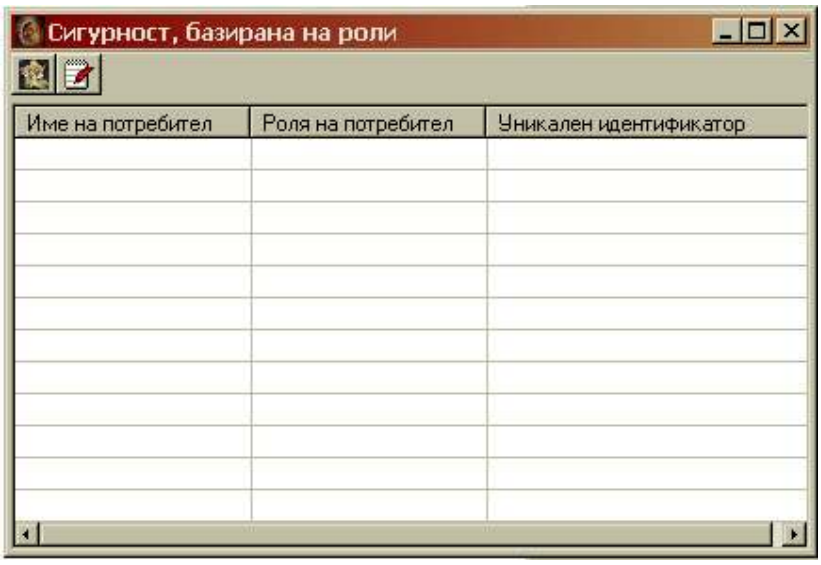


Фиг 45 Страница XML/AES - опитна постановка

Формите, използвани в опитната постановка - Сигурност, базирана на роля



Фиг.46 Форма SecurityUserName

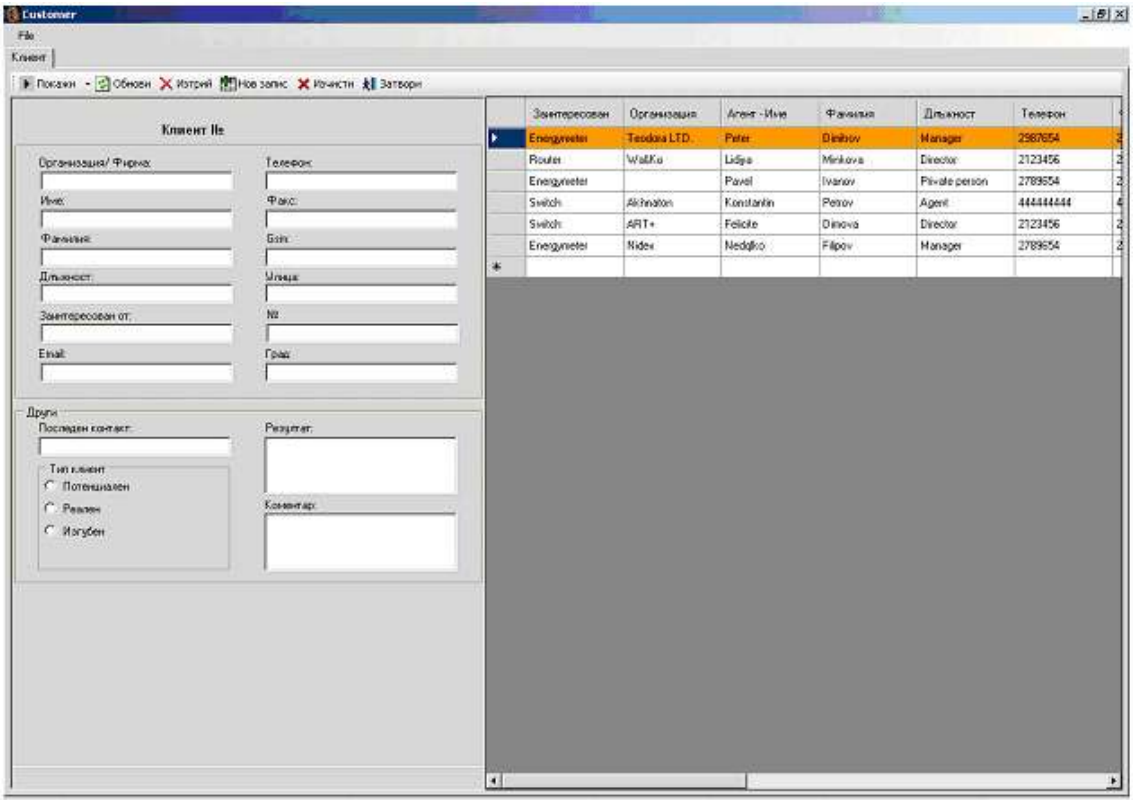


Фиг.47 Форма SecurityMain – главна форма

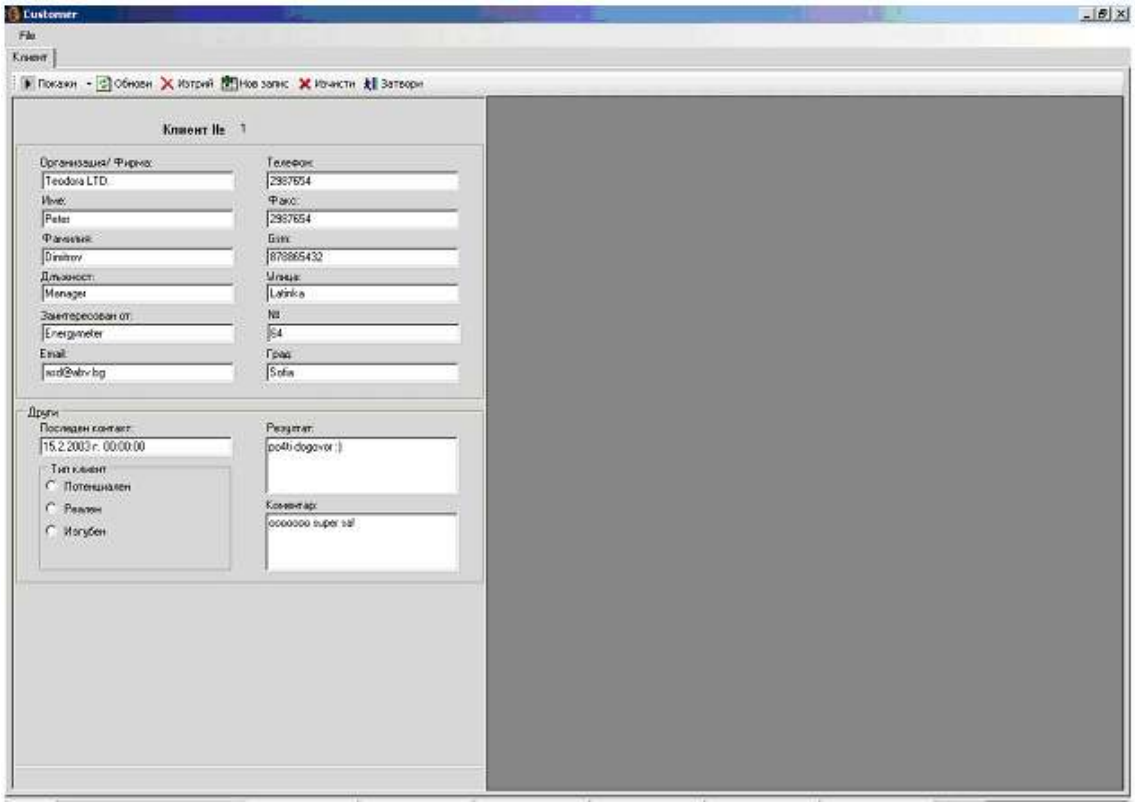
Фиг.48 Форма SecurityChangeRole

Форма frmCustomer

Формата frmCustomer (Клиент) се използва за получаване, добавяне и изменение на информацията за клиенти на фирмата.



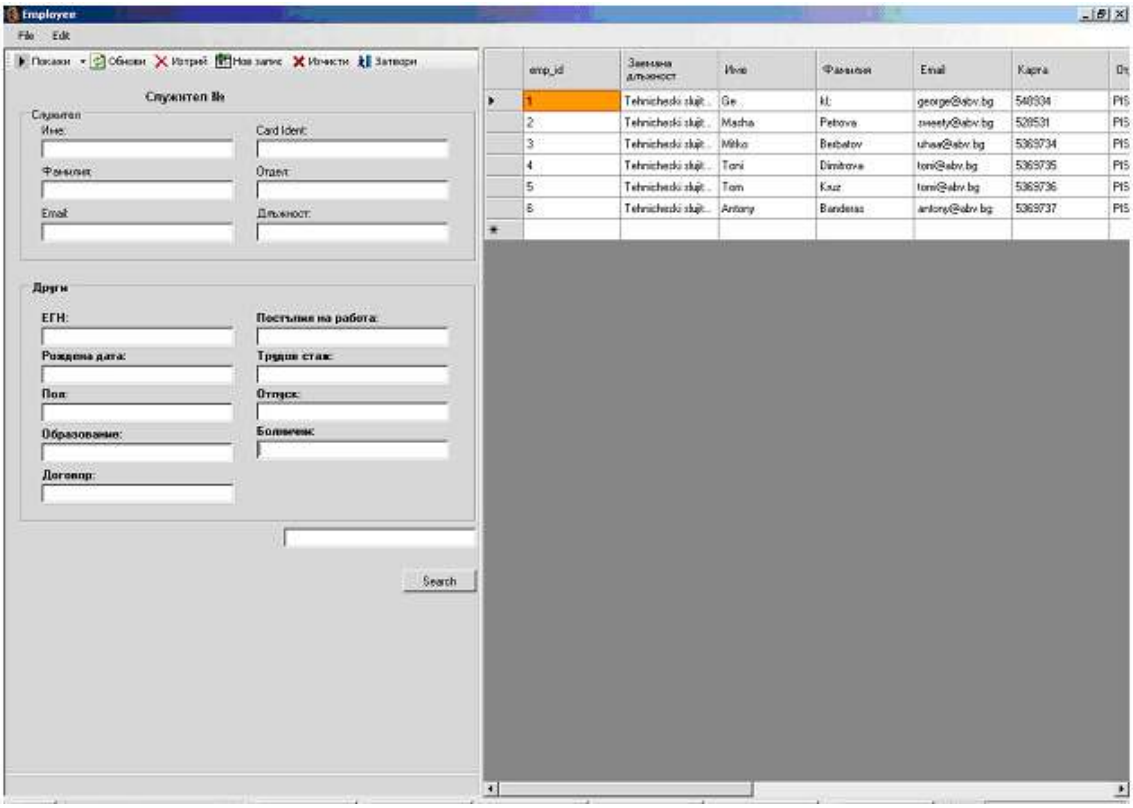
Фиг.49 Форма Customer с показани записи



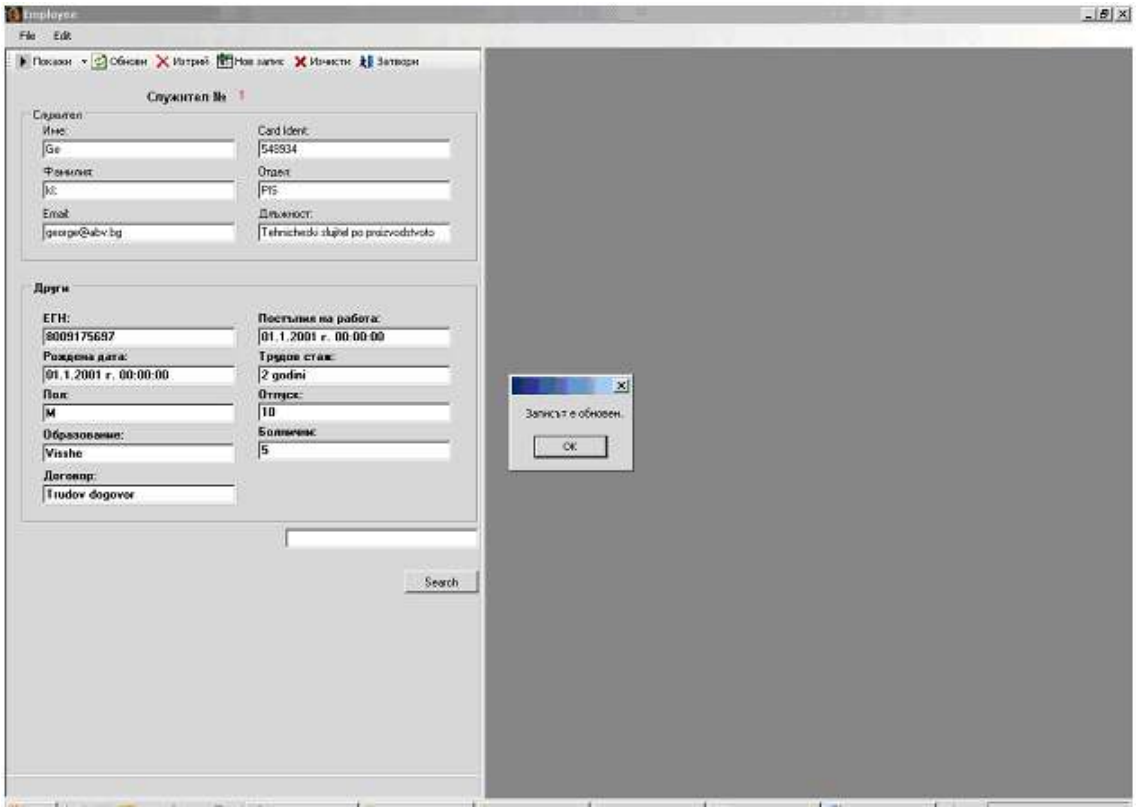
Фиг.50 Форма Customer със записи готови за редактиране

Форма frmEmployee

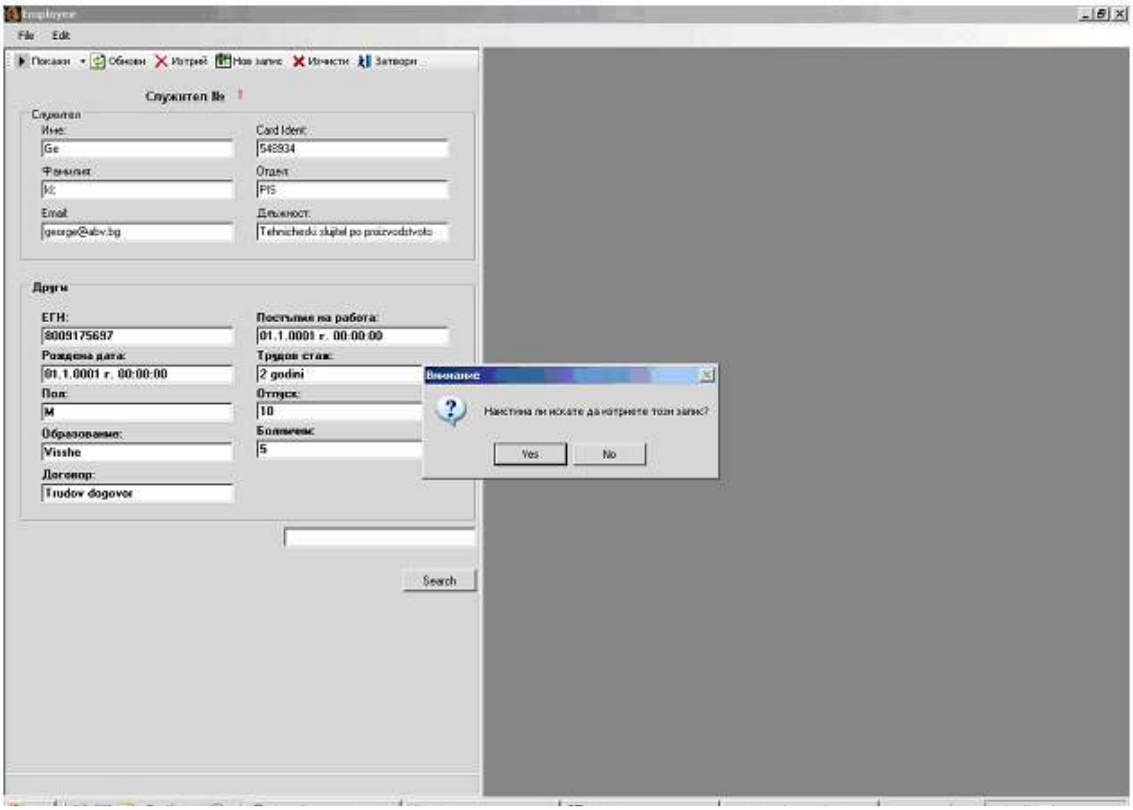
Формата frmEmployee (Служител) се използва за получаване, добавяне и изменение на информацията за служители на фирмата.



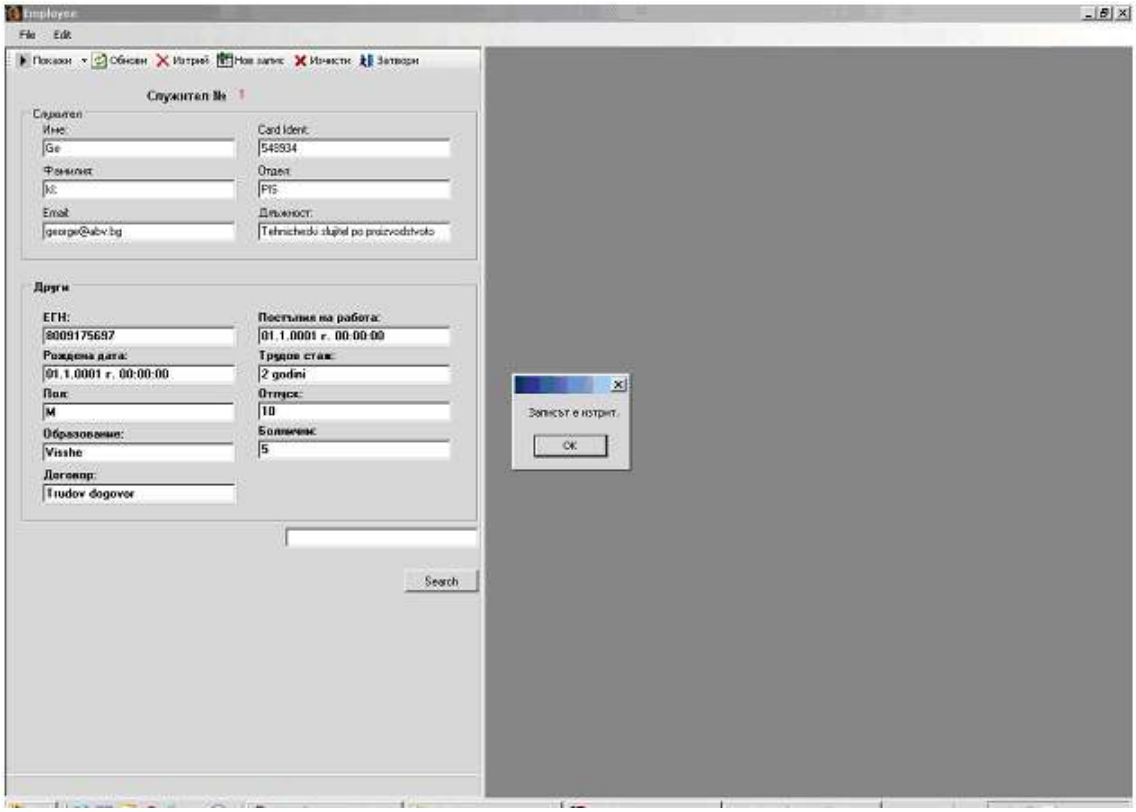
Фиг.51 Форма Employee с показани записи



Фиг.52 Форма Employee с показан диалогов прозорец за обновен запис



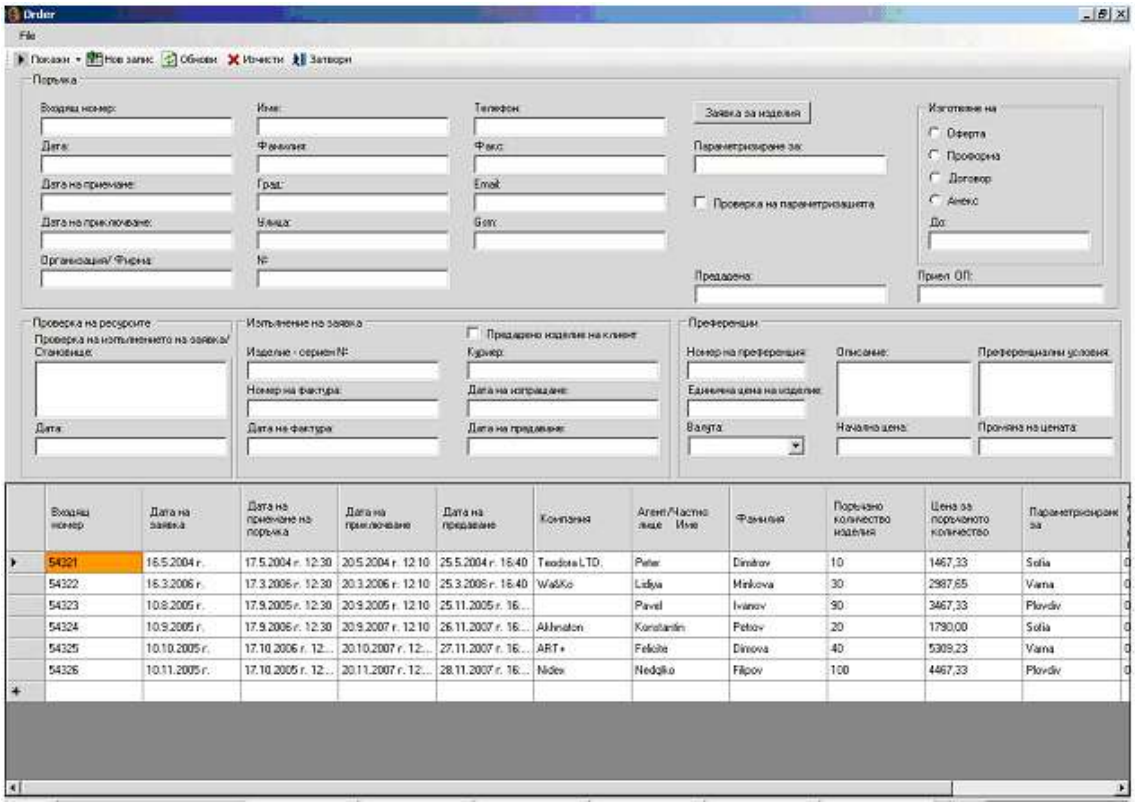
Фиг.53 Диалог, предупреждаващ за изтриване на запис



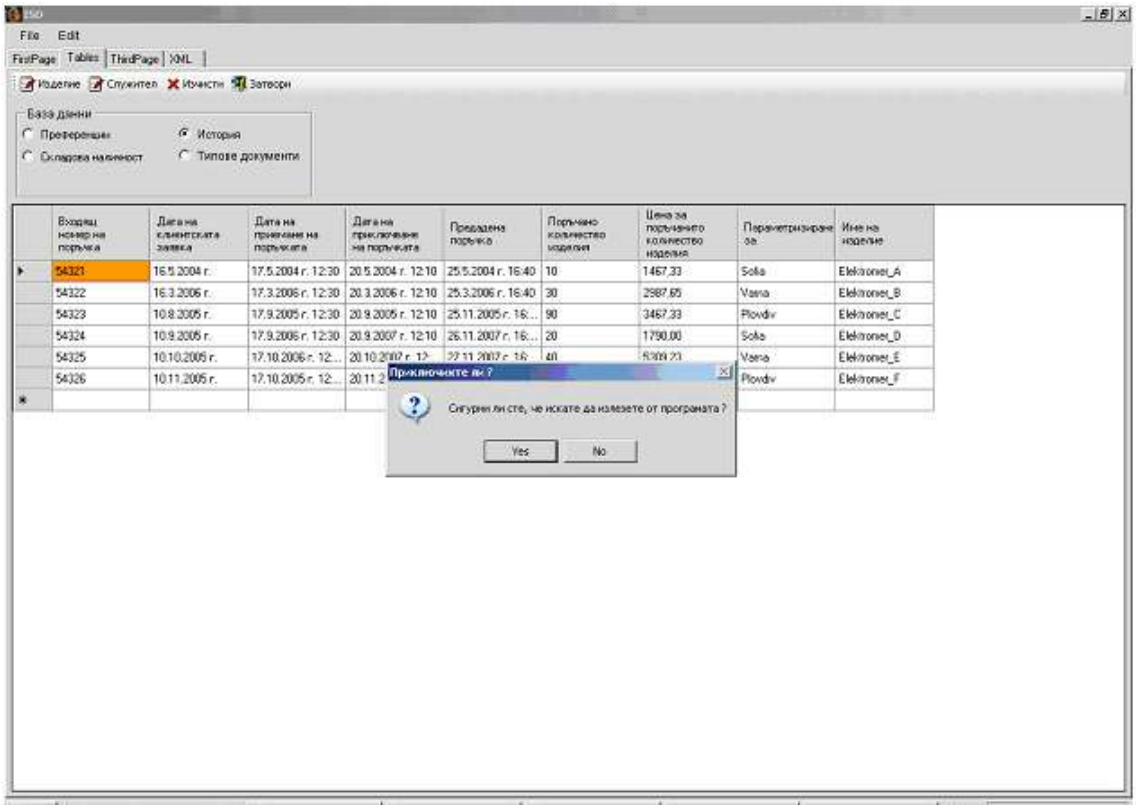
Фиг.54 Форма Employee с показан диалогов прозорец за изтрит запис

Форма **frmOrder**

Форма frmOrder (Поръчки) се използва за получаване, добавяне и редактиране на информацията за определена поръчка.



Фиг.55 Форма Order с показани записи



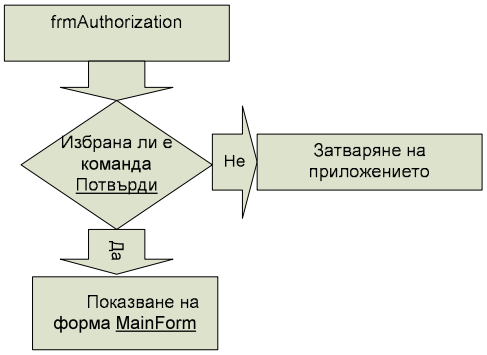
Фиг.56 Стандартен диалог, предупреждаващ за загуба на данни при случаен изход от програмата

6.5 Детайлно проектиране

При детайлното проектиране се извършва подробен дизайн на подмодулите в приложението, на базата на направеното предварително проектиране. Тук се определят действията и взимодействията между отделните интерфейси на модула.

Следващите две фигури представят диаграми на действията и връзките между отделните подмодули.

Фиг.57 показва действията, които са необходими да бъдат извършени при стартиране на приложението. След визуализиране на форма Authorization потребителя въвежда своето потребителско име и парола. Предоставя се възможност за избор – да потвърди или да откаже идентификацията. При потвърждаване се показва основната форма MainForm; при отказ – приложението се затваря.



Фиг. 57 Диаграма на действията

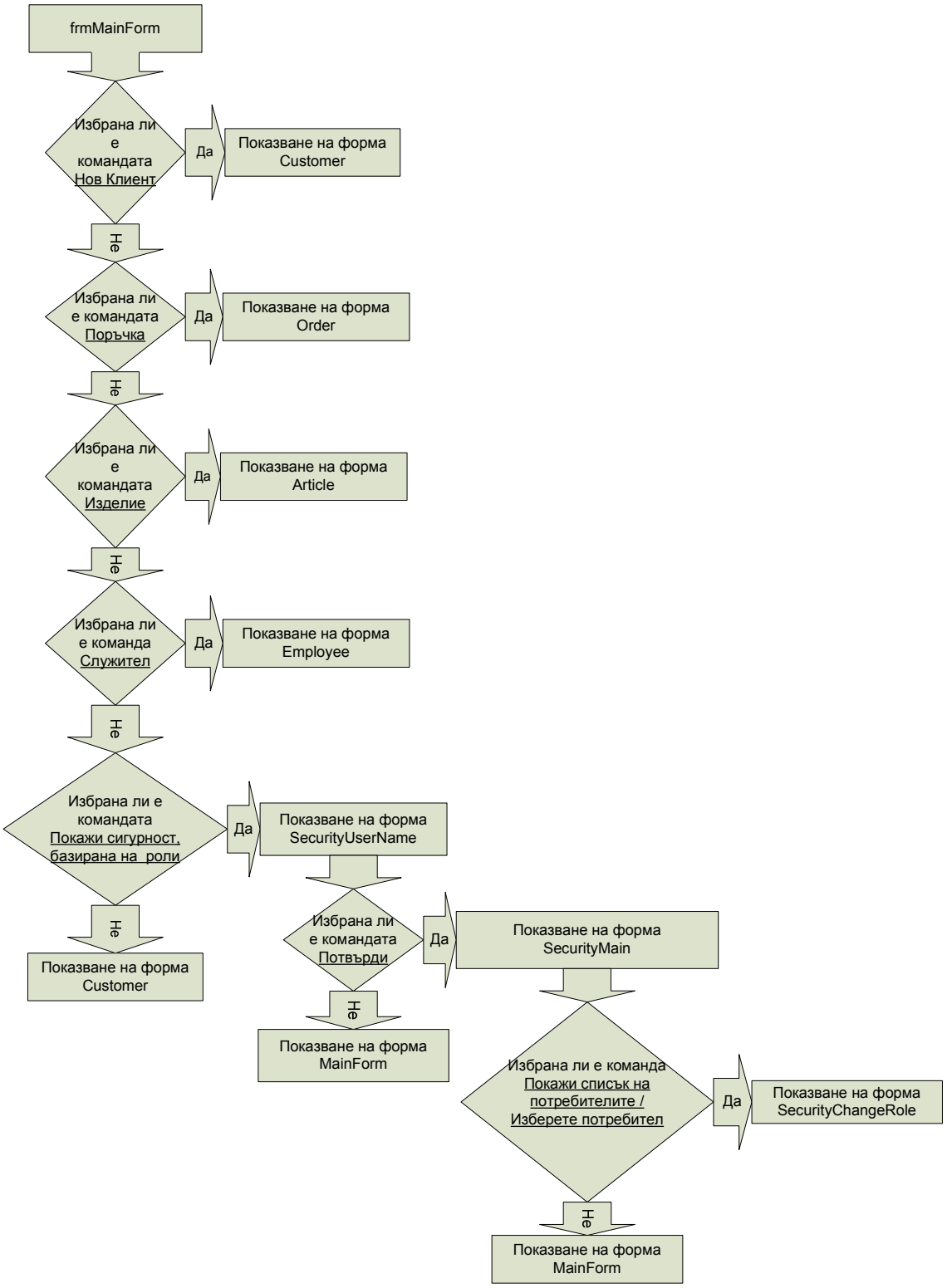
След премината идентификация и оторизация, и в зависимост от ролята на потребителя е определен необходимия му интерефейс.

След визуализация на главната форма се предлага отново възможност за избор, в зависимост от неговите нужди:

1. За работа с подмодул Клиент е необходимо да бъде избрана команда Нов Клиент.
2. За работа с подмодул Поръчка е необходимо да бъде избрана команда Поръчка.
3. За работа с подмодул Изделие е необходимо да бъде избрана команда Изделие.
4. За работа с подмодул Служител е необходимо да бъде избрана команда Служител.

Ако ролята на потребител е Operator, то подмодулът Служител е активен само за извеждане на информация за отделен служител, без права за актуализация на данните.

Ако ролята на потребител е Manager, то той няма права за изтриване на информация.



Фиг.58 Диаграма на действията и връзките на подмодул MainForm с останалите

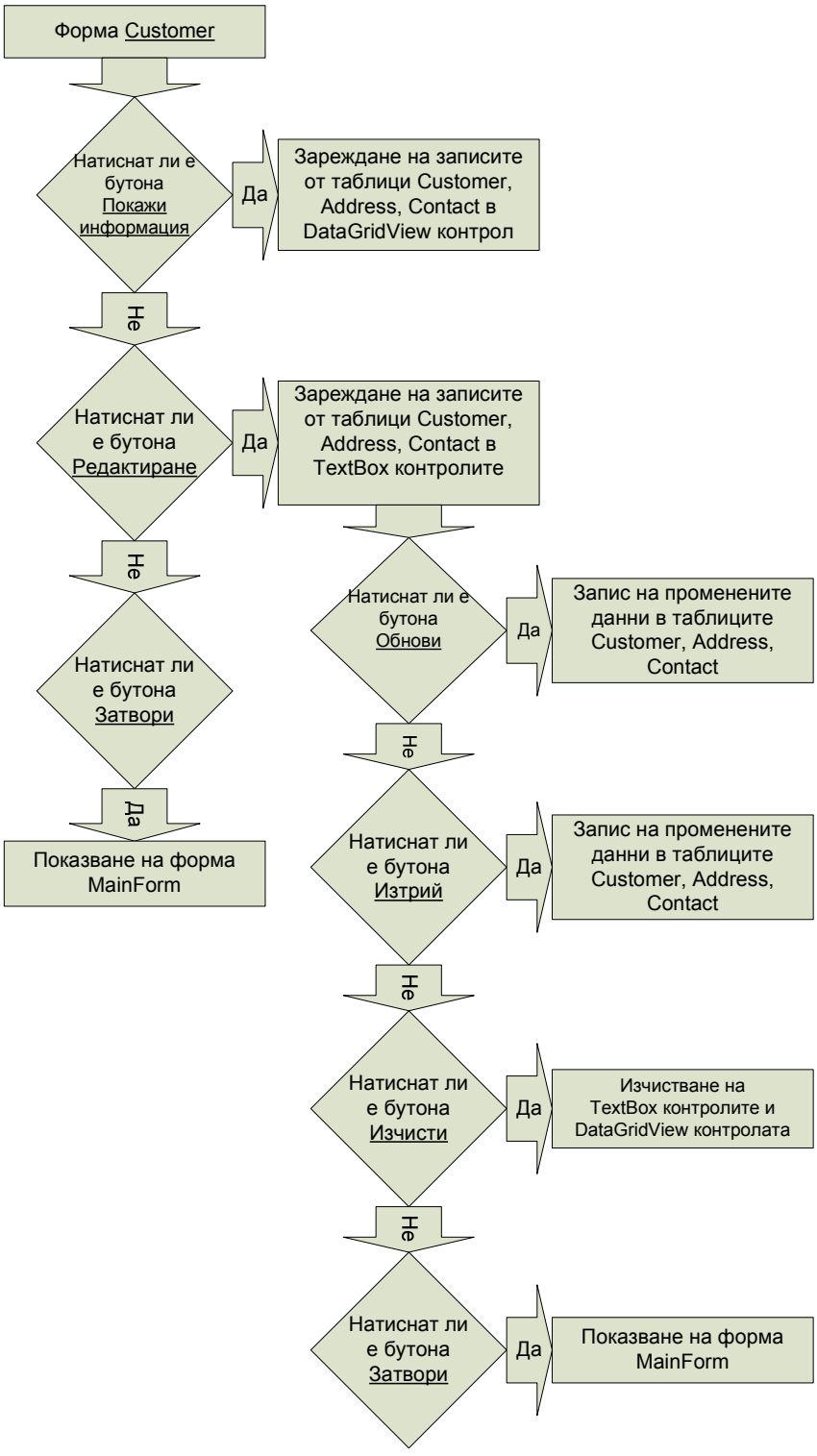
Подмодулът Customer

Подмодулът Customer се състои от формата frmCustomer, съдържаща данните за клиенти на компанията. Диаграмата на действията в подмодула е показана на Фиг.59.

След визуализация на форма Customer и в зависимост от правата на конкретния потребител предоставените възможности за избор са следните:

1. Информацията за клиенти на компанията да бъде изведена само за четене.
2. Да се внасят промени в информацията за клиенти на компанията.
3. Да бъдат изтривани редове от таблица Клиент.
4. Изчистване на TextBox контролите върху формата.
5. Затваряне на подмодул Клиент.

Аналогични са възможностите за избор и за останалите подмодули, на описваният модул.



Фиг.59 Подмодул Клиент

Аналогични описания на деиствията може да се направи за останалите подмодули.

Подмодул Employee

За подмодул Employee, записите ще се зареждат от таблиците: Employee и EmployeeDetails.

Подмодул Order

За подмодул Order, записите ще се зареждат от таблиците: Order, OrderDetails, OrderOther, OrderOtherShipMethod.

Подмодул Article

За подмодул Article, записите ще се зареждат от таблиците: Article, ACF, ACOI, AME, AMD, AE, AM.

6.6 Опитна постановка за използване на собствена технология за безопасност, базирана на роли

Опитната постановка, която се описва изисква автентикация на потребителя, и в зависимост от неговата роля се предоставят различни права за достъп. В основната форма ще се изобразява списък на всички потребители и ще се предоставя възможност ролята на избран потребител да се променя. Списъкът с потребители ще се съдържа във файла : Users.xml Потребителите ще се отнасят към групи – admin, manager, operator:

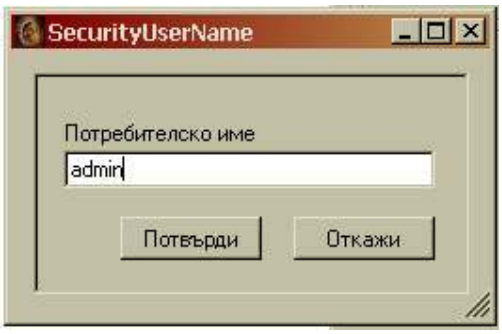
```
<?xml version="1.0" encoding="utf-8" ?>
   <users>
   <user name="admin" id="1" role="admin" />
   <user name="manager" id="2" role="manager" />
                    .
                    .
                    .
  </users>
```

Логиката за определяне на личностите и ролите на потребителите се изнася в отделни класове – CustIdentity.cs и CustPrincipal.cs – **Приложение 2.**

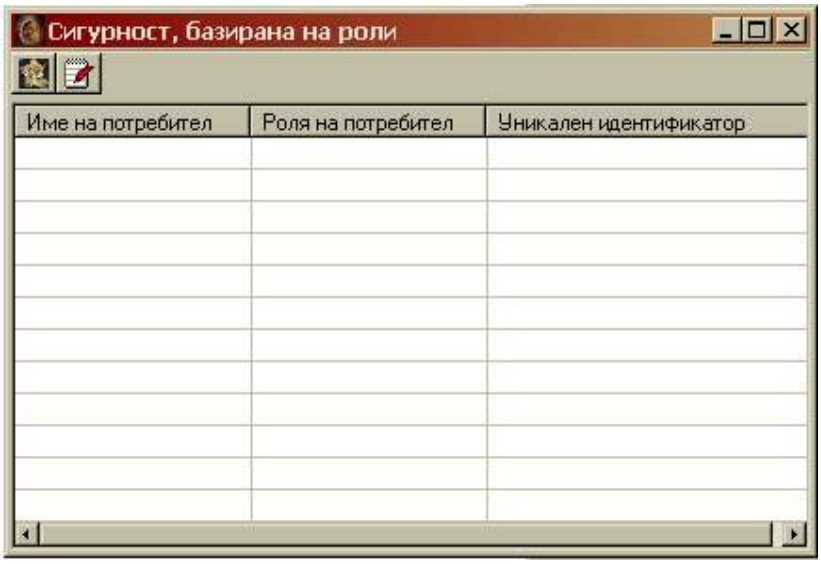
Ще се направи кратко описание на потребителския интерфейс:

Използва се контрола btnRole, при чието задействане се показва формата frmSecurityUserName (Фиг.60) , като достъпът

до главната форма frmSecurityMain (Фиг.61) се блокира, до преминаване на авторизацията.



Фиг.60 Форма SecurityUserName



Фиг.61 Форма SecurityMain – главна форма

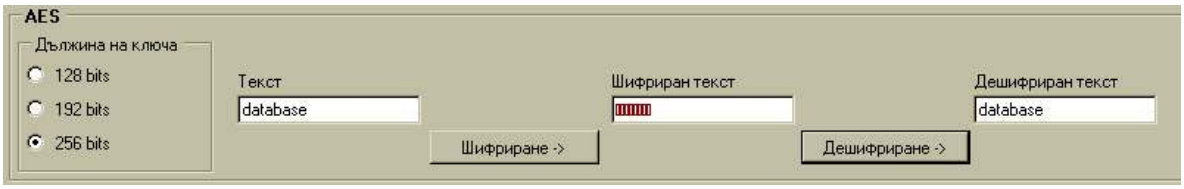
Фиг.62 Форма SecurityChangeRole

Фиг В ни показва формата, която ще бъде използвана – frmSecurityChangeRole за промяна на ролята на потребител.

При отваряне на формата frmSecurityUserName се проверява името на потребителя и неговата роля. В зависимост от неговата роля, интерфейсът се променя по следния начин:

1. Ако ролята на потребителя е admin – има достъп до цялата(пълната) функционалност на опитната постановка – може не само да разглежда пълният списък на потребителите, но и да изменя техните роли. Измененията настъпват след натискане на бутона btnRole отново.
2. Ако ролята на потребителя е manager – може да преглежда списъка, към който принадлежат потребителите.
3. Ако ролята на потребителя е operator – за него не са достъпни останалите форми.

6.7 Опитна постановка за криптиране на информация чрез AES



Фиг.63 Опитна постановка за шифриране на данни, с помощта на AES

С помощта на опитната постановка се показва криптиране на въведена от потребителя информация, чрез описания в Глава 2 алгоритъм за криптиране AES.

Съществува възможност за избор на дължина на ключ.

Алгоритъмът AES се използва в настоящата работа за шифриране на въведената от потребителя парола при Регистрация на потребител. **Приложение 4** представя имплементацията на шифъра с помощта на език за програмиране C# на базата на неговото описание, направено в Глава 2. Ще се даде кратко описание на основните методи в алгоритъма:

```
   //изброим тип променливи на класа AES
   // Bits128, Bits192, Bits256 – дължината на ключа в битове
public enum KeySize { Bits128, Bits192, Bits256 };

   //конструктор на класа AES
public Aes(KeySize keySize, byte[] keyBytes)
    {
      SetNbNkNr(keySize);

      this.key = new byte[this.Nk * 4];  // 16, 24, 32 bytes
      keyBytes.CopyTo(this.key, 0);

      BuildSbox();
      BuildInvSbox();
      BuildRcon();
      KeyExpansion();
    }

  //метод за шифриране на въведения текст
public void Cipher(byte[] input, byte[] output);

   //метод за дешифриране на въведения текст
public void InvCipher(byte[] input, byte[] output) ;

   //метод за отместване на редове
private void ShiftRows();

   //метод за разбъркване на колони
private void MixColumns();

   //метод за отместване на редове при дешифриране
private void InvShiftRows();

   // метод за разбъркване на колони при дешифриране
private void InvShiftRows();
```

6.8 Опитна постановка за прочитане на базата данни от XML файл

Данните могат да бъдат записвани и в XML(*Extensible Markup Language*) файл. XML предствлява набор от правила за съставяне на текстово – базирани формати, които улесняват структурирането на данни. Той има широка употреба в съвременните софтуерни технологии, защото предоставя универсален формат за съхранение и обмен на информация.

В конкретния случай файлът е както следва:

```
<?xml version="1.0" encoding="utf-8" ?>
- <mysql>
- <database name="asydb">
- <table name="cust">
- <row>
<field name="CUST_id">1</field> ....
```

Фиг.64 Извличане на инфромация от XML файл

Той се записва под името DataBase.xml. Негови елементи са *table*, *row*, с атрибут *name* . Целта на тази опитна постановка е да бъде извлечена информацията, която се съдържа в XML документа, както и имената на всички елементи и атрибути от него.

Резултатът от изпълнение на програмата е Фиг.64.

Заключение

Настоящата дипломна работа описа създаването на модул от система за ресурсно планиране, а именно единна база данни, която да бъде защитена от външни и злонамерени атаки. За постигането на тази цел бяха решени следните задачи:

- Проучване на историята на информационните системи за управление на производството, описване замисъл и цели за разработката на системата, целите и задачите на документооборота.
- Създаване на концептуален модел на база данни, за детайлизиране на архитектурата, връзките и ограниченията на данните, както и нива за достъп до различни елементи на данните.
- Създаване на логически модел на базата от данни (БД). Нормализиране на базата.
- Избор на СУБД
- Създаване на физически модел на базата от данни.
- Реализация на базата от данни – чрез инструментални средства се разработва прототип на БД, съгласно вече създадения проект.
- Създаване (проектиране) на интерфейс за взаимодействие с БД.
- Защита на данните:
  - Идентификация на потребители
  - Категории потребители
  - Ограничения за цялостност:
    - Ограничения за значението на колона
    - Ограничение за възможен ключ
    - Ограничение за първичен ключ
    - Ограничение за външен ключ
  - Двойно криптиране на паролите

Използвана литература

1 Нина Синягина,И.Мирчев,И.Д.,Ц.Х., Защита на компютърната информация,издателство "Неофит Рилски", 2005г.

2 Шапиро И. Д., "Управление проектами". СПБ, 1996 «Два-Три»

3 Андреева В. И. "Делопроизводство" - М. "Бизнес-школа "Интел-Синтез"", 1997.

4 Вендров А.М. "Один из подходов к выбору средств проектирования баз данных и приложений. "СУБД"". 1995, №3.

5 С.Наков и колектив, "Програмиране за .NET Framework", БАРС , София 2004

6 Тужаров Хр., {"Въведение в базите данни (Бази данни) "}, (2007)

7 Гавердовский А.А. "Концепция построения систем автоматизации документооборота" // Открытые системы. -1997. -No.1. - С. 29-34.

8 Шуремов Е.Н. "Компьютерный анализ бизнеса" // Мир ПК. -1998. - No.1. - С. 23-31.

9 Bell D., Morrey I., Pogh J. Software Engineering. A programming Approach. Prentice Hall, 1992. 338 с.

10 Буч Г. "Объектно-ориентированное проектирование с примерами применения", М.: Конкорд, 2000. 519 с.

11 Шапиро И. Д., "Управление проектами", СПБ, 1996 «Два-Три»

12 Вендров А.М. "Один из подходов к выбору средств проектирования баз данных и приложений. "СУБД"". 1995, №3.

13 Андреева В. И. "Делопроизводство" - М. «Бизнес-школа «Интел-Синтез»», 1997.

14 Мартин Дж. “Организация баз данных в вычислительных системах”.

15 Нечаев В.И, "Елементи на криптографията.Основи на защита на информацията"

16 Щербаков Л., "Приложна криптография. Използване на криптографски интерфейси", Русская редакция, Москва 2003

Интернет ресурси

1 Дж. Чандлер "Cryptography 101",Введение в криптографию
http://www.compdoc.ru/secur/cript/intrcrypt/

2 Сайт на организацията Bay Breeze Software, която предлага средства за моделиране на БД.
http://www.baybreezesoft.com/

3 Христо Тужаров,Проект "Открито образование",Бази данни, 2007
http://tuj.asenevtsi.com/

4 Пахчанян А.А., Романов Д.И. Рынок ПО:
Обзор систем элек-тронного документооборота:
[http://www.cnews.ru/comments/2002/05/17/content2.shtml],05.07.
2001.

5 IBM,Integrating DB2 Universal Database for Iseries with Microsoft ADO.NET, April 2005
http://www.redbooks.ibm.com/redbooks/pdfs/sg246440.pdf

6 Brian Gladman, "UK Cryptography and Information Security Policy Issues"
http://fp.gladman.plus.com/cryptography_technology/index.htm

7 Консалтинговая группа "ТЕРМИКА",Энциклопедия делопроизводства
http://www.termika.ru/dou/enc/index.html

Приложения

Приложение 1

Описание на таблиците на базата данни

Customer /клиент/			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
CUST_id	PK NOT NULL UNIQUE	BIGINT (20)	Идентификационно число- уникален номер за всеки клиент.
CUST_Concerned	NOT NULL	VARCHAR (255)	Заинтересован от...
CUST_Company	NOT NULL	VARCHAR (20)	Име на клиент (фирма).
CUST_PosAgent	NOT NULL	VARCHAR (60)	Длъжност, заемана от представител (агент) на клиента.

Address /адрес на клиент/			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
ADDR_id	PK NOT NULL UNIQUE	BIGINT (20)	Първичен ключ за релацията **CustomerAddress**.
ADDR_Strcomp	NOT NULL	VARCHAR (60)	Наименование на улицата.
ADDR_Number	NOT NULL	INT (11)	Номер на сграда.
ADDR_Posagent	NOT NULL	VARCHAR (30)	Град.

Contact /информация за клиент/			
Атрибут	Значение **Null**	Тип на	Описание

	Ограничения	данните	
CC_id	PK NOT NULL UNIQUE	BIGINT (20)	Първичен ключ на релацията **CustomerContact.**
CC_ FirstN	NULL	VARCHAR (60)	Собствено име на клиент- частно лице (или агент).
CC_ LastN	NULL	VARCHAR (60)	Фамилия на клиент-частно лице (или агент).
CC_ Ph	NOT NULL	VARCHAR (16)	Телефонен номер на конкретния човек.
CC_ Fax	NOT NULL	VARCHAR (16)	Факс.
CC_ Email	NOT NULL	VARCHAR (32)	E-mail.
CC_GSM	NOT NULL	VARCHAR (16)	GSM номер.
CC_ LastContact	NOT NULL	DATE	Последен контакт с клиента .
CC_ Result	NULL	VARCHAR(255)	Резултат от последния контакт.
CC _Type_id	NOT NULL FK	BIGINT	Идентифицира типа на клиента. Външен ключ,който съответства напървичния ключ на релацията **Type.**
CC_ Comment	NULL	VARCHAR (255)	Коментари.

CAAC **//**			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
CAAC_CUST_id	NOT NULL FK	BIGINT (20)	Външен ключ,който съответства на първичния ключ на релацията **Customer.**
CAAC_ADDR_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на

			първичния ключ на релацията **CustomerAddress.**
CAAC_CC_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **CustomerContact.**

Type **/тип на клиента/**			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
TYPE_id	PK NOT NULL	BIGINT (20)	Първичен ключ на релацията **Type.**
TYPE_Cust	NO NULL	TINYINT (4)	Тип клиент: 1=потенциален 2=реален 3=изгубен.

Sales **/продажби/**			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
SAL _id	NOT NULL PK	BIGINT (20)	Първичен ключ на релацията **Sales.**
SAL_Art_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Article.**
SAL_Ord_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Order.**
SAL_SSO_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **SalesSpecialOffer.**

SpecialOffer /отстъпки/			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
SSO_id	NOT NULL PK UNIQUE	BIGINT (20)	Първичен ключ на релацията **SalesSpecialOffer.**
SSO_Key	NOT NULL	INT (11)	Номер на преференция.
SSO_Description	NULL	VARCHAR (512)	Преференции- описание.
SSO_Depend	NOT NULL	VARCHAR (150)	Преференциални условия.
SSO_UnitPrice	NOT NULL	DECIMAL (6,2) UNSIGNED	Единична цена на изделие.
SSO_Currency	NOT NULL	VARCHAR (4)	Валута.
SSO_StartDate	NOT NULL	DATE	Първоначална цена- дата.
SSO_EndDate	NULL	DATE	Дата на промяна на цената.

Order /поръчани изделия/			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
ORD_id	NOT NULL PK UNIQUE	BIGINT (20)	Първичен ключ на релацията **Order.**
ORD_IncN	NOT NULL	INT (11)	Входящ номер на конкретната поръчка.
ORD_Date	NOT NULL	DATE	Дата на клиентската заявка.
ORD_Startdate	NOT NULL	TIMESTAMP	Дата на приемане на поръчката.
ORD_EndDate	NOT NULL	TIMESTAMP	Дата на приключване на поръчката.

ORD_ReleaseDate	NOT NULL	TIMESTAMP	Поръчката е предадена на...
ORD_OOSM_id	NOT NULL	BIGINT (20)	Външен ключ,който съответства на първичния ключ на релацията **OOSM.**

Ship Method /доставка на изделие/

Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
OOSM_id	NOT NULL PK	BIGINT (20)	Първичен ключ на релацията **OrderOtherShipMethod.**
OOSM_PersCust	NOT NULL	TINYINT (1)	Предадено изделие на клиента- 1=да/0=не (false=0 ;true=1).
OOSM_DatePers	NULL	DATE	Дата на предаване.
OOSM_CourierName	NULL	VARCHAR (16)	Изпратено изделие по куриер на клиента- име на куриер.
OOSM_DateCourier	NULL	DATE	Дата на изпращане.

Order Details /подробности за поръчано изделие/

Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
OD_id	NOT NULL PK	BIGINT (20)	Първичен ключ на релацията **OrderDetails.**
OD_Quantity	NOT NULL	INT (11)	Поръчани количества изделия.
OD_QuantityPrice	NOT NULL	DECIMAL (8,2) UNSIGNED	Цена за поръчаните количества изделия.

OD_Parametris	NOT NULL	VARCHAR (32)	Параметризиране за....
OD_Verification	NOT NULL	TINYINT (1)	Проверка на параметризацията- 1=да/0=не.
OD_MakeOffer	NOT NULL	TINYINT (4)	Изготвяне на : 1=оферта 2=проформа 3=договор 4=анекс
OD_DateMake	NOT NULL	DATE	Дата, до която да бъде изготвено едно от горепосочените.
OD_DateAccept	NOT NULL	TIMESTAMP	Дата на приемане

Order Other /информация за проверка на изпълнение на заявка/			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
OO_id	NOT NULL PK UNIQUE	BIGINT (20)	Първичен ключ на релацията **OrderOther.**
OO_ControlOrder	NOT NULL	VARCHAR (128)	Проверка на изпълнението на заявката – становище.
OO_DateControl	NOT NULL	DATE	Дата на проверка.
OO_InvNum	NOT NULL UNIQUE	INT (11)	Номер на фактура.
OO_DateInvoice	NOT NULL UNIQUE	DATE	Дата на фактура.

OASM //			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание

OASM_OO_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **OrderOther.**
OASM_Art_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Article.**
OASM_OOSM_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **OrderOtherShipMethod.**
OASM_OO_id, OASM_Art_id, OASM_OOSM_id -> PK, UNIQUE			

OESU //			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
OESU_OD_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **OrderDetails.**
OESU_Emp_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Employee.**
OESU_SSO_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **SalesSpecialOffer.**
OESU_OD_id, OESU_Emp_id, OESU_SSO_id -> PK, UNIQUE			

ORCU //			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
ORCU_id	NOT NULL	BIGINT	Първичен ключ на

	PK UNIQUE	(20)	релацията **ORCU.**
ORCU_CUST_id	NOT NULL FK	BIGINT (20)	Външен ключ,който съответства на първичния ключ на релацията **Customer.**
ORCU_ORD_id	NOT NULL FK	BIGINT (20)	Външен ключ,който съответства на първичния ключ на релацията **Order.**
ORCU_ODOO_id	NOT NULL FK	BIGINT (20)	Външен ключ,който съответства на първичния ключ на релацията **ODOO.**

ODOO
//

Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
ODOO_id	NOT NULL PK UNIQUE	BIGINT (20)	Първичен ключ на релацията **OrderOther.**
ODOO_OD_id	NOT NULL FK	BIGINT (20)	Външен ключ,който съответства на първичния ключ на релацията **OD.**
ODOO_OO_id	NOT NULL FK	BIGINT (20)	Външен ключ,който съответства на първичния ключ на релацията **OO.**
ODOO_EMP_id	NOT NULL FK	BIGINT (20)	Външен ключ,който съответства на първичния ключ на релацията **EMP.**

History
/история/

Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
Hist_id	NOT NULL	BIGINT	Първичен ключ на

	PK	(20)	релацията **History.**
Hist_Art_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Article.**
Hist_Ord_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Order.**
Hist_OD_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **OrderDetails.**

Article /изделие/

Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
Art_id	NOT NULL PK UNIQUE	BIGINT (20)	Първичен ключ на релацията **Article.**
Art_Name	NULL	VARCHAR (20)	Име на изделие.
Art_VersionFirm	NULL	VARCHAR (15)	Версия firmware.
Art_Specification	NULL	VARCHAR (20)	Спецификация на изделието.
Art_VariantFittig	NULL	VARCHAR (60)	Документиран вариант на монтаж/модификация по клиентска заявка.
Art_Parametr	NULL	VARCHAR (15)	Параметризиране за...
Art_FinishedArticle	NOT NULL	TINYINT (1)	Готово изделие – годно/негодно (1=да/0=не) за продажба.
Art_CardNumber	NOT NULL	SMALLINT (6)	Карта №.
Art_PushShair	NOT NULL	SMALLINT	Количка №.

		(6)	
Art_Date	NOT NULL	TIMESTAMP	Дата.
Art_GuarantyTime	NULL	DATE	Гаранционен срок.

Document /типове документи/			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
Doc_id	NOT NULL PK UNIQUE	BIGINT (20)	Първичен ключ на релацията **Document.**
Doc_Type	NOT NULL	VARCHAR (20)	Тип на документа (в случая: ISO Процедури).
Doc_Title	NOT NULL	VARCHAR (50)	Вид на документа (пример: Клиентска заявка, информация от клилент и др.).
Doc_Modification	NOT NULL	TINYINT (1)	Изменение (редакция) на документа-да/не(1=да/0=не).
Doc_DateMod	NULL	TIMESTAMP	Дата на изменение (редакция).

Article Document //			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
AD_Art_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Article.**
AD_Doc_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Document.**
AD_Art_id, AD_Doc_id -> PK, UNIQUE			

Article Storehouse /изделия на склад			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
ASt_id	NOT NULL PK UNIQUE	BIGINT (20)	Първичен ключ на релацията **ArticleStorehouse.**
ASt_QuantityStore	NOT NULL	INT (11)	Складова наличност.
ASt_Date	NOT NULL	DATE	Дата.

Article Review /информация от клиент за изделие/			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
AR_id	NOT NULL PK	BIGINT (20)	Първичен ключ на релацията **ArticleReview.**
AR_Opinion	NULL	VARCHAR (255)	Мнения от клиент.
AR_Recommend	NULL	VARCHAR (255)	Препоръки от клиент.
AR_DateReview	NOT NULL	DATE	Дата
AR_Cust_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Customer.**
AR_Rait_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Raiting.**

Raiting

//			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
RAIT _id	NOT NULL PK UNIQUE	BIGINT (20)	Първичен ключ на релацията **Raiting.**
RAIT_Indicator	NULL	VARCHAR (64)	Показател.
RAIT_Value	NULL	TINYINT (4)	Оценка: 1- отлична 2- много добра 3- добра 4- средна 5- слаба

Article Execution /изпълнение/			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
AE_id	NOT NULL PK UNIQUE	BIGINT (20)	Първичен ключ на релацията **ArticleExecution.**
AE_Asemblirane	NOT NULL	TINYINT (1)	Асемблиране- да/не. (1=да/0=не)
AE_Kalibrovka	NOT NULL	TINYINT (1)	Калибровка- да/не. (1=да/0=не)
AE_FP1	NOT NULL	TINYINT (1)	ФП1- да/не. (1=да/0=не)
AE_NegTemp	NOT NULL	TINYINT (1)	-20С- да/не. (1=да/0=не)
AE_PosTemp	NOT NULL	TINYINT (1)	+55С- да/не. (1=да/0=не)
AE_MK	NOT NULL	TINYINT (1)	МК- да/не. (1=да/0=не)
AE_FP2	NOT NULL	TINYINT (1)	ФП2- да/не. (1=да/0=не)
AE_Packing	NOT NULL	TINYINT (1)	Опаковка- да/не (1=да/0=не)
AE_DateExecution	NOT NULL	DATE	Дата на изпълнение.

Article Module /маршрутна карта на модул/			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
AM_id	NOT NULL PK UNIQUE	BIGINT (20)	Първичен ключ на релацията **ArticleModule.**
AM_VersionVPO	NOT NULL	VARCHAR (20)	Версия на ВПО.
AM_AME_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **ArticleModuleExec.**
AM_AMD_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **ArticleModuleDefect.**
AM_Ord_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Order.**

Article Claim Outside Inspection //			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
ACOI_id	NOT NULL PK UNIQUE	BIGINT (20)	Първичен ключ на релацията **ArticleClaimOutsideInspection.**
ACOI_MechConstr	NOT NULL	VARCHAR (32)	Механична конструкция.
ACOI_LeadSeal	NOT NULL	VARCHAR (32)	Пломби.

ACOI_StickingPl	NOT NULL	VARCHAR (32)	Гаранционна лепенка.
ACOI_Emp_Send_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Employee** (предал ОП).
ACOI_Emp_Accept_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Employee** (приел).

Article Module Defect /информация за дефект на изделие/			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
AMD_id	NOT NULL PK	BIGINT (20)	Първичен ключ на релацията **ArticleModuleDefect.**
AMD_Description	NOT NULL	VARCHAR (255)	Описание на дефект (недостатък).
AMD_Cause	NOT NULL	VARCHAR (255)	Причини за дефект (недостатък).
AMD_ Instructions	NOT NULL	VARCHAR (255)	Предписания.

Article Result Claim /резултат от рекламация от клиент/			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
ARC_id	NOT NULL PK	BIGINT (20)	Първичен ключ на релацията

			ArticleResultClaim.
ARC_ResInsp	NOT NULL	VARCHAR (255)	Резултат от преглед в сервиз.
ARC_ClaimRec	NOT NULL	TINYINT (1)	Рекламацията се зачита: да/не(1=да/0=не)
ARC_ArticleRepair	NOT NULL	TINYINT (4)	Изделието е: 1=ремонтирано 2=заменено с нов сериен номер 3=друго
ARC_Conclusion	NOT NULL	VARCHAR (64)	Заключение.
ARC_LPEArticle	NOT NULL	TINYINT (1)	Изделието е проверено в Лаборатория за Проверка на Електромери- годно/негодно(1=да/0=не) за експлоатация.
ARC_DateResult	NOT NULL	TIMESTAMP	Дата.
ARC_Guaranty	NOT NULL	VARCHAR (32)	Гаранционен срок на изделието.
ARC_Note	NULL	VARCHAR (255)	Забележка.
ARC_Emp_id	NOT NULL	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Employee.**
ARC_Cust_id	NOT NULL	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Customer.**

Article Claim for… /информация за рекламации от клиент за дефектни или повредени изделия /

Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
ACF_id	NOT NULL PK	BIGINT (20)	Първичен ключ на релацията **ArticleClaimFor.**
ACF_DateClaim	NOT NULL	DATE	Дата на рекламацията.
ACF_Cust_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Customer.**
ACF_Defect	NOT NULL	VARCHAR (255)	Описание на дефекта.
ACF_TakeEvidence	NOT NULL	TINYINT (1)	Снети показания: да/не
ACF_PPP	NOT NULL	DATE	П.П.П
ACF_InvNum	NOT NULL UNIQUE	INT (11)	Фактура №.
ACF_DateInvoice	NOT NULL UNIQUE	DATE	Дата на фактура.
ACF_PIS	NOT NULL	DATE	ПИС
ACF_GuaranteeArt	NOT NULL	TINYINT (1)	Изделието е в гаранционен срок: да/не.
ACF_ReportERP	NOT NULL	TINYINT (1)	Приложен протокол от "ЕРП"- да/не.

Article Module Execute /изпълнение/

Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
AME_id	NOT NULL PK UNIQUE	BIGINT (20)	Първичен ключ на релацията **ArticleModuleExec.**
AME_InKeeping4S	NOT NULL	TINYINT (1)	Съответствие с ЧС- да/не (1=да/0=не).
AME_MontajVtulki	NOT NULL	TINYINT	Монтаж на втулки и

		(1)	LCD- да/не (1=да/0=не).
AME_CheckUpPWR	NOT NULL	TINYINT (1)	Проверка на PWR- да/не (1=да/0=не).
AME_Record_MSP430	NOT NULL	TINYINT (1)	Запис на MSP430- да/не (1=да/0=не).
AME_Control	NOT NULL	TINYINT (1)	Проверка- да/не (1=да/0=не).
AME_Emp_Send_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Employee** (проверил).
AME_Emp_Accept_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Employee** (изпълнил).
AME_Date	NOT NULL	TIMESTAMP	Дата

AOA //			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
AOA_ACF_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **ArticleClaimForOut InspArtResultClaim.**
AOA_ACOI_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **ArticleClaimOutsideIns pection.**
AOA_ARC_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на

			първичния ключ на релацията **ArticleResultClaim.**
AOA_ACF_id, AOA_ACOI_id, AOA_ARC_id -> PK, UNIQUE			

AOE **//**			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
AOE_ART_id	NOT NULL FK	BIGINT (20)	Външен ключ,който съответства на първичния ключ на релацията **ART.**
AOE_AS_id	NOT NULL FK	BIGINT (20)	Външен ключ,който съответства на първичния ключ на релацията **As.**
AOE_AE_id	NOT NULL FK	BIGINT (20)	Външен ключ,който съответства на първичния ключ на релацията **AE.**
AOE_AM_id	NOT NULL FK	BIGINT (20)	Външен ключ,който съответства на първичния ключ на релацията **AM.**
AOE_ART_id, AOE_AS_id, AOE_AE_id, AOE_AM_id -> PK, UNIQUE			

Employee **/служител/**			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
Emp_id	NOT NULL PK UNIQUE	BIGINT (20)	Първичен ключ на релацията **Employee.**
Emp_Pos	NOT NULL	VARCHAR (60)	Заемана длъжност от служител.
Emp_FirstN	NOT NULL	VARCHAR (60)	Собствено име на служител.
Emp_LastN	NOT NULL	VARCHAR (60)	Фамилия на служител.

Emp_Username	NOT NULL	VARCHAR (20)	Име на потребител.
Emp_Password	NOT NULL	VARCHAR (20)	Парола.
Emp_Email	NOT NULL	VARCHAR (30)	E-mail.
Emp_CardIdent	NOT NULL	VARCHAR (32)	Карта №.
Emp_Section	NOT NULL	VARCHAR (50)	Име на отдел.
Emp_EMBD_id	NOT NULL FK	BIGINT (20)	Външен ключ, който съответства на първичния ключ на релацията **Employee Details.**

Employee Details /детайли за служител/			
Атрибут	Значение **Null** Ограничения	Тип на данните	Описание
EMPD_id	NOT NULL PK UNIQUE	BIGINT (20)	Първичен ключ на релацията **EmployeeDetails.**
EMPD_Egn	NOT NULL	VARCHAR (10)	Егн на служител.
EMPD_BirthD	NOT NULL	DATE	Рождена дата на служител.
EMPD_Sex	NOT NULL	CHAR (1)	Пол: M=Мъж F=Жена
EMPD_LengthOfService	NOT NULL	VARCHAR (16)	Трудов стаж на служител.
EMPD_CountLeave	NULL	SMALLINT (6)	Брой дни- отпуск.
EMPD_CountSickLeave	NULL	SMALLINT (6)	Брой дни- болнични.
EMPD_EntranceOffice	NOT NULL	DATE	Постъпване на работа- дата.

EMPD_Education	NOT NULL	VARCHAR (32)	Образование на служител.
EMPD_TypeContract	NOT NULL	VARCHAR (32)	Договор (тип).

Приложение 2

Създаване на схема на базата данни - SQL скрипт

```
# Host: localhost    Database: asydb
# ------------------------------------------------------
# Server version 5.0.51a-community-nt

CREATE DATABASE `asydb` /*!40100 DEFAULT CHARACTER SET utf8 */;
USE `asydb`;

#
# Table structure for table acf
#

CREATE TABLE `acf` (
  `ACF_id` bigint(20) NOT NULL,
  `ACF_DateClaim` date NOT NULL default '0000-00-00',
  `ACF_CUST_id` bigint(20) NOT NULL default '0',
  `ACF_Defect` varchar(255) NOT NULL default '',
  `ACF_TakeEvidence` tinyint(1) NOT NULL default '0',
  `ACF_PPP` date NOT NULL default '0000-00-00',
  `ACF_InvNum` int(11) NOT NULL default '0',
  `ACF_DateInvoice` date NOT NULL default '0000-00-00',
  `ACF_PIS` date NOT NULL default '0000-00-00',
  `ACF_GuaranteeArt` tinyint(1) NOT NULL default '0',
  `ACF_ReportERP` tinyint(1) NOT NULL default '0',
  PRIMARY KEY  (`ACF_id`),
  UNIQUE KEY `ACF_InvNum` (`ACF_InvNum`),
  UNIQUE KEY `ACF_DateInvoice` (`ACF_DateInvoice`),
  KEY `ACF_CUST_id` (`ACF_CUST_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


#
# Table structure for table acoi
#

CREATE TABLE `acoi` (
  `ACOI_id` bigint(20) NOT NULL,
  `ACOI_MechConstr` varchar(32) NOT NULL default '',
  `ACOI_LeadSeal` varchar(32) NOT NULL default '',
  `ACOI_StickingPl` varchar(32) NOT NULL default '',
  `ACOI_Emp_Send_id` bigint(20) NOT NULL,
  `ACOI_Emp_Accept_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`ACOI_id`),
  KEY `ACOI_Emp_Send_id` (`ACOI_Emp_Send_id`),
  KEY `ACOI_Emp_Accept_id` (`ACOI_Emp_Accept_id`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table ad
#

CREATE TABLE `ad` (
  `AD_ART_id` bigint(20) NOT NULL,
  `AD_DOC_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`AD_ART_id`,`AD_DOC_id`),
  KEY `AD_DOC_id` (`AD_DOC_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table addr
#

CREATE TABLE `addr` (
  `ADDR_id` bigint(20) NOT NULL,
  `ADDR_strcomp` varchar(60) NOT NULL default '',
  `ADDR_number` int(11) NOT NULL default '0',
  `ADDR_posagent` varchar(30) NOT NULL default '',
  PRIMARY KEY  (`ADDR_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table ae
#

CREATE TABLE `ae` (
  `AE_id` bigint(20) NOT NULL,
  `AE_Asemblirane` tinyint(1) default NULL,
  `AE_Kalibrovka` tinyint(1) default NULL,
  `AE_FP1` tinyint(1) default NULL,
  `AE_NegTemp` tinyint(1) default NULL,
  `AE_PosTemp` tinyint(1) default NULL,
  `AE_MK` tinyint(1) default NULL,
  `AE_FP2` tinyint(1) default NULL,
  `AE_Packing` tinyint(1) default NULL,
  `AE_DateExecution` date NOT NULL default '0000-00-00',
  PRIMARY KEY  (`AE_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table am
#

CREATE TABLE `am` (
  `AM_id` bigint(20) NOT NULL,
  `AM_VersionVPO` varchar(20) NOT NULL default '',
  `AM_AME_id` bigint(20) NOT NULL,
  `AM_AMD_id` bigint(20) NOT NULL,
  `AM_ORD_id` bigint(20) NOT NULL ,
  PRIMARY KEY  (`AM_id`),
  KEY `AM_AME_id` (`AM_AME_id`),
  KEY `AM_AMD_id` (`AM_AMD_id`),
  KEY `AM_ORD_id` (`AM_ORD_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table amd
#

CREATE TABLE `amd` (
  `AMD_id` bigint(20) NOT NULL,
  `AMD_Description` varchar(255) NOT NULL default '',
  `AMD_Cause` varchar(255) NOT NULL default '',
```

```
  `AMD_Instruct` varchar(255) NOT NULL default '',
  PRIMARY KEY  (`AMD_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table ame
#

CREATE TABLE `ame` (
  `AME_id` bigint(20) NOT NULL,
  `AME_InKeeping4S` tinyint(1) NOT NULL,
  `AME_MontajVtulki` tinyint(1) NOT NULL,
  `AME_CheckUpPWR` tinyint(1) NOT NULL,
  `AME_Record_MSP430` tinyint(1) NOT NULL,
  `AME_Control` tinyint(1) NOT NULL,
  `AME_Emp_Send_id` bigint(20) NOT NULL,
  `AME_Emp_Accept_id` bigint(20) NOT NULL,
  `AME_Date` timestamp NOT NULL default CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP,
  PRIMARY KEY  (`AME_id`),
  KEY `AME_Emp_Send_id` (`AME_Emp_Send_id`),
  KEY `AME_Emp_Accept_id` (`AME_Emp_Accept_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table aoa
#

CREATE TABLE `aoa` (
  `AOA_ACF_id` bigint(20) NOT NULL,
  `AOA_ACOI_id` bigint(20) NOT NULL,
  `AOA_ARC_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`AOA_ACF_id`,`AOA_ACOI_id`,`AOA_ARC_id`),
  KEY `AOA_ACOI_id` (`AOA_ACOI_id`),
  KEY `AOA_ARC_id` (`AOA_ARC_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table aoe
#

CREATE TABLE `aoe` (
  `AOE_ART_id` bigint(20) NOT NULL,
  `AOE_ASt_id` bigint(20) NOT NULL,
  `AOE_AE_id` bigint(20) NOT NULL,
  `AOE_AM_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`AOE_ART_id`,`AOE_ASt_id`,`AOE_AE_id`,`AOE_AM_id`),
  KEY `AOE_ASt_id` (`AOE_ASt_id`),
  KEY `AOE_AE_id` (`AOE_AE_id`),
  KEY `AOE_AM_id` (`AOE_AM_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table ar
#

CREATE TABLE `ar` (
  `AR_id` bigint(20) NOT NULL,
  `AR_Opinion` text,
  `AR_Recommend` text,
  `AR_DateReview` date NOT NULL default '0000-00-00',
  `AR_Cust_id` bigint(20) NOT NULL,
  `AR_Rait_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`AR_id`),
  KEY `FK_ar_cust_id` (`AR_Cust_id`),
  KEY `FK_ar_rait_id` (`AR_Rait_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
#
# Table structure for table arc
#

CREATE TABLE `arc` (
  `ARC_id` bigint(20) NOT NULL,
  `ARC_ResInsp` varchar(255) NOT NULL,
  `ARC_ClaimRec` tinyint(1) NOT NULL,
  `ARC_ArticleRepair` tinyint(4) NOT NULL,
  `ARC_Conclusion` varchar(64) NOT NULL,
  `ARC_LPEArticle` tinyint(1) NOT NULL,
  `ARC_DateResult` timestamp NOT NULL default CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP,
  `ARC_Guaranty` varchar(32) NOT NULL,
  `ARC_Note` varchar(255) NOT NULL,
  `ARC_EMP_id` bigint(20) NOT NULL,
  `ARC_CUST_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`ARC_id`),
  KEY `ARC_EMP_id` (`ARC_EMP_id`),
  KEY `ARC_CUST_id` (`ARC_CUST_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table art
#

CREATE TABLE `art` (
  `ART_id` bigint(20) NOT NULL,
  `ART_Name` varchar(20) default NULL,
  `ART_VersionFirm` varchar(15) default NULL,
  `ART_Specification` varchar(20) default NULL,
  `ART_VariantFitting` varchar(60) default NULL,
  `ART_Parametr` varchar(15) default NULL,
  `ART_FinishedArticle` tinyint(1) NOT NULL,
  `ART_CardNumber` smallint(6) NOT NULL,
  `ART_PushShair` smallint(6) NOT NULL,
  `ART_Date` timestamp NOT NULL default CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP,
  `ART_GuarantyTime` date default NULL,
  PRIMARY KEY  (`ART_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table ast
#

CREATE TABLE `ast` (
  `ASt_id` bigint(20) NOT NULL,
  `ASt_QuantityStore` int(11) NOT NULL,
  `ASt_Date` date NOT NULL default '0000-00-00',
  PRIMARY KEY  (`ASt_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table caac
#

CREATE TABLE `caac` (
  `CAAC_CUST_id` bigint(20) NOT NULL,
  `CAAC_ADDR_id` bigint(20) NOT NULL,
  `CAAC_CC_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`CAAC_CUST_id`,`CAAC_ADDR_id`,`CAAC_CC_id`),
  KEY `CAAC_ADDR_id` (`CAAC_ADDR_id`),
  KEY `CAAC_CC_id` (`CAAC_CC_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table cc
#
```

```
CREATE TABLE `cc` (
  `CC_id` bigint(20) NOT NULL de,
  `CC_TYPE_id` bigint(20) NOT NULL,
  `CC_Ph` varchar(16) NOT NULL ,
  `CC_Fax` varchar(16) NOT NULL,
  `CC_Email` varchar(32) NOT NULL,
  `CC_Gsm` varchar(16) NOT NULL,
  `CC_LastContact` date NOT NULL default '0000-00-00',
  `CC_Result` varchar(255) default NULL,
  `CC_Comment` varchar(255) default NULL,
  `CC_First` varchar(60) default NULL,
  `CC_LastN` varchar(60) default NULL,
  PRIMARY KEY  (`CC_id`),
  KEY `CC_TYPE_id` (`CC_TYPE_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table cust
#

CREATE TABLE `cust` (
  `CUST_id` bigint(20) NOT NULL,
  `CUST_concerned` varchar(255) NOT NULL default '',
  `CUST_company` varchar(20) NOT NULL default '',
  `CUST_posagent` varchar(60) NOT NULL default '',
  PRIMARY KEY  (`CUST_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table doc
#

CREATE TABLE `doc` (
  `DOC_id` bigint(20) NOT NULL,
  `DOC_Type` varchar(20) NOT NULL,
  `DOC_Title` varchar(50) NOT NULL,
  `DOC_Modification` tinyint(1) NOT NULL,
  `DOC_DateMod` timestamp NULL default NULL,
  PRIMARY KEY  (`DOC_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table emp
#

CREATE TABLE `emp` (
  `EMP_id` bigint(20) NOT NULL,
  `EMP_Pos` varchar(60) NOT NULL,
  `EMP_FirstN` varchar(60) NOT NULL,
  `EMP_LastN` varchar(60) NOT NULL,
  `EMP_Username` varchar(20) NOT NULL,
  `EMP_Password` varchar(20) NOT NULL,
  `EMP_Email` varchar(30) NOT NULL,
  `EMP_CardIdent` varchar(32) NOT NULL,
  `EMP_Section` varchar(50) NOT NULL,
  `EMP_EMBD_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`EMP_id`),
  KEY `EMP_EMBD_id` (`EMP_EMBD_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table empd
#

CREATE TABLE `empd` (
  `EMPD_id` bigint(20) NOT NULL,
```

```
  `EMPD_Egn` varchar(10) NOT NULL,
  `EMPD_BirthD` date NOT NULL default '0000-00-00',
  `EMPD_Sex` char(1) NOT NULL ,
  `EMPD_LengthOfService` varchar(16) NOT NULL,
  `EMPD_CountLeave` smallint(6) default NULL,
  `EMPD_CountSickLeave` smallint(6) default NULL,
  `EMPD_EntranceOffice` date NOT NULL default '0000-00-00',
  `EMPD_Education` varchar(32) NOT NULL,
  `EMPD_TypeContract` varchar(32) NOT NULL,
  PRIMARY KEY  (`EMPD_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table hist
#

CREATE TABLE `hist` (
  `HIST_id` bigint(20) NOT NULL,
  `HIST_ART_id` bigint(20) NOT NULL,
  `HIST_ORD_id` bigint(20) NOT NULL,
  `HIST_OD_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`HIST_id`),
  KEY `HIST_ART_id` (`HIST_ART_id`),
  KEY `HIST_ORD_id` (`HIST_ORD_id`),
  KEY `HIST_OD_id` (`HIST_OD_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;




#
# Table structure for table oasm
#

CREATE TABLE `oasm` (
  `OASM_OO_id` bigint(20) NOT NULL,
  `OASM_ART_id` bigint(20) NOT NULL,
  `OASM_OOSM_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`OASM_OO_id`,`OASM_ART_id`,`OASM_OOSM_id`),
  KEY `OASM_ART_id` (`OASM_ART_id`),
  KEY `OASM_OOSM_id` (`OASM_OOSM_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table od
#

CREATE TABLE `od` (
  `OD_id` bigint(20) NOT NULL,
  `OD_quantity` int(11) NOT NULL,
  `OD_quantityprice` decimal(8,2) unsigned NOT NULL default '0.00',
  `OD_parametris` varchar(32) NOT NULL,
  `OD_verification` tinyint(1) NOT NULL,
  `OD_MakeOffer` tinyint(4) NOT NULL,
  `OD_DateMake` date NOT NULL default '0000-00-00',
  `OD_DateAccept` timestamp NOT NULL default CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP,
  PRIMARY KEY  (`OD_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table odoo
#

CREATE TABLE `odoo` (
  `ODOO_id` bigint(20) NOT NULL,
  `ODOO_OD_id` bigint(20) NOT NULL,
  `ODOO_OO_id` bigint(20) NOT NULL,
```

```
  `ODOO_EMP_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`ODOO_id`),
  KEY `FK_odoo_od_id` (`ODOO_OD_id`),
  KEY `FK_odoo_oo_id` (`ODOO_OO_id`),
  KEY `FK_odoo_emp_id` (`ODOO_EMP_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table oesu
#

CREATE TABLE `oesu` (
  `OESU_OD_id` bigint(20) NOT NULL,
  `OESU_EMP_id` bigint(20) NOT NULL,
  `OESU_SSO_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`OESU_OD_id`,`OESU_EMP_id`,`OESU_SSO_id`),
  KEY `OESU_EMP_id` (`OESU_EMP_id`),
  KEY `OESU_SSO_id` (`OESU_SSO_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


#
# Table structure for table oo
#

CREATE TABLE `oo` (
  `OO_id` bigint(20) NOT NULL,
  `OO_ControlOrder` varchar(128) NOT NULL,
  `OO_DateControl` date NOT NULL default '0000-00-00',
  `OO_InvNum` int(11) NOT NULL,
  `OO_DateInvoice` date NOT NULL default '0000-00-00',
  PRIMARY KEY  (`OO_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table oosm
#

CREATE TABLE `oosm` (
  `OOSM_id` bigint(20) NOT NULL,
  `oosm_perscust` tinyint(1) NOT NULL,
  `oosm_datepers` date default NULL,
  `oosm_couriername` varchar(16) default NULL,
  `oosm_datecourier` date default NULL,
  PRIMARY KEY  (`OOSM_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table orcu
#

CREATE TABLE `orcu` (
  `ORCU_id` bigint(20) NOT NULL,
  `ORCU_CUST_id` bigint(20) NOT NULL,
  `ORCU_ORD_id` bigint(20) NOT NULL,
  `ORCU_ODOO_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`ORCU_id`),
  KEY `ORCU_CUST_id` (`ORCU_CUST_id`),
  KEY `ORCU_ORD_id` (`ORCU_ORD_id`),
  KEY `ORCU_ODOO_id` (`ORCU_ODOO_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table ord
#

CREATE TABLE `ord` (
```

```
  `ORD_id` bigint(20) NOT NULL,
  `ORD_incn` int(11) NOT NULL,
  `ORD_date` date NOT NULL default '0000-00-00',
  `ORD_startdate` timestamp NOT NULL default CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP,
  `ORD_enddate` timestamp NOT NULL default '0000-00-00 00:00:00',
  `ORD_releasedate` timestamp NOT NULL default '0000-00-00 00:00:00',
  `ORD_OOSM_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`ORD_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;



#
# Table structure for table rait
#

CREATE TABLE `rait` (
  `RAIT_id` bigint(20) NOT NULL,
  `RAIT_Indicator` varchar(64) default NULL,
  `RAIT_Value` tinyint(4) default NULL,
  PRIMARY KEY  (`RAIT_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table sal
#

CREATE TABLE `sal` (
  `SAL_id` bigint(20) NOT NULL,
  `SAL_ART_id` bigint(20) NOT NULL,
  `SAL_ORD_id` bigint(20) NOT NULL,
  `SAL_SSO_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`SAL_id`),
  KEY `SAL_ART_id` (`SAL_ART_id`),
  KEY `SAL_ORD_id` (`SAL_ORD_id`),
  KEY `SAL_SSO_id` (`SAL_SSO_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table sso
#

CREATE TABLE `sso` (
  `SSO_id` bigint(20) NOT NULL,
  `SSO_Key` int(11) NOT NULL,
  `SSO_Description` text NOT NULL,
  `SSO_Depend` varchar(150) NOT NULL,
  `SSO_UnitPrice` decimal(6,2) unsigned NOT NULL default '0.00',
  `SSO_Currency` varchar(4) NOT NULL default '',
  `SSO_StartDate` date NOT NULL default '0000-00-00',
  `SSO_EndDate` date default NULL,
  PRIMARY KEY  (`SSO_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Table structure for table type
#

CREATE TABLE `type` (
  `TYPE_id` bigint(20) NOT NULL,
  `TYPE_Cust` tinyint(4) NOT NULL,
  PRIMARY KEY  (`TYPE_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
#  Foreign keys for table acf
```

```
#

ALTER TABLE `acf`
  ADD FOREIGN KEY (`ACF_CUST_id`) REFERENCES `cust` (`CUST_id`);

#
#  Foreign keys for table acoi
#

ALTER TABLE `acoi`
  ADD FOREIGN KEY (`ACOI_Emp_Send_id`) REFERENCES `emp` (`EMP_id`),
  ADD FOREIGN KEY (`ACOI_Emp_Accept_id`) REFERENCES `emp` (`EMP_id`);

#
#  Foreign keys for table ad
#

ALTER TABLE `ad`
  ADD FOREIGN KEY (`AD_ART_id`) REFERENCES `art` (`ART_id`),
  ADD FOREIGN KEY (`AD_DOC_id`) REFERENCES `doc` (`DOC_id`);

#
#  Foreign keys for table am
#

ALTER TABLE `am`
  ADD FOREIGN KEY (`AM_AME_id`) REFERENCES `ame` (`AME_id`),
  ADD FOREIGN KEY (`AM_AMD_id`) REFERENCES `amd` (`AMD_id`),
  ADD FOREIGN KEY (`AM_ORD_id`) REFERENCES `ord` (`ORD_id`);

#
#  Foreign keys for table ame
#

ALTER TABLE `ame`
  ADD FOREIGN KEY (`AME_Emp_Send_id`) REFERENCES `emp` (`EMP_id`),
  ADD FOREIGN KEY (`AME_Emp_Accept_id`) REFERENCES `emp` (`EMP_id`);

#
#  Foreign keys for table aoa
#

ALTER TABLE `aoa`
  ADD FOREIGN KEY (`AOA_ACF_id`) REFERENCES `acf` (`ACF_id`),
  ADD FOREIGN KEY (`AOA_ACOI_id`) REFERENCES `acoi` (`ACOI_id`),
  ADD FOREIGN KEY (`AOA_ARC_id`) REFERENCES `arc` (`ARC_id`);

#
#  Foreign keys for table aoe
#

ALTER TABLE `aoe`
  ADD FOREIGN KEY (`AOE_ART_id`) REFERENCES `art` (`ART_id`),
  ADD FOREIGN KEY (`AOE_ASt_id`) REFERENCES `ast` (`ASt_id`),
  ADD FOREIGN KEY (`AOE_AE_id`) REFERENCES `ae` (`AE_id`),
  ADD FOREIGN KEY (`AOE_AM_id`) REFERENCES `am` (`AM_id`);

#
#  Foreign keys for table ar
#

ALTER TABLE `ar`
  ADD FOREIGN KEY (`AR_Cust_id`) REFERENCES `cust` (`CUST_id`),
  ADD FOREIGN KEY (`AR_Rait_id`) REFERENCES `rait` (`RAIT_id`);

#
#  Foreign keys for table arc
```

```
#

ALTER TABLE `arc`
  ADD FOREIGN KEY (`ARC_EMP_id`) REFERENCES `emp` (`EMP_id`),
  ADD FOREIGN KEY (`ARC_CUST_id`) REFERENCES `cust` (`CUST_id`);

#
#  Foreign keys for table caac
#

ALTER TABLE `caac`
  ADD FOREIGN KEY (`CAAC_CUST_id`) REFERENCES `cust` (`CUST_id`) ON DELETE CASCADE ON UPDATE
CASCADE,
  ADD FOREIGN KEY (`CAAC_ADDR_id`) REFERENCES `addr` (`ADDR_id`) ON DELETE CASCADE ON UPDATE
CASCADE,
  ADD FOREIGN KEY (`CAAC_CC_id`) REFERENCES `cc` (`CC_id`) ON DELETE CASCADE ON UPDATE
CASCADE;

#
#  Foreign keys for table cc
#

ALTER TABLE `cc`
  ADD FOREIGN KEY (`CC_TYPE_id`) REFERENCES `type` (`TYPE_id`);

#
#  Foreign keys for table emp
#

ALTER TABLE `emp`
  ADD FOREIGN KEY (`EMP_EMBD_id`) REFERENCES `empd` (`EMPD_id`);

#
#  Foreign keys for table hist
#

ALTER TABLE `hist`
  ADD FOREIGN KEY (`HIST_ART_id`) REFERENCES `art` (`ART_id`),
  ADD FOREIGN KEY (`HIST_ORD_id`) REFERENCES `ord` (`ORD_id`),
  ADD FOREIGN KEY (`HIST_OD_id`) REFERENCES `od` (`OD_id`);

#
#  Foreign keys for table oasm
#

ALTER TABLE `oasm`
  ADD FOREIGN KEY (`OASM_OO_id`) REFERENCES `oo` (`OO_id`),
  ADD FOREIGN KEY (`OASM_ART_id`) REFERENCES `art` (`ART_id`),
  ADD FOREIGN KEY (`OASM_OOSM_id`) REFERENCES `oosm` (`OOSM_id`);

#
#  Foreign keys for table odoo
#

ALTER TABLE `odoo`
  ADD FOREIGN KEY (`ODOO_EMP_id`) REFERENCES `emp` (`EMP_id`),
  ADD FOREIGN KEY (`ODOO_OD_id`) REFERENCES `od` (`OD_id`),
  ADD FOREIGN KEY (`ODOO_OO_id`) REFERENCES `oo` (`OO_id`);

#
#  Foreign keys for table oesu
#

ALTER TABLE `oesu`
  ADD FOREIGN KEY (`OESU_OD_id`) REFERENCES `od` (`OD_id`),
  ADD FOREIGN KEY (`OESU_EMP_id`) REFERENCES `emp` (`EMP_id`),
  ADD FOREIGN KEY (`OESU_SSO_id`) REFERENCES `sso` (`SSO_id`);
```

```
#
#  Foreign keys for table orcu
#

ALTER TABLE `orcu`
  ADD FOREIGN KEY (`ORCU_CUST_id`) REFERENCES `cust` (`CUST_id`),
  ADD FOREIGN KEY (`ORCU_ORD_id`) REFERENCES `ord` (`ORD_id`),
  ADD FOREIGN KEY (`ORCU_ODOO_id`) REFERENCES `odoo` (`ODOO_id`);

#
#  Foreign keys for table sal
#

ALTER TABLE `sal`
  ADD FOREIGN KEY (`SAL_ART_id`) REFERENCES `art` (`ART_id`),
  ADD FOREIGN KEY (`SAL_ORD_id`) REFERENCES `ord` (`ORD_id`),
  ADD FOREIGN KEY (`SAL_SSO_id`) REFERENCES `sso` (`SSO_id`);
```

Приложение 3

Класове CustIdentity.cs и CustPrincipal.cs, използвани в опитната постановка

```
using System;
using System.Security.Principal;
using System.Xml;

namespace CustSecurity
{
    public class CustIdentity : IIdentity
    {
        private bool isAuth;
        private string name;
        private string authType;
        private int id;

        public CustomIdentity()
        {
            this.isAuth = false;
            this.authType = String.Empty;
            this.name = String.Empty;
            this.id = -1;
        }

        public CustomIdentity(string userName)
        {
            this.id = this.AuthUserName(userName);
            this.name = userName;
            this.isAuth = true;
            this.authType = "Частен тип authentication.";
        }

        public int ID
        {
            get { return this.id; }
        }
        #region IIdentity Members
```

```
        public bool IsAuthenticated
        {
            get
            {
                return this.isAuth;
            }
        }
        public string Name
        {
            get
            {
                return this.name;
            }
        }

        public string AuthenticationType
        {
            get
            {
                return this.authType;
            }
        }

        #endregion
        private int AuthUserName(string name)
        {
            XmlTextReader xmlReader = new XmlTextReader("Users.xml");
            xmlReader.WhitespaceHandling = WhitespaceHandling.None;
            while (xmlReader.Read())
            {
                if (xmlReader["name"] == name)
                    return Int32.Parse(xmlReader["id"]);
            }
            throw new
System.Security.SecurityException(String.Format("Потребител {0} не е
намерен в базата данни.", name));
        }
    }
}
```

```
using System;
using System.Security.Principal;
using System.Xml;

namespace CustSecurity
{
    public class CustPrincipal : IPrincipal
    {
        private CustomIdentity indentity;
        private string role;

        public CustomPrincipal(CustomIdentity identity)
        {
            this.indentity = identity;
            this.role = this.GetUserRole();
        }
        public IIdentity Identity
        {
```

```
            get
            {
                return this.indentity;
            }
        }
        public bool IsInRole(string role)
        {
            return role == this.role;
        }
        private string GetUserRole()
        {
            XmlTextReader xmlReader = new XmlTextReader("Users.xml");
            xmlReader.WhitespaceHandling = WhitespaceHandling.None;
            while (xmlReader.Read())
            {
                if (xmlReader["name"] == this.indentity.Name)
                    return xmlReader["role"];
            }
            throw new
System.Security.SecurityException(String.Format("Ролята на потребителя {0}
не е намерена в базата данни.",
                this.indentity.Name));
        }
    }
}
```

Приложение 4

Клас AES
[C#]

```
using System;

namespace AesLib
{
  public class Aes  // Advanced Encryption Standard
  {
    public enum KeySize { Bits128, Bits192, Bits256 };  // key size, in
bits, for construtor

    private int Nb;         // block size in 32-bit words.  Always 4 for
AES.  (128 bits).
    private int Nk;         // key size in 32-bit words.  4, 6, 8.  (128,
192, 256 bits).
    private int Nr;         // number of rounds. 10, 12, 14.

    private byte[] key;     // the seed key. size will be 4 * keySize from
ctor.
    private byte[,] Sbox;   // Substitution box
```

```
    private byte[,] iSbox;  // inverse Substitution box
    private byte[,] w;      // key schedule array.
    private byte[,] Rcon;   // Round constants.
    private byte[,] State;  // State matrix

    public Aes(KeySize keySize, byte[] keyBytes)
    {
      SetNbNkNr(keySize);

      this.key = new byte[this.Nk * 4];  // 16, 24, 32 bytes
      keyBytes.CopyTo(this.key, 0);

      BuildSbox();
      BuildInvSbox();
      BuildRcon();
      KeyExpansion();  // expand the seed key into a key schedule and store
in w

    }  // Aes constructor

    public void Cipher(byte[] input, byte[] output)  // encipher 16-bit
input
    {
      // state = input
      this.State = new byte[4,Nb];  // always [4,4]
      for (int i = 0; i < (4 * Nb); ++i)
      {
        this.State[i % 4, i / 4] = input[i];
      }

      AddRoundKey(0);

      for (int round = 1; round <= (Nr - 1); ++round)  // main round loop
      {
        SubBytes();
        ShiftRows();
        MixColumns();
        AddRoundKey(round);
      }  // main round loop

      SubBytes();
      ShiftRows();
      AddRoundKey(Nr);

      // output = state
      for (int i = 0; i < (4 * Nb); ++i)
      {
        output[i] = this.State[i % 4, i / 4];
      }

    }  // Cipher()

    public void InvCipher(byte[] input, byte[] output)  // decipher 16-bit
input
    {
      // state = input
      this.State = new byte[4,Nb];  // always [4,4]
      for (int i = 0; i < (4 * Nb); ++i)
      {
        this.State[i % 4, i / 4] = input[i];
      }
```

```
      AddRoundKey(Nr);

      for (int round = Nr-1; round >= 1; --round)  // main round loop
      {
        InvShiftRows();
        InvSubBytes();
        AddRoundKey(round);
        InvMixColumns();
      }  // end main round loop for InvCipher

      InvShiftRows();
      InvSubBytes();
      AddRoundKey(0);

      // output = state
      for (int i = 0; i < (4 * Nb); ++i)
      {
        output[i] = this.State[i % 4, i / 4];
      }

    }  // InvCipher()

    private void SetNbNkNr(KeySize keySize)
    {
      this.Nb = 4;     // block size always = 4 words = 16 bytes = 128 bits
for AES

      if (keySize == KeySize.Bits128)
      {
        this.Nk = 4;   // key size = 4 words = 16 bytes = 128 bits
        this.Nr = 10;  // rounds for algorithm = 10
      }
      else if (keySize == KeySize.Bits192)
      {
        this.Nk = 6;   // 6 words = 24 bytes = 192 bits
        this.Nr = 12;
      }
      else if (keySize == KeySize.Bits256)
      {
        this.Nk = 8;   // 8 words = 32 bytes = 256 bits
        this.Nr = 14;
      }
    }  // SetNbNkNr()

    private void BuildSbox()
    {
      this.Sbox = new byte[16,16] {  // populate the Sbox matrix
    /* 0     1     2     3     4     5     6     7     8     9     a     b
c     d     e     f */
    /*0*/  {0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01,
0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76},
    /*1*/  {0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4,
0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0},
    /*2*/  {0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5,
0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15},
    /*3*/  {0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12,
0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75},
    /*4*/  {0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b,
0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84},
```

```
    /*5*/  {0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb,
0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf},
    /*6*/  {0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9,
0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8},
    /*7*/  {0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6,
0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2},
    /*8*/  {0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7,
0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73},
    /*9*/  {0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee,
0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb},
    /*a*/  {0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3,
0xac, 0x62, 0x91, 0x95, 0xe4, 0x79},
    /*b*/  {0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56,
0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08},
    /*c*/  {0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd,
0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a},
    /*d*/  {0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35,
0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e},
    /*e*/  {0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e,
0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf},
    /*f*/  {0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99,
0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16} };

    }  // BuildSbox()

    private void BuildInvSbox()
    {
      this.iSbox = new byte[16,16] {  // populate the iSbox matrix
    /* 0     1     2     3     4     5     6     7     8     9     a     b
c     d     e     f */
    /*0*/  {0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40,
0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb},
    /*1*/  {0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e,
0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb},
    /*2*/  {0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c,
0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e},
    /*3*/  {0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b,
0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25},
    /*4*/  {0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4,
0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92},
    /*5*/  {0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15,
0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84},
    /*6*/  {0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4,
0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06},
    /*7*/  {0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf,
0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b},
    /*8*/  {0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2,
0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73},
    /*9*/  {0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9,
0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e},
    /*a*/  {0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7,
0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b},
    /*b*/  {0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb,
0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4},
    /*c*/  {0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12,
0x10, 0x59, 0x27, 0x80, 0xec, 0x5f},
    /*d*/  {0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5,
0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef},
    /*e*/  {0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb,
0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61},
```

```
    /*f*/  {0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69,
0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d} };

    }  // BuildInvSbox()

    private void BuildRcon()
    {
      this.Rcon = new byte[11,4] { {0x00, 0x00, 0x00, 0x00},
                                   {0x01, 0x00, 0x00, 0x00},
                                   {0x02, 0x00, 0x00, 0x00},
                                   {0x04, 0x00, 0x00, 0x00},
                                   {0x08, 0x00, 0x00, 0x00},
                                   {0x10, 0x00, 0x00, 0x00},
                                   {0x20, 0x00, 0x00, 0x00},
                                   {0x40, 0x00, 0x00, 0x00},
                                   {0x80, 0x00, 0x00, 0x00},
                                   {0x1b, 0x00, 0x00, 0x00},
                                   {0x36, 0x00, 0x00, 0x00} };
    }  // BuildRcon()

    private void AddRoundKey(int round)
    {

      for (int r = 0; r < 4; ++r)
      {
        for (int c = 0; c < 4; ++c)
        {
          this.State[r,c] = (byte) ( (int)this.State[r,c] ^
(int)w[(round*4)+c,r] );
        }
      }
    }  // AddRoundKey()

    private void SubBytes()
    {
      for (int r = 0; r < 4; ++r)
      {
        for (int c = 0; c < 4; ++c)
        {
          this.State[r,c] = this.Sbox[ (this.State[r,c] >> 4),
(this.State[r,c] & 0x0f) ];
        }
      }
    }  // SubBytes

    private void InvSubBytes()
    {
      for (int r = 0; r < 4; ++r)
      {
        for (int c = 0; c < 4; ++c)
        {
          this.State[r,c] = this.iSbox[ (this.State[r,c] >> 4),
(this.State[r,c] & 0x0f) ];
        }
      }
    }  // InvSubBytes

    private void ShiftRows()
    {
      byte[,] temp = new byte[4,4];
      for (int r = 0; r < 4; ++r)  // copy State into temp[]
```

```
      {
        for (int c = 0; c < 4; ++c)
        {
          temp[r,c] = this.State[r,c];
        }
      }

      for (int r = 1; r < 4; ++r)  // shift temp into State
      {
        for (int c = 0; c < 4; ++c)
        {
          this.State[r,c] = temp[ r, (c + r) % Nb ];
        }
      }
    }  // ShiftRows()

    private void InvShiftRows()
    {
      byte[,] temp = new byte[4,4];
      for (int r = 0; r < 4; ++r)  // copy State into temp[]
      {
        for (int c = 0; c < 4; ++c)
        {
          temp[r,c] = this.State[r,c];
        }
      }
      for (int r = 1; r < 4; ++r)  // shift temp into State
      {
        for (int c = 0; c < 4; ++c)
        {
          this.State[r, (c + r) % Nb ] = temp[r,c];
        }
      }
    }  // InvShiftRows()

    private void MixColumns()
    {
      byte[,] temp = new byte[4,4];
      for (int r = 0; r < 4; ++r)  // copy State into temp[]
      {
        for (int c = 0; c < 4; ++c)
        {
          temp[r,c] = this.State[r,c];
        }
      }

      for (int c = 0; c < 4; ++c)
      {
        this.State[0,c] = (byte) ( (int)gfmultby02(temp[0,c]) ^
(int)gfmultby03(temp[1,c]) ^
                                   (int)gfmultby01(temp[2,c]) ^
(int)gfmultby01(temp[3,c]) );
        this.State[1,c] = (byte) ( (int)gfmultby01(temp[0,c]) ^
(int)gfmultby02(temp[1,c]) ^
                                   (int)gfmultby03(temp[2,c]) ^
(int)gfmultby01(temp[3,c]) );
        this.State[2,c] = (byte) ( (int)gfmultby01(temp[0,c]) ^
(int)gfmultby01(temp[1,c]) ^
                                   (int)gfmultby02(temp[2,c]) ^
(int)gfmultby03(temp[3,c]) );
```

```
        this.State[3,c] = (byte) ( (int)gfmultby03(temp[0,c]) ^
(int)gfmultby01(temp[1,c]) ^
                                   (int)gfmultby01(temp[2,c]) ^
(int)gfmultby02(temp[3,c]) );
      }
    }  // MixColumns

    private void InvMixColumns()
    {
      byte[,] temp = new byte[4,4];
      for (int r = 0; r < 4; ++r)  // copy State into temp[]
      {
        for (int c = 0; c < 4; ++c)
        {
          temp[r,c] = this.State[r,c];
        }
      }

      for (int c = 0; c < 4; ++c)
      {
        this.State[0,c] = (byte) ( (int)gfmultby0e(temp[0,c]) ^
(int)gfmultby0b(temp[1,c]) ^
                                   (int)gfmultby0d(temp[2,c]) ^
(int)gfmultby09(temp[3,c]) );
        this.State[1,c] = (byte) ( (int)gfmultby09(temp[0,c]) ^
(int)gfmultby0e(temp[1,c]) ^
                                   (int)gfmultby0b(temp[2,c]) ^
(int)gfmultby0d(temp[3,c]) );
        this.State[2,c] = (byte) ( (int)gfmultby0d(temp[0,c]) ^
(int)gfmultby09(temp[1,c]) ^
                                   (int)gfmultby0e(temp[2,c]) ^
(int)gfmultby0b(temp[3,c]) );
        this.State[3,c] = (byte) ( (int)gfmultby0b(temp[0,c]) ^
(int)gfmultby0d(temp[1,c]) ^
                                   (int)gfmultby09(temp[2,c]) ^
(int)gfmultby0e(temp[3,c]) );
      }
    }  // InvMixColumns

    private static byte gfmultby01(byte b)
    {
      return b;
    }

    private static byte gfmultby02(byte b)
    {
      if (b < 0x80)
        return (byte)(int)(b <<1);
      else
        return (byte)( (int)(b << 1) ^ (int)(0x1b) );
    }

    private static byte gfmultby03(byte b)
    {
      return (byte) ( (int)gfmultby02(b) ^ (int)b );
    }

    private static byte gfmultby09(byte b)
    {
      return (byte)( (int)gfmultby02(gfmultby02(gfmultby02(b))) ^
                     (int)b );
```

```
    }

    private static byte gfmultby0b(byte b)
    {
      return (byte)( (int)gfmultby02(gfmultby02(gfmultby02(b))) ^
                     (int)gfmultby02(b) ^
                     (int)b );
    }

    private static byte gfmultby0d(byte b)
    {
      return (byte)( (int)gfmultby02(gfmultby02(gfmultby02(b))) ^
                     (int)gfmultby02(gfmultby02(b)) ^
                     (int)(b) );
    }

    private static byte gfmultby0e(byte b)
    {
      return (byte)( (int)gfmultby02(gfmultby02(gfmultby02(b))) ^
                     (int)gfmultby02(gfmultby02(b)) ^
                     (int)gfmultby02(b) );
    }

    private void KeyExpansion()
    {
      this.w = new byte[Nb * (Nr+1), 4];  // 4 columns of bytes corresponds
to a word

      for (int row = 0; row < Nk; ++row)
      {
        this.w[row,0] = this.key[4*row];
        this.w[row,1] = this.key[4*row+1];
        this.w[row,2] = this.key[4*row+2];
        this.w[row,3] = this.key[4*row+3];
      }

      byte[] temp = new byte[4];

      for (int row = Nk; row < Nb * (Nr+1); ++row)
      {
        temp[0] = this.w[row-1,0]; temp[1] = this.w[row-1,1];
        temp[2] = this.w[row-1,2]; temp[3] = this.w[row-1,3];

        if (row % Nk == 0)
        {
          temp = SubWord(RotWord(temp));

          temp[0] = (byte)( (int)temp[0] ^ (int)this.Rcon[row/Nk,0] );
          temp[1] = (byte)( (int)temp[1] ^ (int)this.Rcon[row/Nk,1] );
          temp[2] = (byte)( (int)temp[2] ^ (int)this.Rcon[row/Nk,2] );
          temp[3] = (byte)( (int)temp[3] ^ (int)this.Rcon[row/Nk,3] );
        }
        else if ( Nk > 6 && (row % Nk == 4) )
        {
          temp = SubWord(temp);
        }

        // w[row] = w[row-Nk] xor temp
        this.w[row,0] = (byte) ( (int)this.w[row-Nk,0] ^ (int)temp[0] );
        this.w[row,1] = (byte) ( (int)this.w[row-Nk,1] ^ (int)temp[1] );
        this.w[row,2] = (byte) ( (int)this.w[row-Nk,2] ^ (int)temp[2] );
```

```
        this.w[row,3] = (byte) ( (int)this.w[row-Nk,3] ^ (int)temp[3] );

      }  // for loop
    }  // KeyExpansion()

    private byte[] SubWord(byte[] word)
    {
      byte[] result = new byte[4];
      result[0] = this.Sbox[ word[0] >> 4, word[0] & 0x0f ];
      result[1] = this.Sbox[ word[1] >> 4, word[1] & 0x0f ];
      result[2] = this.Sbox[ word[2] >> 4, word[2] & 0x0f ];
      result[3] = this.Sbox[ word[3] >> 4, word[3] & 0x0f ];
      return result;
    }

    private byte[] RotWord(byte[] word)
    {
      byte[] result = new byte[4];
      result[0] = word[1];
      result[1] = word[2];
      result[2] = word[3];
      result[3] = word[0];
      return result;
    }

    public  void Dump()
    {
      Console.WriteLine("Nb = " + Nb + " Nk = " + Nk + " Nr = " + Nr);
      Console.WriteLine("\nThe key is \n" + DumpKey() );
      Console.WriteLine("\nThe Sbox is \n" + DumpTwoByTwo(Sbox));
      Console.WriteLine("\nThe w array is \n" + DumpTwoByTwo(w));
      Console.WriteLine("\nThe State array is \n" + DumpTwoByTwo(State));
    }

    public string DumpKey()
    {
      string s = "";
      for (int i = 0; i < key.Length; ++i)
        s += key[i].ToString("x2") + " ";
      return s;
    }

    public string DumpTwoByTwo(byte[,] a)
    {
      string s ="";
      for (int r = 0; r < a.GetLength(0); ++r)
      {
        s += "["+r+"]" + " ";
        for (int c = 0; c < a.GetLength(1); ++c)
        {
          s += a[r,c].ToString("x2") + " " ;
        }
        s += "\n";
      }
      return s;
    }

  }  // class Aes
}  // ns AesLib
```