ВВЕДЕНИЕ (вводная часть, постановка целей, задач)

1. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

**1.1 Описание предметной области** (выделение сущностей, описание характеристик выделенных сущностей, описание автоматизируемых процессов, описание участников автоматизируемых процессов)

**1.2. Построение ER-модели** (у меня есть картинка модели, я тебе покажу отдельно, но нужно описать эту ER-модель)

**1.3 Логическая модель**.

**1.4 Физическая модель**.

**1.5 Нормализация полученных отношений** (процесс перехода от 1НФ к 3НФ нужно описать)

2. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ.

**2.1 Процесс создания таблиц БД**

**2.2 Процесс создания связей между таблицами**.

3. РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ РАБОТЫ С БАЗОЙ ДАННЫХ.

**3.1 Процесс формирования форм для работы с таблицами базы данных**.

**3.2 Процесс формирования запросов**.

**3.3 Процесс формирования отчетов**.

**3.4 Процесс создания главной кнопочной формы**

**3.5 Разграничение доступа к БД**

ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ

ЗАКЛЮЧЕНИЕ

ВВЕДЕНИЕ

В современных условиях эффективное управление медицинскими учреждениями невозможно без использования информационных технологий. Автоматизация процессов регистрации пациентов, ведения медицинских карт, выдачи талонов на прием к врачам позволяет существенно повысить качество обслуживания, сократить время обработки данных и избежать ошибок, связанных с человеческим фактором.

Данная курсовая работа посвящена проектированию и разработке базы данных для автоматизации работы регистратуры поликлиники. Предметной областью являются данные о пациентах, врачах, медицинских картах, талонах и диагнозах. Создание единой информационной системы позволит систематизировать эти данные, облегчить доступ к ним и автоматизировать основные процессы работы регистратуры.

Цель работы - спроектировать и реализовать базу данных "Регистратура", предназначенную для автоматизации учета пациентов, ведения медицинских карт, выдачи талонов на прием к врачам, а также хранения справочной информации о врачах и диагнозах.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Выполнить анализ предметной области, выделить основные объекты автоматизации и их характеристики.
2. Построить концептуальную модель данных в виде ER-диаграммы.
3. Разработать логическую и физическую модели базы данных, выполнить нормализацию отношений.
4. Реализовать базу данных средствами СУБД Microsoft Access, создать таблицы и связи между ними.
5. Разработать приложение для работы с базой данных, включающее формы для ввода и редактирования данных, запросы и отчеты.
6. Реализовать механизм разграничения доступа к данным для различных категорий пользователей.
7. Подготовить инструкцию пользователя по работе с приложением.

Практическая значимость работы заключается в возможности использования разработанной базы данных и приложения для повышения эффективности работы регистратуры поликлиники, улучшения качества обслуживания пациентов и автоматизации рутинных операций по вводу и обработке данных.

1. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

1.1 Описание предметной области

Предметной областью для разрабатываемой базы данных является регистратура поликлиники. Основная задача регистратуры - организация приема пациентов врачами различных специальностей. Регистратура выполняет функции записи пациентов на прием, выдачи талонов, ведения медицинских карт, предоставления справочной информации о графике работы врачей и доступных услугах.

В рамках предметной области выделены следующие сущности:

1. Пациент - человек, обращающийся в поликлинику за медицинской помощью.
2. Врач - медицинский работник, оказывающий услуги по диагностике и лечению заболеваний.
3. Медицинская карта - документ, содержащий сведения о состоянии здоровья пациента, историю его обращений в поликлинику и результаты обследований.
4. Талон - документ, подтверждающий запись пациента на прием к врачу в определенное время.
5. Диагноз - заключение врача о состоянии здоровья пациента, поставленное на основании обследования и кодифицированное по международной классификации болезней (МКБ-10).

Характеристики выделенных сущностей:

* Пациент: ФИО, дата рождения, контактный телефон, номер медицинской карты.
* Врач: ФИО, специальность, график работы, номер кабинета.
* Медицинская карта: номер, дата заведения, сведения о прикреплении к участку, перенесенные заболевания, результаты обследований.
* Талон: номер, дата и время приема, ФИО пациента, ФИО врача, номер кабинета.
* Диагноз: код по МКБ-10, название, описание.

В процессе работы регистратуры необходимо автоматизировать следующие процессы:

1. Ведение справочников пациентов, врачей и диагнозов.
2. Запись пациентов на прием к врачам с учетом их специальности и графика работы.
3. Формирование и печать талонов на прием.
4. Ведение медицинских карт пациентов с возможностью добавления диагнозов и результатов обследований.
5. Поиск и выборка данных по различным критериям (по ФИО пациента, по специальности врача, по диагнозу и т.д.)
6. Формирование отчетов о работе регистратуры (количество принятых пациентов, нагрузка на врачей и т.п.)

Участниками автоматизируемых процессов являются:

1. Регистраторы - сотрудники регистратуры, осуществляющие запись пациентов на прием, выдачу талонов, ведение медицинских карт.
2. Врачи - медицинские работники, оказывающие услуги по диагностике и лечению заболеваний, ведущие прием пациентов.
3. Пациенты - лица, обращающиеся в поликлинику за медицинской помощью.

Между сущностями предметной области установлены следующие связи:

* Пациент - Медицинская карта: один-к-одному. Каждому пациенту соответствует одна медицинская карта.
* Пациент - Талон: один-ко-многим. Каждый пациент может иметь несколько талонов на прием, но каждый талон связан только с одним пациентом.
* Врач - Талон: один-ко-многим. Каждый врач может вести прием по нескольким талонам, но каждый талон связан только с одним врачом.
* Диагноз - Медицинская карта: многие-ко-многим. В одной медицинской карте может быть указано несколько диагнозов, и один диагноз может встречаться в нескольких картах. Для реализации этой связи потребуется дополнительная таблица.

Указанные связи позволят организовать хранение и обработку данных в соответствии с реальными процессами работы регистратуры, обеспечить целостность и непротиворечивость информации, упростить формирование запросов и отчетов.

1.2. Построение ER-модели

ER-модель (Entity-Relationship Model) - это концептуальная модель данных, которая используется для описания предметной области в терминах сущностей, их атрибутов и связей между ними. ER-модель позволяет наглядно представить структуру данных, выявить основные объекты автоматизации и определить правила их взаимодействия.

Основными элементами ER-модели являются:

1. Сущности - реальные или виртуальные объекты предметной области, информация о которых должна храниться в базе данных.
2. Атрибуты - свойства сущностей, характеризующие их состояние и однозначно идентифицирующие каждый экземпляр сущности.
3. Связи - ассоциации между сущностями, отражающие их взаимодействие в рамках предметной области.

Для предметной области "Регистратура" на ER-модели должны быть отражены следующие сущности:

1. Пациент (Patient) Атрибуты: ID\_Patient (ключевой атрибут), LastName, FirstName, MiddleName, BirthDate, Phone, ID\_MedCard (внешний ключ)
2. Врач (Doctor) Атрибуты: ID\_Doctor (ключевой атрибут), LastName, FirstName, MiddleName, Speciality, CabinetNumber
3. Медицинская карта (MedCard) Атрибуты: ID\_MedCard (ключевой атрибут), ID\_Patient (внешний ключ), StartDate
4. Талон (Ticket) Атрибуты: ID\_Ticket (ключевой атрибут), ID\_Patient (внешний ключ), ID\_Doctor (внешний ключ), VisitDateTime, CabinetNumber
5. Диагноз (Diagnosis) Атрибуты: ID\_Diagnosis (ключевой атрибут), DiagnosisCode (по МКБ-10), DiagnosisName, Description
6. Диагнозы в медкарте (MedCard\_Diagnosis) - ассоциативная сущность для реализации связи "многие-ко-многим" между сущностями MedCard и Diagnosis Атрибуты: ID\_MedCard (внешний ключ), ID\_Diagnosis (внешний ключ), DiagnosisDate

Связи между сущностями:

1. Пациент - Медицинская карта: один-к-одному. Каждому пациенту соответствует одна медицинская карта. Связь реализуется через внешний ключ ID\_MedCard в сущности Patient.
2. Пациент - Талон: один-ко-многим. Каждый пациент может иметь несколько талонов на прием, но каждый талон связан только с одним пациентом. Связь реализуется через внешний ключ ID\_Patient в сущности Ticket.
3. Врач - Талон: один-ко-многим. Каждый врач может вести прием по нескольким талонам, но каждый талон связан только с одним врачом. Связь реализуется через внешний ключ ID\_Doctor в сущности Ticket.
4. Медицинская карта - Диагноз: многие-ко-многим. В одной медицинской карте может быть указано несколько диагнозов, и один диагноз может встречаться в нескольких картах. Для реализации этой связи используется дополнительная ассоциативная сущность MedCard\_Diagnosis, которая содержит внешние ключи ID\_MedCard и ID\_Diagnosis.

ER-модель позволяет наглядно представить структуру базы данных, определить состав и взаимосвязи сущностей, выявить необходимые атрибуты и ключевые поля. На основе ER-модели в дальнейшем будет разработана логическая и физическая модели базы данных, сформированы таблицы и связи между ними.

1.3 Логическая модель

Логическая модель - это представление структуры базы данных на уровне таблиц, полей и связей между ними. Логическая модель разрабатывается на основе концептуальной (ER-модели) и является промежуточным звеном между концептуальным и физическим уровнями проектирования базы данных.

Основная цель построения логической модели - детализация и уточнение структуры данных, определение состава таблиц, атрибутов и связей в соответствии с правилами нормализации и требованиями конкретной СУБД. Логическая модель позволяет оптимизировать структуру базы данных, исключить избыточность и противоречивость данных, обеспечить целостность и непротиворечивость информации.

Для предметной области "Регистратура" логическая модель будет включать следующие таблицы:

1. Patient (Пациент)

* PatientID (INT, PRIMARY KEY)
* LastName (VARCHAR(50), NOT NULL)
* FirstName (VARCHAR(50), NOT NULL)
* MiddleName (VARCHAR(50))
* BirthDate (DATE, NOT NULL)
* Phone (VARCHAR(20))
* MedCardID (INT, FOREIGN KEY REFERENCES MedCard(MedCardID))

1. Doctor (Врач)

* DoctorID (INT, PRIMARY KEY)
* LastName (VARCHAR(50), NOT NULL)
* FirstName (VARCHAR(50), NOT NULL)
* MiddleName (VARCHAR(50))
* Speciality (VARCHAR(100), NOT NULL)
* CabinetNumber (VARCHAR(10), NOT NULL)

1. MedCard (Медицинская карта)

* MedCardID (INT, PRIMARY KEY)
* PatientID (INT, FOREIGN KEY REFERENCES Patient(PatientID))
* StartDate (DATE, NOT NULL)

1. Ticket (Талон)

* TicketID (INT, PRIMARY KEY)
* PatientID (INT, FOREIGN KEY REFERENCES Patient(PatientID))
* DoctorID (INT, FOREIGN KEY REFERENCES Doctor(DoctorID))
* VisitDateTime (DATETIME, NOT NULL)
* CabinetNumber (VARCHAR(10), NOT NULL)

1. Diagnosis (Диагноз)

* DiagnosisID (INT, PRIMARY KEY)
* DiagnosisCode (VARCHAR(20), NOT NULL)
* DiagnosisName (VARCHAR(255), NOT NULL)
* Description (TEXT)

1. MedCard\_Diagnosis (Диагнозы в медкарте)

* MedCardID (INT, FOREIGN KEY REFERENCES MedCard(MedCardID))
* DiagnosisID (INT, FOREIGN KEY REFERENCES Diagnosis(DiagnosisID))
* DiagnosisDate (DATE, NOT NULL)
* PRIMARY KEY (MedCardID, DiagnosisID)

Связи между таблицами:

1. Таблицы Patient и MedCard связаны отношением один-к-одному через внешний ключ MedCardID в таблице Patient.
2. Таблицы Patient и Ticket связаны отношением один-ко-многим через внешний ключ PatientID в таблице Ticket.
3. Таблицы Doctor и Ticket связаны отношением один-ко-многим через внешний ключ DoctorID в таблице Ticket.
4. Таблицы MedCard и Diagnosis связаны отношением многие-ко-многим через промежуточную таблицу MedCard\_Diagnosis, которая содержит внешние ключи MedCardID и DiagnosisID.

Для каждого поля таблиц определены тип данных, ограничения NULL/NOT NULL, ключевые поля (PRIMARY KEY) и внешние ключи (FOREIGN KEY) для организации связей между таблицами.

На основе логической модели в дальнейшем будет разработана физическая модель базы данных, выполнено создание таблиц, индексов и связей средствами конкретной СУБД (в нашем случае - Microsoft Access).

1.4 Физическая модель

Физическая модель базы данных представляет собой реализацию логической модели средствами конкретной СУБД с учетом особенностей ее архитектуры, типов данных и ограничений. Физическая модель определяет структуру таблиц, индексов, связей, а также физическое размещение данных на носителях информации.

Для реализации базы данных "Регистратура" будет использована СУБД Microsoft Access. Ниже приведено описание таблиц и связей физической модели:

Таблица "Patient"

* PatientID (счетчик, первичный ключ)
* LastName (текстовый, обязательное поле)
* FirstName (текстовый, обязательное поле)
* MiddleName (текстовый)
* BirthDate (дата/время, обязательное поле)
* Phone (текстовый)
* MedCardID (число, внешний ключ, подстановка из таблицы "MedCard")

Таблица "Doctor"

* DoctorID (счетчик, первичный ключ)
* LastName (текстовый, обязательное поле)
* FirstName (текстовый, обязательное поле)
* MiddleName (текстовый)
* Speciality (текстовый, обязательное поле)
* CabinetNumber (текстовый, обязательное поле)

Таблица "MedCard"

* MedCardID (счетчик, первичный ключ)
* PatientID (число, внешний ключ, подстановка из таблицы "Patient")
* StartDate (дата/время, обязательное поле)

Таблица "Ticket"

* TicketID (счетчик, первичный ключ)
* PatientID (число, внешний ключ, подстановка из таблицы "Patient")
* DoctorID (число, внешний ключ, подстановка из таблицы "Doctor")
* VisitDateTime (дата/время, обязательное поле)
* CabinetNumber (текстовый, обязательное поле)

Таблица "Diagnosis"

* DiagnosisID (счетчик, первичный ключ)
* DiagnosisCode (текстовый, обязательное поле)
* DiagnosisName (текстовый, обязательное поле)
* Description (длинный текст)

Таблица "MedCard\_Diagnosis"

* MedCardID (число, внешний ключ, подстановка из таблицы "MedCard")
* DiagnosisID (число, внешний ключ, подстановка из таблицы "Diagnosis")
* DiagnosisDate (дата/время, обязательное поле)

Связи между таблицами:

1. Связь один-к-одному между таблицами "Patient" и "MedCard" через поле MedCardID.
2. Связь один-ко-многим между таблицами "Patient" и "Ticket" через поле PatientID.
3. Связь один-ко-многим между таблицами "Doctor" и "Ticket" через поле DoctorID.
4. Связь многие-ко-многим между таблицами "MedCard" и "Diagnosis" через таблицу "MedCard\_Diagnosis".

Для обеспечения целостности данных и оптимизации производительности в таблицах будут созданы необходимые индексы по ключевым полям и внешним ключам.

При реализации физической модели также будут учтены требования к форматам данных, ограничениям на значения полей, правилам валидации и обработке исключительных ситуаций.

В дальнейшем на основе физической модели будет выполнено создание самой базы данных, таблиц, индексов и связей в среде Microsoft Access, а также разработаны объекты доступа к данным (формы, запросы, отчеты).

1.5 Нормализация полученных отношений

Нормализация - это процесс приведения структуры базы данных к виду, обеспечивающему минимальную логическую избыточность, непротиворечивость и целостность данных. Нормализация осуществляется путем последовательного приведения отношений к нормальным формам (1НФ, 2НФ, 3НФ и т.д.).

Алгоритм нормализации отношений:

Шаг 1. Приведение отношений к первой нормальной форме (1НФ):

* Удалить повторяющиеся группы данных, вынеся их в отдельные таблицы.
* Определить первичный ключ для каждой таблицы.
* Атомарность данных: каждый атрибут должен быть неделимым, не содержать множественных значений.

В нашей модели все отношения уже находятся в 1НФ, так как не содержат повторяющихся групп и многозначных атрибутов.

Шаг 2. Приведение отношений ко второй нормальной форме (2НФ):

* Каждый неключевой атрибут должен зависеть от первичного ключа полностью, а не только от его части.
* Если в составном первичном ключе есть атрибуты, от которых не зависят некоторые неключевые атрибуты, то их нужно вынести в отдельную таблицу вместе с этими неключевыми атрибутами.

В нашей модели все отношения находятся во 2НФ, так как в каждой таблице неключевые атрибуты зависят от первичного ключа полностью.

Шаг 3. Приведение отношений к третьей нормальной форме (3НФ):

* Ни один неключевой атрибут не должен зависеть от другого неключевого атрибута (должны быть устранены транзитивные зависимости).
* Если есть неключевые атрибуты, которые зависят от других неключевых атрибутов, их нужно вынести в отдельную таблицу.

Проверим наши отношения на соответствие 3НФ:

Таблица "Patient" - все атрибуты зависят только от первичного ключа PatientID, транзитивных зависимостей нет.

Таблица "Doctor" - все атрибуты зависят только от первичного ключа DoctorID, транзитивных зависимостей нет.

Таблица "MedCard" - все атрибуты зависят только от первичного ключа MedCardID, транзитивных зависимостей нет.

Таблица "Ticket" - все атрибуты зависят только от первичного ключа TicketID, транзитивных зависимостей нет.

Таблица "Diagnosis" - все атрибуты зависят только от первичного ключа DiagnosisID, транзитивных зависимостей нет.

Таблица "MedCard\_Diagnosis" - составной первичный ключ (MedCardID, DiagnosisID), неключевой атрибут DiagnosisDate зависит от первичного ключа полностью.

Таким образом, все отношения в нашей модели уже находятся в 3НФ, поэтому дополнительная декомпозиция не требуется.

В результате нормализации мы получили структуру базы данных, которая обеспечивает минимальную избыточность, непротиворечивость и целостность данных, а также упрощает дальнейшую разработку приложения.

1. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

В этом разделе мы рассмотрим процесс физической реализации базы данных "Регистратура" в СУБД Microsoft Access. Физическое проектирование включает в себя создание таблиц, определение типов данных, установление связей между таблицами и определение ограничений целостности.

2.1 Процесс создания таблиц БД

Для создания таблиц в Microsoft Access выполните следующие шаги:

1. Откройте Microsoft Access и создайте новую пустую базу данных.
2. Во вкладке "Создание" выберите пункт "Конструктор таблиц".
3. В открывшемся окне введите имя первой таблицы (например, "Patient") и начните определять поля:

* В столбце "Имя поля" введите название атрибута (например, "PatientID").
* В столбце "Тип данных" выберите соответствующий тип (например, "Счетчик" для автоинкрементного числового поля).
* При необходимости введите описание поля и определите свойства (размер, формат, маску ввода и т.д.).

1. Повторите шаг 3 для каждого атрибута таблицы "Patient". Ниже приведены типы данных для атрибутов:

* PatientID - Счетчик
* LastName - Короткий текст
* FirstName - Короткий текст
* MiddleName - Короткий текст
* BirthDate - Дата/время
* Phone - Короткий текст
* MedCardID - Длинное целое число

1. Определите первичный ключ таблицы, выделив поле "PatientID" и нажав на кнопку "Ключевое поле" на панели инструментов.
2. Сохраните структуру таблицы, закройте окно конструктора и введите имя таблицы (например, "Patient").
3. Аналогичным образом создайте остальные таблицы со следующей структурой:

Таблица "Doctor":

* DoctorID - Счетчик
* LastName - Короткий текст
* FirstName - Короткий текст
* MiddleName - Короткий текст
* Speciality - Короткий текст
* CabinetNumber - Короткий текст

Таблица "MedCard":

* MedCardID - Счетчик
* PatientID - Длинное целое число
* StartDate - Дата/время

Таблица "Ticket":

* TicketID - Счетчик
* PatientID - Длинное целое число
* DoctorID - Длинное целое число
* VisitDateTime - Дата/время
* CabinetNumber - Короткий текст

Таблица "Diagnosis":

* DiagnosisID - Счетчик
* DiagnosisCode - Короткий текст
* DiagnosisName - Короткий текст
* Description - Длинный текст

Таблица "MedCard\_Diagnosis":

* MedCardID - Длинное целое число
* DiagnosisID - Длинное целое число
* DiagnosisDate - Дата/время

После создания всех таблиц ваша база данных будет готова к установлению связей между таблицами и дальнейшей разработке приложения.

2.2 Процесс создания связей между таблицами

Для создания связей между таблицами в Microsoft Access выполните следующие шаги:

1. Откройте вкладку "Работа с базами данных" и выберите пункт "Схема данных".
2. В открывшемся окне "Добавление таблицы" выберите все необходимые таблицы и нажмите "Добавить", затем закройте окно.
3. На схеме данных расположите таблицы так, чтобы связываемые поля были рядом друг с другом.
4. Для создания связи один-ко-многим перетащите первичный ключ (например, "PatientID") из главной таблицы ("Patient") на соответствующее поле внешнего ключа (например, "PatientID") в подчиненной таблице ("Ticket").
5. В появившемся окне "Изменение связей" установите флажки "Обеспечение целостности данных", "Каскадное обновление связанных полей" и "Каскадное удаление связанных записей". Это обеспечит согласованность данных при изменениях в главной таблице.
6. Повторите шаги 4-5 для создания остальных связей:

* Связь один-к-одному между таблицами "Patient" и "MedCard" через поля "PatientID" и "MedCardID".
* Связь один-ко-многим между таблицами "Doctor" и "Ticket" через поля "DoctorID".

1. Для связи многие-ко-многим между таблицами "MedCard" и "Diagnosis" создайте две связи один-ко-многим через промежуточную таблицу "MedCard\_Diagnosis":

* Связь между "MedCard" и "MedCard\_Diagnosis" через поле "MedCardID".
* Связь между "Diagnosis" и "MedCard\_Diagnosis" через поле "DiagnosisID".

После создания всех связей ваша схема данных будет отражать логическую структуру базы данных.

Правила обеспечения целостности данных:

1. Целостность сущностей: каждая запись в таблице должна быть уникальной и идентифицируемой по первичному ключу.
2. Целостность по ссылкам: внешние ключи в подчиненных таблицах должны ссылаться на существующие записи в главных таблицах.
3. Каскадное обновление связанных полей: при изменении значения первичного ключа в главной таблице автоматически обновляются соответствующие значения внешних ключей в подчиненных таблицах.
4. Каскадное удаление связанных записей: при удалении записи в главной таблице автоматически удаляются все связанные записи в подчиненных таблицах.
5. Ограничения на значения полей: с помощью свойств полей (например, "Обязательное поле", "Условие на значение") можно задать допустимые значения и форматы данных.

Соблюдение этих правил поможет обеспечить согласованность и непротиворечивость данных в вашей базе данных.

1. РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ РАБОТЫ С БАЗОЙ ДАННЫХ

3.1 Процесс формирования форм для работы с таблицами базы данных

Формы в Microsoft Access - это объекты базы данных, которые служат для удобного ввода, просмотра и редактирования данных в таблицах. Формы предоставляют пользовательский интерфейс, позволяющий работать с данными без непосредственного взаимодействия с таблицами.

Для создания форм в Microsoft Access выполните следующие шаги:

1. На вкладке "Создание" выберите пункт "Мастер форм".
2. В открывшемся окне выберите таблицу или запрос, на основе которых будет создаваться форма (например, "Patient"), и нажмите "Далее".
3. Выберите поля, которые должны отображаться в форме (например, "PatientID", "LastName", "FirstName", "MiddleName", "BirthDate", "Phone", "MedCardID"), и нажмите "Далее".
4. Выберите внешний вид формы (например, в один столбец) и нажмите "Далее".
5. Выберите стиль оформления формы и нажмите "Далее".
6. Введите имя формы (например, "frmPatient") и выберите опцию "Открыть форму для просмотра и ввода данных", затем нажмите "Готово".

Созданная форма откроется в режиме просмотра и будет готова к вводу и редактированию данных.

Для нашей базы данных "Регистратура" рекомендуется создать следующие формы:

1. frmPatient - форма для ввода и редактирования данных о пациентах.
2. frmDoctor - форма для ввода и редактирования данных о врачах.
3. frmMedCard - форма для просмотра и редактирования медицинских карт пациентов.
4. frmTicket - форма для создания и редактирования талонов на прием к врачу.
5. frmDiagnosis - форма для ввода и редактирования справочника диагнозов.
6. frmMedCardDiagnosis - форма для ввода и редактирования диагнозов в медицинских картах пациентов.

Кроме того, можно создать дополнительные формы для удобства навигации и работы с данными:

1. frmMain - главная форма приложения с кнопками для открытия других форм.
2. frmSearch - форма для поиска пациентов, врачей или талонов по различным критериям.
3. frmReports - форма для формирования и просмотра отчетов.

При создании форм следует учитывать эргономичность и удобство использования, располагать поля и элементы управления в логическом порядке, использовать понятные подписи и подсказки.

После создания всех необходимых форм можно приступать к разработке запросов и отчетов для обработки и анализа данных в базе данных.

Да, для реализации дополнительных форм, таких как frmSearch и frmReports, потребуются запросы.

Запросы в Microsoft Access - это объекты, которые позволяют извлекать, фильтровать и обрабатывать данные из одной или нескольких таблиц базы данных. С помощью запросов можно выполнять выборку данных по определенным критериям, объединять данные из разных таблиц, рассчитывать итоговые значения и т.д.

Например, для формы frmSearch потребуются запросы для поиска:

1. Пациентов по ФИО, дате рождения или номеру медицинской карты.
2. Врачей по ФИО, специальности или номеру кабинета.
3. Талонов по дате приема, ФИО пациента или врача.

Для формы frmReports запросы будут использоваться для формирования отчетов:

1. Список пациентов, прикрепленных к определенному врачу.
2. Количество талонов, выданных за определенный период.
3. Список пациентов с определенным диагнозом.
4. Загруженность врачей (количество принятых пациентов) за определенный период.

Запросы также могут использоваться в других формах для фильтрации и сортировки данных, для проверки наличия связанных записей перед удалением данных и т.д.

Таким образом, при разработке приложения для работы с базой данных "Регистратура" потребуется создание как минимум следующих запросов:

1. qryPatientSearch - запрос для поиска пациентов по различным критериям.
2. qryDoctorSearch - запрос для поиска врачей по различным критериям.
3. qryTicketSearch - запрос для поиска талонов по различным критериям.
4. qryPatientsByDoctor - запрос для получения списка пациентов определенного врача.
5. qryTicketsByPeriod - запрос для подсчета количества выданных талонов за период.
6. qryPatientsByDiagnosis - запрос для получения списка пациентов с определенным диагнозом.
7. qryDoctorsWorkload - запрос для расчета загруженности врачей за период.

Конечно, это лишь примеры возможных запросов. В зависимости от требований к функциональности приложения и отчетности, список запросов может быть расширен или изменен.

В следующем подразделе мы подробнее рассмотрим процесс создания запросов в Microsoft Access.

3.2 Процесс формирования запросов

В Microsoft Access запросы можно создавать как в режиме конструктора, так и с помощью SQL. Рассмотрим процесс создания запросов разных типов для нашей базы данных "Регистратура".

1. Запрос на выборку (qryPatientsByDoctor):

* На вкладке "Создание" выберите пункт "Конструктор запросов".
* В открывшемся окне добавьте таблицы "Patient", "MedCard" и "Ticket".
* Выберите необходимые поля из каждой таблицы (например, "LastName", "FirstName", "MiddleName", "BirthDate" из таблицы "Patient", "VisitDateTime" из таблицы "Ticket").
* В строке "Условие отбора" для поля "DoctorID" введите нужный идентификатор врача.
* Сохраните запрос под именем "qryPatientsByDoctor".

1. Запрос с параметром (qryTicketsByPeriod):

* Создайте запрос на выборку, добавив таблицу "Ticket".
* Выберите поля "TicketID", "PatientID", "DoctorID", "VisitDateTime".
* В строке "Условие отбора" для поля "VisitDateTime" введите выражение Between [Введите начальную дату:] And [Введите конечную дату:].
* Сохраните запрос под именем "qryTicketsByPeriod".

1. Перекрестный запрос (qryDoctorsWorkload):

* Создайте запрос на выборку, добавив таблицы "Doctor" и "Ticket".
* Добавьте поле "LastName" из таблицы "Doctor" и поле "VisitDateTime" из таблицы "Ticket".
* Перейдите в режим конструктора и нажмите кнопку "Перекрестный запрос" на панели инструментов.
* В строке "Групповая операция" для поля "LastName" выберите значение "Группировка", а для поля "VisitDateTime" - "Счет".
* В строке "Перекрестная таблица" для поля "VisitDateTime" выберите значение "Заголовки строк".
* Сохраните запрос под именем "qryDoctorsWorkload".

1. SQL-запрос (qryPatientsByDiagnosis):

* Создайте новый запрос в режиме SQL.
* Введите следующий SQL-код:

SELECT p.LastName, p.FirstName, p.MiddleName, p.BirthDate, d.DiagnosisName FROM (Patient AS p INNER JOIN MedCard AS mc ON p.PatientID = mc.PatientID) INNER JOIN (MedCard\_Diagnosis AS md INNER JOIN Diagnosis AS d ON md.DiagnosisID = d.DiagnosisID) ON mc.MedCardID = md.MedCardID WHERE d.DiagnosisName = [Введите название диагноза:];

* Сохраните запрос под именем "qryPatientsByDiagnosis".

Этот SQL-запрос выполняет выборку пациентов с определенным диагнозом, используя соединения таблиц "Patient", "MedCard", "MedCard\_Diagnosis" и "Diagnosis". Параметр [Введите название диагноза:] позволяет задавать значение диагноза при выполнении запроса.

Созданные запросы можно использовать в формах для отображения отфильтрованных данных или в отчетах для формирования итоговых данных.

Следующим шагом будет создание отчетов на основе разработанных запросов.

Конечно! Вот русские названия для четырех запросов:

1. Запрос на выборку (qryPatientsByDoctor):
   * Название на русском: "Пациенты выбранного врача" (запросПациентыВрача)
2. Запрос с параметром (qryTicketsByPeriod):
   * Название на русском: "Талоны за период" (запросТалоныЗаПериод)
3. Перекрестный запрос (qryDoctorsWorkload):
   * Название на русском: "Загруженность врачей" (запросЗагруженностьВрачей)
4. SQL-запрос (qryPatientsByDiagnosis):
   * Название на русском: "Пациенты по диагнозу" (запросПациентыПоДиагнозу)

Русские названия запросов:

1. запросПациентыВрача
2. запросТалоныЗаПериод
3. запросЗагруженностьВрачей
4. запросПациентыПоДиагнозу

Эти названия более естественно звучат на русском языке и отражают суть каждого запроса. Вы можете использовать их при описании запросов в вашей курсовой работе.

Например:

1. запросПациентыВрача - запрос на выборку, который выводит список пациентов, относящихся к выбранному врачу.
2. запросТалоныЗаПериод - запрос с параметром, позволяющий получить список талонов, выданных за определенный период времени.
3. запросЗагруженностьВрачей - перекрестный запрос, показывающий количество принятых пациентов для каждого врача.
4. запросПациентыПоДиагнозу - SQL-запрос, выполняющий выборку пациентов с указанным диагнозом.

Надеюсь, эти русские названия помогут сделать вашу курсовую работу более понятной и удобной для чтения.

Давайте перейдем к следующему шагу - созданию отчетов.

3.3 Процесс формирования отчетов

Отчеты в Microsoft Access - это объекты, которые служат для представления данных в удобном для чтения и печати виде. Отчеты позволяют группировать, сортировать и форматировать данные, добавлять заголовки, примечания и итоговые значения.

Для нашей базы данных "Регистратура" предлагаю создать следующие отчеты:

1. Отчет "Список пациентов" (отчетСписокПациентов):

* На вкладке "Создание" выберите пункт "Мастер отчетов".
* Выберите таблицу "Patient" и добавьте поля "PatientID", "LastName", "FirstName", "MiddleName", "BirthDate".
* Выберите уровень группировки (например, по полю "LastName").
* Выберите сортировку данных (например, по полю "FirstName").
* Выберите макет и стиль отчета.
* Введите имя отчета "отчетСписокПациентов" и нажмите "Готово".

1. Отчет "Талоны на сегодня" (отчетТалоныНаСегодня):

* Создайте запрос, выбирающий данные из таблиц "Ticket", "Patient" и "Doctor" для талонов с датой приема, равной текущей дате.
* На основе этого запроса создайте отчет с помощью мастера отчетов.
* Выберите поля "TicketID", "VisitDateTime", "LastName" (пациента), "FirstName" (пациента), "LastName" (врача), "FirstName" (врача).
* Сгруппируйте данные по полю "DoctorID" и отсортируйте по полю "VisitDateTime".
* Введите имя отчета "отчетТалоныНаСегодня".

1. Отчет "Диагнозы пациентов" (отчетДиагнозыПациентов):

* Создайте запрос на основе таблиц "Patient", "MedCard", "MedCard\_Diagnosis" и "Diagnosis", выбирающий ФИО пациента, название диагноза и дату установления диагноза.
* Создайте отчет на основе этого запроса.
* Сгруппируйте данные по полю "PatientID" и добавьте сортировку по полю "DiagnosisDate".
* Введите имя отчета "отчетДиагнозыПациентов".

1. Отчет "Загруженность врачей" (отчетЗагруженностьВрачей):

* Используйте ранее созданный запрос "запросЗагруженностьВрачей".
* Создайте отчет на основе этого запроса.
* Настройте внешний вид отчета, добавив заголовки столбцов и итоговые значения.
* Введите имя отчета "отчетЗагруженностьВрачей".

Созданные отчеты можно открывать, просматривать, редактировать и выводить на печать. Кроме того, отчеты могут быть связаны с формами, чтобы пользователи могли формировать отчеты с различными параметрами (например, за определенный период или по конкретному врачу).

Дальнейшая настройка отчетов (добавление логотипов, изменение цветовой схемы, добавление примечаний и т.д.) может быть выполнена в режиме конструктора отчетов.

После создания всех необходимых отчетов можно перейти к разработке главной кнопочной формы для удобной навигации по приложению.

Конечно, давайте подробнее рассмотрим процесс создания отчета "Талоны на сегодня" (отчетТалоныНаСегодня).

Алгоритм действий:

1. Создание запроса для отбора данных:

* Откройте вкладку "Создание" и выберите "Конструктор запросов".
* Добавьте таблицы "Ticket", "Patient" и "Doctor" в область "Схема данных".
* Дважды щелкните на нужных полях из каждой таблицы, чтобы добавить их в бланк запроса: "TicketID", "VisitDateTime", "LastName" (из таблицы "Patient"), "FirstName" (из таблицы "Patient"), "LastName" (из таблицы "Doctor"), "FirstName" (из таблицы "Doctor").
* В строке "Условие отбора" для поля "VisitDateTime" введите выражение: Date() Это условие отберет только те талоны, у которых дата приема совпадает с текущей датой.
* Сохраните запрос под именем "запросТалоныНаСегодня".

1. Создание отчета на основе запроса:

* На вкладке "Создание" выберите "Мастер отчетов".
* В качестве источника данных выберите созданный запрос "запросТалоныНаСегодня".
* Выберите все поля для отображения в отчете.
* На следующем шаге выберите группировку данных по полю "LastName" (из таблицы "Doctor"). Это сгруппирует талоны по врачам.
* Добавьте уровень сортировки по полю "VisitDateTime", чтобы талоны были отсортированы по времени приема.
* Выберите макет для отчета (например, ступенчатый) и стиль оформления.
* Введите имя отчета "отчетТалоныНаСегодня" и нажмите "Готово".

1. Настройка отчета в режиме конструктора:

* Откройте созданный отчет "отчетТалоныНаСегодня" в режиме конструктора.
* В области "Заголовок отчета" добавьте название отчета, например, "Талоны на [текущая дата]". Для отображения текущей даты используйте выражение =Format(Date(),"dd.mm.yyyy").
* Настройте ширину и расположение полей в областях "Верхний колонтитул группы", "Область данных" и "Нижний колонтитул группы".
* Добавьте итоги (например, подсчет количества талонов) в область "Примечание группы".
* Отформатируйте отчет по своему усмотрению: измените шрифты, цвета, добавьте линии и рамки.
* Сохраните изменения в отчете.

1. Просмотр и печать отчета:

* Откройте отчет "отчетТалоныНаСегодня" в режиме просмотра.
* Проверьте, что данные отображаются корректно, и отчет выглядит так, как задумано.
* При необходимости внесите дополнительные изменения в режиме конструктора.
* Отправьте отчет на печать или экспортируйте в нужном формате (например, PDF).

Этот подробный алгоритм поможет вам создать отчет "Талоны на сегодня". Аналогичным образом можно создавать и другие отчеты, используя соответствующие запросы и настраивая внешний вид и группировку данных в соответствии с требованиями.

3.4 Процесс создания главной кнопочной формы

Главная кнопочная форма - это специальная форма, которая служит отправной точкой для навигации по приложению базы данных. Она содержит кнопки или другие элементы управления, при нажатии на которые открываются различные формы, отчеты или выполняются определенные действия.

Для создания главной кнопочной формы в Microsoft Access выполните следующие шаги:

1. Создание новой формы:

* На вкладке "Создание" выберите "Конструктор форм".
* В открывшемся окне конструктора форм выберите инструмент "Прямоугольник" на панели элементов управления и нарисуйте область для размещения кнопок.
* Добавьте к форме заголовок, например, "Главное меню".

1. Добавление кнопок на форму:

* На панели элементов управления выберите инструмент "Кнопка" и нарисуйте кнопку внутри формы.
* В открывшемся окне "Мастер командных кнопок" выберите категорию действия (например, "Работа с формой") и действие (например, "Открыть форму").
* Выберите форму, которую нужно открыть по нажатию на кнопку (например, "frmPatient").
* Введите подпись для кнопки (например, "Пациенты") и выберите рисунок, если необходимо.
* Повторите шаги 2.1-2.4 для добавления кнопок, открывающих другие формы (например, "frmDoctor", "frmTicket", "frmDiagnosis").

1. Добавление кнопок для отчетов:

* Аналогично добавьте кнопки для открытия отчетов.
* В мастере командных кнопок выберите категорию "Работа с отчетом" и действие "Просмотр отчета".
* Выберите нужный отчет (например, "отчетСписокПациентов") и настройте подпись и рисунок для кнопки.

1. Настройка внешнего вида формы:

* В режиме конструктора измените размеры и расположение кнопок, чтобы они выглядели аккуратно и были удобны для использования.
* Настройте цвета, шрифты и другие элементы оформления формы.

1. Установка главной кнопочной формы в качестве стартовой:

* Откройте вкладку "Файл" и выберите "Параметры".
* В разделе "Текущая база данных" нажмите кнопку "Параметры приложения".
* В поле "Отображаемая форма" выберите созданную главную кнопочную форму (например, "frmMain").
* Нажмите "ОК" и закройте базу данных.

Теперь при открытии базы данных будет автоматически открываться главная кнопочная форма, с которой пользователи смогут легко получить доступ ко всем основным формам и отчетам.

Главная кнопочная форма сделает навигацию по вашему приложению более удобной и понятной для пользователей.

Далее мы рассмотрим, как реализовать разграничение доступа к объектам базы данных для разных групп пользователей.

3.5 Разграничение доступа к БД

Разграничение доступа к базе данных позволяет определить, какие пользователи или группы пользователей имеют право на просмотр, изменение или удаление данных в определенных таблицах, запросах, формах и отчетах. Это важно для обеспечения безопасности и конфиденциальности данных.

В Microsoft Access для разграничения доступа можно использовать встроенные средства управления пользователями и группами. Вот алгоритм настройки разграничения доступа:

1. Включение защиты базы данных:

* Откройте вкладку "Файл" и выберите "Параметры".
* В разделе "Текущая база данных" нажмите кнопку "Шифрование паролем".
* Введите пароль для базы данных и подтвердите его.
* Сохраните и закройте базу данных.

1. Создание групп пользователей:

* Откройте базу данных и перейдите на вкладку "Файл" > "Параметры" > "Центр управления безопасностью" > "Параметры центра управления безопасностью".
* В разделе "Группы пользователей и разрешения" нажмите "Управление группами".
* Создайте новые группы, например, "Администраторы", "Врачи", "Регистраторы".

1. Добавление пользователей в группы:

* В окне "Управление группами" выберите группу и нажмите "Изменить группу".
* Нажмите "Добавить пользователей" и введите имена пользователей для выбранной группы.
* Назначьте пользователям пароли и сохраните изменения.

1. Назначение разрешений для групп:

* В окне "Управление группами" выберите группу и нажмите "Изменить разрешения".
* Назначьте права доступа для каждого объекта базы данных (таблиц, запросов, форм, отчетов).
* Например, для группы "Регистраторы" можно предоставить полный доступ к формам "frmPatient" и "frmTicket", но запретить изменение данных в таблицах "Doctor" и "Diagnosis".
* Сохраните изменения.

1. Тестирование разграничения доступа:

* Закройте базу данных и откройте ее снова.
* Введите имя пользователя и пароль для одной из созданных учетных записей.
* Проверьте, что пользователь имеет доступ только к тем объектам, которые ему разрешены.
* Повторите проверку для других учетных записей.

После настройки разграничения доступа пользователи смогут работать только с теми частями базы данных, которые им необходимы для выполнения своих обязанностей. Это снизит риск случайного или намеренного повреждения данных и обеспечит конфиденциальность информации.

Обратите внимание, что для полноценного разграничения доступа может потребоваться более сложная настройка, например, с использованием VBA или внешних средств защиты. Но для простых случаев встроенных возможностей Microsoft Access обычно достаточно.

На этом раздел "Разработка приложения для работы с базой данных" можно считать завершенным. Пожалуйста, сообщите, если у вас есть какие-либо вопросы или дополнения.

ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы была спроектирована и реализована база данных "Регистратура" для автоматизации работы регистратуры поликлиники. Были успешно решены поставленные задачи:

1. Проведен анализ предметной области, выявлены основные объекты автоматизации (пациенты, врачи, медицинские карты, талоны, диагнозы) и их характеристики.
2. Построена концептуальная модель данных в виде ER-диаграммы, отражающая сущности предметной области и связи между ними.
3. На основе ER-модели разработаны логическая и физическая модели базы данных, обеспечивающие эффективное хранение и обработку данных.
4. База данных реализована средствами СУБД Microsoft Access, созданы таблицы, связи, индексы и ограничения целостности.
5. Разработано приложение для работы с базой данных, включающее формы для ввода и редактирования данных, запросы для поиска и фильтрации информации, отчеты для вывода сводных данных.
6. Реализован механизм разграничения доступа к объектам базы данных для различных категорий пользователей (администраторы, врачи, регистраторы).

В результате внедрения разработанной базы данных и приложения были автоматизированы ключевые процессы работы регистратуры:

* Ведение электронных медицинских карт пациентов, включая информацию о диагнозах и результатах обследований.
* Регистрация пациентов и выдача талонов на прием к врачам с учетом их специализации и графика работы.
* Поиск и фильтрация данных о пациентах, врачах и талонах по различным критериям.
* Формирование отчетов о работе регистратуры, загруженности врачей и диагнозах пациентов.

Использование базы данных "Регистратура" позволяет существенно повысить эффективность работы регистратуры поликлиники за счет:

* Сокращения времени на поиск и обработку информации.
* Уменьшения количества ошибок и неточностей, связанных с ручным вводом и хранением данных.
* Обеспечения быстрого доступа к актуальной информации о пациентах и врачах.
* Автоматизации формирования отчетов и анализа данных.
* Разграничения доступа к данным, что повышает безопасность и конфиденциальность информации.

Таким образом, разработанная база данных и приложение полностью соответствуют потребностям регистратуры поликлиники и могут быть рекомендованы к внедрению. Дальнейшее развитие системы может включать интеграцию с другими информационными системами медицинского учреждения, добавление новых функций (например, онлайн-запись на прием) и расширение возможностей анализа данных.

Опыт проектирования и реализации базы данных "Регистратура" демонстрирует важность применения современных информационных технологий в сфере здравоохранения для повышения качества обслуживания пациентов и эффективности работы медицинских учреждений.

Конечно, я могу предоставить вам данные без кавычек и запятых, чтобы их было удобнее копировать и вставлять в вашу базу данных. Вот обновленные версии данных для каждой таблицы:

Таблица "Patient":

1 Иванов Иван Иванович 1990-05-10 +7(921)123-45-67 1

2 Петрова Елена Сергеевна 1985-09-20 +7(931)234-56-78 2

3 Сидоров Андрей Викторович 1978-02-15 +7(911)345-67-89 3

4 Кузнецова Ольга Александровна 1992-11-03 +7(921)456-78-90 4

5 Смирнов Сергей Иванович 1980-07-28 +7(931)567-89-01 5

6 Новикова Татьяна Михайловна 1995-04-12 +7(911)678-90-12 6

7 Соколов Дмитрий Алексеевич 1983-12-25 +7(921)789-01-23 7

8 Федорова Анна Владимировна 1989-06-07 +7(931)890-12-34 8

9 Попов Александр Николаевич 1975-03-18 +7(911)901-23-45 9

10 Васильева Екатерина Андреевна 1998-08-30 +7(921)012-34-56 10

Таблица "Doctor":

1 Кузьмин Игорь Борисович Терапевт 201

2 Ковалева Ирина Владимировна Кардиолог 205

3 Макаров Сергей Николаевич Невролог 210

4 Зайцева Ольга Ивановна Эндокринолог 215

5 Волков Андрей Петрович Хирург 220

6 Соловьева Елена Александровна Гинеколог 225

7 Егоров Александр Сергеевич Уролог 230

8 Никитина Татьяна Михайловна Офтальмолог 235

9 Лебедев Дмитрий Алексеевич Отоларинголог 240

10 Павлова Наталья Викторовна Дерматолог 245

Таблица "MedCard":

1 1 2020-01-15

2 2 2019-05-20

3 3 2018-11-10

4 4 2021-03-08

5 5 2017-09-25

6 6 2020-06-12

7 7 2019-12-01

8 8 2018-07-18

9 9 2021-02-03

10 10 2017-10-30

Таблица "Ticket":

1 1 1 2023-04-10 09:00:00 201

2 3 2 2023-04-10 10:30:00 205

3 5 3 2023-04-11 14:00:00 210

4 2 1 2023-04-11 15:30:00 201

5 7 4 2023-04-12 11:00:00 215

6 9 5 2023-04-12 13:30:00 220

7 4 6 2023-04-13 10:00:00 225

8 6 7 2023-04-13 12:30:00 230

9 10 8 2023-04-14 09:30:00 235

10 8 9 2023-04-14 11:00:00 240

Таблица "Diagnosis":

1 J00 Острый назофарингит (насморк) Воспаление носоглотки

2 J02 Острый фарингит Воспаление глотки

3 J03 Острый тонзиллит Воспаление миндалин

4 J04 Острый ларингит и трахеит Воспаление гортани и трахеи

5 J06 Острые инфекции верхних дыхательных путей множественной и неуточненной локализации ОРВИ

6 I10 Эссенциальная (первичная) гипертензия Повышенное артериальное давление

7 E11 Сахарный диабет 2 типа Нарушение усвоения глюкозы

8 M54 Дорсалгия Боль в спине

9 K29 Гастрит и дуоденит Воспаление желудка и двенадцатиперстной кишки

10 H10 Конъюнктивит Воспаление конъюнктивы глаза

11 I20 Стенокардия Боль в груди, вызванная ишемией миокарда

12 G43 Мигрень Повторяющиеся приступы головной боли

13 E03 Другие формы гипотиреоза Снижение функции щитовидной железы

14 K80 Желчнокаменная болезнь (холелитиаз) Наличие камней в желчном пузыре

15 N70 Сальпингит и оофорит Воспаление маточных труб и яичников

16 N92 Обильные, частые и нерегулярные менструации Нарушения менструального цикла

17 N30 Цистит Воспаление мочевого пузыря

18 N40 Гиперплазия предстательной железы Доброкачественное увеличение простаты

19 H52 Нарушения рефракции и аккомодации Близорукость, дальнозоркость, астигматизм

20 L23 Аллергический контактный дерматит Воспаление кожи, вызванное аллергеном

21 L70 Угри (акне) Воспаление волосяных фолликулов и сальных желез

Таблица "MedCard\_Diagnosis":

1 1 2020-01-15

1 3 2020-02-20

2 2 2019-05-20

2 6 2019-07-10

3 4 2018-11-10

4 7 2021-03-08

5 5 2017-09-25

6 8 2020-06-12

7 9 2019-12-01

8 10 2018-07-18

Теперь данные представлены в виде простых строк с разделением полей пробелами, что должно облегчить их копирование и вставку в соответствующие таблицы Microsoft Access.