1.1 Описание предметной области

Предметной областью данной курсовой работы является регистратура поликлиники. Основная цель автоматизации процессов в регистратуре - повышение эффективности работы медицинского учреждения, улучшение качества обслуживания пациентов и облегчение труда медицинского персонала.

В рамках предметной области были выделены следующие сущности:

1. Пациент: представляет собой человека, обращающегося за медицинской помощью. Характеристики сущности: фамилия, имя, отчество, дата рождения, телефон.
2. Врач: представляет собой медицинского работника, оказывающего услуги пациентам. Характеристики сущности: фамилия, имя, отчество, специализация, номер кабинета.
3. Медицинская карта: содержит информацию о пациенте и его истории болезни. Характеристики сущности: код пациента, дата заведения, номер полиса ОМС, страховая компания.
4. Талон: представляет собой документ, дающий право на прием у врача в определенное время. Характеристики сущности: код пациента, код врача, дата приема, время приема, номер кабинета.
5. Диагноз: представляет собой заключение врача о состоянии здоровья пациента. Характеристики сущности: код по МКБ-10, название, описание.
6. Диагнозы в медкарте: связующая сущность для реализации связи "многие-ко-многим" между сущностями "Диагноз" и "Медицинская карта". Характеристики сущности: код диагноза, код медкарты.

Основные процессы, требующие автоматизации:

1. Регистрация пациентов и заведение медицинских карт.
2. Запись пациентов на прием к врачу и выдача талонов.
3. Ведение расписания приема врачей.
4. Внесение диагнозов в медицинские карты пациентов.
5. Формирование отчетности по работе регистратуры.

Участниками автоматизируемых процессов являются:

1. Регистратор: сотрудник регистратуры, осуществляющий запись пациентов на прием, заведение медицинских карт и выдачу талонов.
2. Врач: медицинский работник, оказывающий услуги пациентам и вносящий диагнозы в медицинские карты.
3. Пациент: человек, обращающийся за медицинской помощью и получающий талоны на прием.

Связи между сущностями:

1. Пациент - Медицинская карта: один-к-одному. Каждому пациенту соответствует одна медицинская карта.
2. Пациент - Талон: один-ко-многим. Каждый пациент может иметь несколько талонов на прием, но каждый талон связан только с одним пациентом.
3. Врач - Талон: один-ко-многим. Каждый врач может вести прием по нескольким талонам, но каждый талон связан только с одним врачом.
4. 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 используется дополнительная таблица "Диагнозы в медкарте".

1.2. Построение ER-модели

ER-модель (Entity-Relationship Model) - это концептуальная модель данных, используемая для представления структуры базы данных на высоком уровне абстракции. Она служит для описания объектов предметной области (сущностей), их свойств (атрибутов) и взаимосвязей между ними. ER-модель помогает разработчикам баз данных лучше понять структуру данных и облегчает процесс проектирования БД.

Основные элементы ER-модели:

1. Сущность (Entity): объект предметной области, информация о котором должна быть сохранена в базе данных. Изображается в виде прямоугольника с названием сущности внутри.
2. Атрибут (Attribute): свойство сущности, которое характеризует ее экземпляры. Изображается в виде овала, соединенного линией с сущностью, к которой он относится. Каждый атрибут подписывается своим названием.
3. Связь (Relationship): ассоциация между двумя или более сущностями, отражающая их взаимодействие в предметной области. Изображается в виде ромба, соединенного линиями с участвующими сущностями. Каждая связь подписывается своим названием и дополняется обозначением кардинальности (один-к-одному, один-ко-многим, многие-ко-многим).

В ER-модели для регистратуры поликлиники должны быть отражены следующие сущности:

1. Пациент
2. Врач
3. Медицинская карта
4. Талон
5. Диагноз
6. Диагнозы в медкарте

Связи между сущностями:

1. Пациент - Медицинская карта: один-к-одному.
2. Пациент - Талон: один-ко-многим.
3. Врач - Талон: один-ко-многим.
4. Диагноз - Диагнозы в медкарте: один-ко-многим.
5. Медицинская карта - Диагнозы в медкарте: один-ко-многим.

Каждая сущность должна иметь свои атрибуты, перечисленные ранее в описании предметной области.

При построении ER-модели следует учитывать правила нормализации данных, чтобы избежать избыточности и обеспечить целостность данных. Также важно правильно определить кардинальность связей между сущностями, чтобы точно отразить характер их взаимодействия в предметной области.

В итоге, ER-модель регистратуры поликлиники должна наглядно представлять структуру базы данных, облегчая процесс ее дальнейшего проектирования и реализации.

1.3. Логическая модель

Логическая модель данных - это более детальное представление структуры базы данных, основанное на концептуальной (ER) модели. Она описывает объекты БД (таблицы), их атрибуты (столбцы) и связи между объектами. Логическая модель не зависит от конкретной СУБД и служит промежуточным звеном между концептуальным и физическим проектированием базы данных.

Цели логической модели:

1. Детализация структуры БД, полученной на этапе концептуального проектирования.
2. Определение таблиц, их столбцов и связей между таблицами.
3. Обеспечение основы для дальнейшего физического проектирования БД.
4. Документирование структуры БД для разработчиков и пользователей.

Логическая модель для регистратуры поликлиники будет включать следующие таблицы:

1. Пациент (Код пациента, Фамилия, Имя, Отчество, Дата рождения, Телефон)
2. Врач (Код врача, Фамилия, Имя, Отчество, Специализация, Номер кабинета)
3. Медицинская карта (Номер медкарты, Код пациента, Дата заведения, Номер полиса ОМС, Страховая компания)
4. Талон (Номер талона, Код пациента, Код врача, Дата приема, Время приема, Номер кабинета)
5. Диагноз (Код диагноза, Код по МКБ-10, Название, Описание)
6. Диагнозы в медкарте (Код записи, Код диагноза, Номер медкарты)

Связи между таблицами:

1. Пациент - Медицинская карта: один-к-одному. Связь устанавливается по полю "Код пациента".
2. Пациент - Талон: один-ко-многим. Связь устанавливается по полю "Код пациента".
3. Врач - Талон: один-ко-многим. Связь устанавливается по полю "Код врача".
4. Медицинская карта - Диагнозы в медкарте: один-ко-многим. Связь устанавливается по полю "Номер медкарты".
5. Диагноз - Диагнозы в медкарте: один-ко-многим. Связь устанавливается по полю "Код диагноза".

При построении логической модели важно определить первичные и внешние ключи для каждой таблицы:

* Первичный ключ: уникальный идентификатор записи в таблице (обычно это суррогатный ключ, например, "Код пациента").
* Внешний ключ: атрибут, который ссылается на первичный ключ другой таблицы, обеспечивая связь между таблицами.

Логическая модель данных служит основой для дальнейшего физического проектирования базы данных и ее реализации в конкретной СУБД. Она позволяет детально описать структуру БД, обеспечивая ее целостность и непротиворечивость.

1.4. Физическая модель

Физическая модель данных - это представление структуры базы данных, адаптированное под конкретную СУБД (в данном случае, MS Access). Она детально описывает таблицы, их столбцы, типы данных, ограничения и индексы. Физическая модель учитывает особенности реализации БД в выбранной СУБД и служит основой для непосредственного создания базы данных.

Цели физической модели:

1. Адаптация логической модели к особенностям конкретной СУБД.
2. Определение типов данных, ограничений и индексов для таблиц и столбцов.
3. Обеспечение целостности и непротиворечивости данных.
4. Оптимизация производительности БД за счет правильного выбора типов данных и индексов.

Описание таблиц и свойств полей:

1. Таблица "Пациент"

* Код пациента (первичный ключ): счетчик
* Фамилия: короткий текст, обязательное поле
* Имя: короткий текст, обязательное поле
* Отчество: короткий текст
* Дата рождения: дата/время, обязательное поле
* Телефон: короткий текст

1. Таблица "Врач"

* Код врача (первичный ключ): счетчик
* Фамилия: короткий текст, обязательное поле
* Имя: короткий текст, обязательное поле
* Отчество: короткий текст
* Специализация: короткий текст, обязательное поле
* Номер кабинета: числовой

1. Таблица "Медицинская карта"

* Номер медкарты (первичный ключ): счетчик
* Код пациента (внешний ключ): числовой, обязательное поле
* Дата заведения: дата/время, обязательное поле
* Номер полиса ОМС: короткий текст
* Страховая компания: короткий текст

1. Таблица "Талон"

* Номер талона (первичный ключ): счетчик
* Код пациента (внешний ключ): числовой, обязательное поле
* Код врача (внешний ключ): числовой, обязательное поле
* Дата приема: дата/время, обязательное поле
* Время приема: дата/время, обязательное поле
* Номер кабинета: числовой

1. Таблица "Диагноз"

* Код диагноза (первичный ключ): счетчик
* Код по МКБ-10: короткий текст, обязательное поле
* Название: короткий текст, обязательное поле
* Описание: длинный текст

1. Таблица "Диагнозы в медкарте"

* Код записи (первичный ключ): счетчик
* Код диагноза (внешний ключ): числовой, обязательное поле
* Номер медкарты (внешний ключ): числовой, обязательное поле

Связи и обеспечение целостности данных:

1. Связь "Пациент - Медицинская карта": один-к-одному, обеспечивается через внешний ключ "Код пациента" в таблице "Медицинская карта".
2. Связь "Пациент - Талон": один-ко-многим, обеспечивается через внешний ключ "Код пациента" в таблице "Талон".
3. Связь "Врач - Талон": один-ко-многим, обеспечивается через внешний ключ "Код врача" в таблице "Талон".
4. Связь "Медицинская карта - Диагнозы в медкарте": один-ко-многим, обеспечивается через внешний ключ "Номер медкарты" в таблице "Диагнозы в медкарте".
5. Связь "Диагноз - Диагнозы в медкарте": один-ко-многим, обеспечивается через внешний ключ "Код диагноза" в таблице "Диагнозы в медкарте".

Для обеспечения целостности данных следует настроить каскадное обновление и удаление связанных записей при изменении или удалении записей в главных таблицах. Это предотвратит появление "осиротевших" записей в подчиненных таблицах.

Физическая модель данных служит основой для реализации базы данных в MS Access. Она учитывает особенности СУБД и обеспечивает эффективное хранение и обработку данных.

1.5. Нормализация отношений

Нормализация - это процесс организации данных в базе данных с целью устранения избыточности, минимизации аномалий при вставке, обновлении и удалении данных, а также для обеспечения целостности и непротиворечивости данных. Предположим, что все отношения (таблицы) в нашей базе данных находятся в первой нормальной форме (1НФ).

Алгоритм нормализации отношений от 1НФ ко 2НФ и 3НФ:

Шаг 1: Приведение отношений ко второй нормальной форме (2НФ)

1. Определить первичный ключ для каждого отношения.
2. Проверить, зависят ли все неключевые атрибуты от полного составного ключа. Если в отношении есть атрибуты, которые зависят только от части составного ключа, то отношение не находится во 2НФ.
3. Для приведения отношения ко 2НФ необходимо:
   * Выделить атрибуты, зависящие только от части составного ключа, в отдельное отношение.
   * Включить в новое отношение часть составного ключа, от которой зависят выделенные атрибуты, а также сами эти атрибуты.
   * В исходном отношении оставить первичный ключ и атрибуты, зависящие от полного составного ключа.
4. Повторить шаги 2-3 для всех отношений.

Шаг 2: Приведение отношений к третьей нормальной форме (3НФ)

1. Проверить, находятся ли все отношения во 2НФ. Если нет, выполнить Шаг 1.
2. Для каждого отношения проверить, существует ли транзитивная зависимость неключевых атрибутов от первичного ключа. Если такая зависимость есть, то отношение не находится в 3НФ.
3. Для приведения отношения к 3НФ необходимо:
   * Выделить транзитивно зависимые атрибуты в отдельное отношение.
   * Включить в новое отношение атрибут, от которого транзитивно зависят выделенные атрибуты, а также сами эти атрибуты.
   * В исходном отношении оставить первичный ключ и атрибуты, непосредственно зависящие от него.
4. Повторить шаги 2-3 для всех отношений.

После выполнения этих шагов все отношения будут приведены к третьей нормальной форме. Это обеспечит минимизацию избыточности данных, уменьшение аномалий при модификации данных и поддержание целостности базы данных.

Рассмотрим пример нормализации для отношения "Талон":

* 1НФ: Отношение "Талон" уже находится в 1НФ, так как все атрибуты содержат атомарные значения и не имеют повторяющихся групп.
* 2НФ: В отношении "Талон" нет составного первичного ключа, поэтому оно автоматически находится во 2НФ.
* 3НФ: В отношении "Талон" нет транзитивных зависимостей неключевых атрибутов от первичного ключа, поэтому оно уже находится в 3НФ.

Таким образом, отношение "Талон" не требует дальнейшей нормализации. Аналогично следует проанализировать остальные отношения в базе данных и при необходимости привести их к 3НФ, следуя описанному алгоритму.

2.1. Процесс создания таблиц БД в СУБД MS Access

Алгоритм создания таблиц:

1. Открыть MS Access и создать новую базу данных.
2. Перейти на вкладку "Создание" и выбрать "Конструктор таблиц".
3. В окне конструктора таблиц ввести название первого поля, выбрать его тип данных и задать свойства.
4. Повторить шаг 3 для всех полей таблицы.
5. Установить первичный ключ для таблицы.
6. Сохранить таблицу, указав ее имя.
7. Повторить шаги 2-6 для каждой таблицы в базе данных.

Создание таблиц и свойства полей:

1. Таблица "Пациент"

* Код пациента: тип данных - "Счетчик", первичный ключ
* Фамилия: тип данных - "Короткий текст", размер поля - 50, обязательное поле
* Имя: тип данных - "Короткий текст", размер поля - 50, обязательное поле
* Отчество: тип данных - "Короткий текст", размер поля - 50
* Дата рождения: тип данных - "Дата/время", формат - "Краткий формат даты", обязательное поле
* Телефон: тип данных - "Короткий текст", размер поля - 20, маска ввода - "+7 (000) 000-00-00"

1. Таблица "Врач"

* Код врача: тип данных - "Счетчик", первичный ключ
* Фамилия: тип данных - "Короткий текст", размер поля - 50, обязательное поле
* Имя: тип данных - "Короткий текст", размер поля - 50, обязательное поле
* Отчество: тип данных - "Короткий текст", размер поля - 50
* Специализация: тип данных - "Короткий текст", размер поля - 50, обязательное поле
* Номер кабинета: тип данных - "Числовой", размер поля - "Целое"

1. Таблица "Медицинская карта"

* Номер медкарты: тип данных - "Счетчик", первичный ключ
* Код пациента: тип данных - "Числовой", размер поля - "Длинное целое", обязательное поле
* Дата заведения: тип данных - "Дата/время", формат - "Краткий формат даты", обязательное поле
* Номер полиса ОМС: тип данных - "Короткий текст", размер поля - 20
* Страховая компания: тип данных - "Короткий текст", размер поля - 50

1. Таблица "Талон"

* Номер талона: тип данных - "Счетчик", первичный ключ
* Код пациента: тип данных - "Числовой", размер поля - "Длинное целое", обязательное поле
* Код врача: тип данных - "Числовой", размер поля - "Длинное целое", обязательное поле
* Дата приема: тип данных - "Дата/время", формат - "Краткий формат даты", обязательное поле
* Время приема: тип данных - "Дата/время", формат - "Время", обязательное поле
* Номер кабинета: тип данных - "Числовой", размер поля - "Целое"

1. Таблица "Диагноз"

* Код диагноза: тип данных - "Счетчик", первичный ключ
* Код по МКБ-10: тип данных - "Короткий текст", размер поля - 10, обязательное поле
* Название: тип данных - "Короткий текст", размер поля - 100, обязательное поле
* Описание: тип данных - "Длинный текст"

1. Таблица "Диагнозы в медкарте"

* Код записи: тип данных - "Счетчик", первичный ключ
* Код диагноза: тип данных - "Числовой", размер поля - "Длинное целое", обязательное поле
* Номер медкарты: тип данных - "Числовой", размер поля - "Длинное целое", обязательное поле

После создания таблиц следует установить связи между ними, как описано в предыдущих разделах. Это обеспечит целостность данных и упростит работу с базой данных.

2.2. Процесс создания связей между таблицами

Для создания связей между таблицами в MS Access необходимо выполнить следующие шаги:

1. В главном окне базы данных перейти на вкладку "Работа с базами данных" и выбрать "Схема данных".
2. В окне "Добавление таблицы" выбрать все таблицы, участвующие в связях, и нажать "Добавить", затем закрыть окно.
3. Разместить таблицы на схеме данных так, чтобы были видны все необходимые поля.
4. Для создания связи между таблицами нужно перетащить первичный ключ главной таблицы на соответствующее поле (внешний ключ) в подчиненной таблице.
5. В появившемся окне "Изменение связей" настроить параметры связи:
   * Установить флажок "Обеспечение целостности данных".
   * Выбрать "Каскадное обновление связанных полей" и "Каскадное удаление связанных записей" (если необходимо).
   * Нажать "Создать".
6. Повторить шаги 4-5 для всех необходимых связей.
7. Сохранить и закрыть схему данных.

Ключевые поля и индексы:

* Первичные ключи (ПК) выбраны для каждой таблицы на этапе создания таблиц (поля типа "Счетчик").
* Внешние ключи (ВК) - это поля в подчиненных таблицах, которые ссылаются на первичные ключи связанных главных таблиц.
* Индексы создаются автоматически для первичных ключей и внешних ключей, участвующих в связях.

Связи между таблицами, ключевые поля и индексы:

1. Связь "Пациент - Медицинская карта" (один-к-одному)
   * ПК: "Код пациента" (таблица "Пациент")
   * ВК: "Код пациента" (таблица "Медицинская карта")
2. Связь "Пациент - Талон" (один-ко-многим)
   * ПК: "Код пациента" (таблица "Пациент")
   * ВК: "Код пациента" (таблица "Талон")
3. Связь "Врач - Талон" (один-ко-многим)
   * ПК: "Код врача" (таблица "Врач")
   * ВК: "Код врача" (таблица "Талон")
4. Связь "Медицинская карта - Диагнозы в медкарте" (один-ко-многим)
   * ПК: "Номер медкарты" (таблица "Медицинская карта")
   * ВК: "Номер медкарты" (таблица "Диагнозы в медкарте")
5. Связь "Диагноз - Диагнозы в медкарте" (один-ко-многим)
   * ПК: "Код диагноза" (таблица "Диагноз")
   * ВК: "Код диагноза" (таблица "Диагнозы в медкарте")

Правила обеспечения целостности данных:

* Не допускаются несовпадающие значения ключевых полей в связанных таблицах.
* Не допускается удаление записей из главной таблицы, если есть связанные с ними записи в подчиненной таблице (без каскадного удаления).
* Не допускается изменение значения первичного ключа в главной таблице, если на него ссылаются записи в подчиненной таблице (без каскадного обновления).

Каскадное обновление и удаление данных:

* При удалении записи в главной таблице автоматически удаляются все связанные записи в подчиненной таблице.
* При изменении значения первичного ключа в главной таблице автоматически обновляются соответствующие значения внешнего ключа во всех связанных записях подчиненной таблицы.

Эти настройки гарантируют целостность данных и предотвращают появление "осиротевших" записей в базе данных.

3.1. Процесс создания форм

Форма в MS Access - это объект базы данных, который служит для ввода, отображения и редактирования данных в удобном для пользователя виде. Формы позволяют создать интерфейс, облегчающий работу с базой данных, и предоставляют возможность взаимодействия с данными без непосредственного использования таблиц.

Алгоритм создания форм через Мастер форм:

1. В главном окне базы данных перейти на вкладку "Создание" и выбрать "Мастер форм".
2. В диалоговом окне "Создание форм" выбрать таблицу или запрос, на основе которого будет создана форма, и нужные поля. Нажать "Далее".
3. Выбрать внешний вид формы (в один столбец, ленточный и т.д.) и нажать "Далее".
4. Выбрать стиль оформления формы и нажать "Далее".
5. Задать имя формы и выбрать "Открыть форму для просмотра и ввода данных", затем нажать "Готово".

Пример создания форм:

1. Форма "Пациенты"

* Источник данных: таблица "Пациент"
* Поля: Код пациента, Фамилия, Имя, Отчество, Дата рождения, Телефон
* Внешний вид: в один столбец
* Стиль: по выбору пользователя

1. Форма "Врачи"

* Источник данных: таблица "Врач"
* Поля: Код врача, Фамилия, Имя, Отчество, Специализация, Номер кабинета
* Внешний вид: ленточный
* Стиль: по выбору пользователя

1. Форма "Медицинские карты"

* Источник данных: таблица "Медицинская карта"
* Поля: Номер медкарты, Код пациента, Дата заведения, Номер полиса ОМС, Страховая компания
* Внешний вид: в один столбец
* Стиль: по выбору пользователя

1. Форма "Талоны"

* Источник данных: таблица "Талон"
* Поля: Номер талона, Код пациента, Код врача, Дата приема, Время приема, Номер кабинета
* Внешний вид: ленточный
* Стиль: по выбору пользователя

1. Форма "Диагнозы"

* Источник данных: таблица "Диагноз"
* Поля: Код диагноза, Код по МКБ-10, Название, Описание
* Внешний вид: в один столбец
* Стиль: по выбору пользователя

После создания форм можно дополнительно настроить их внешний вид, добавить кнопки и другие элементы управления, а также создать подчиненные формы для связанных таблиц (например, добавить подчиненную форму "Талоны" в форму "Пациенты").

Формы обеспечивают удобный интерфейс для работы с базой данных и позволяют ограничить доступ пользователей к определенным данным, предоставляя им только необходимые для работы поля и функции.

3.2. Процесс создания запросов

Запросы в MS Access - это объекты базы данных, которые позволяют извлекать, фильтровать, сортировать и анализировать данные из одной или нескольких таблиц. Запросы могут быть использованы для выборки данных по определенным критериям, выполнения вычислений, обновления или удаления данных, а также для создания новых таблиц на основе результатов запроса.

Создание запросов с помощью мастера запросов:

1. В главном окне базы данных перейти на вкладку "Создание" и выбрать "Мастер запросов".
2. Выбрать тип запроса (простой запрос, перекрестный запрос, повторяющиеся записи, записи без подчиненных) и нажать "OK".
3. Выбрать таблицы и поля, которые будут использоваться в запросе, и нажать "Далее".
4. При необходимости указать условия отбора записей и нажать "Далее".
5. Выбрать порядок сортировки данных (если требуется) и нажать "Далее".
6. Задать имя запроса и выбрать "Открыть запрос для просмотра данных", затем нажать "Готово".

Пример создания запросов:

1. Запрос "Пациенты по врачу" (уже создан)

* Таблицы: "Талон", "Пациент", "Врач"
* Поля: Фамилия пациента, Имя пациента, Отчество пациента, Фамилия врача, Имя врача, Отчество врача, Специализация врача
* Условие отбора: Фамилия врача, Имя врача, Отчество врача (вводятся пользователем)

1. Запрос "Талоны по дате" (уже создан)

* Таблицы: "Талон", "Пациент", "Врач"
* Поля: Дата приема, Время приема, Фамилия пациента, Имя пациента, Отчество пациента, Фамилия врача, Имя врача, Отчество врача, Номер кабинета
* Условие отбора: Дата приема (вводится пользователем)

1. Запрос "Диагнозы пациента"

* Таблицы: "Пациент", "Медицинская карта", "Диагнозы в медкарте", "Диагноз"
* Поля: Фамилия пациента, Имя пациента, Отчество пациента, Код по МКБ-10, Название диагноза
* Условие отбора: Фамилия пациента, Имя пациента, Отчество пациента (вводятся пользователем)

1. Запрос "Количество талонов по врачам" (SQL)

* SQL-запрос:

sql

* SELECT Врач.Фамилия, Врач.Имя, Врач.Отчество, COUNT(Талон.Код врача) AS [Количество талонов]
* FROM Врач LEFT JOIN Талон ON Врач.[Код врача] = Талон.[Код врача]
* GROUP BY Врач.Фамилия, Врач.Имя, Врач.Отчество;
* Описание: запрос подсчитывает количество талонов для каждого врача, используя левое объединение таблиц "Врач" и "Талон", группировку по полям ФИО врача и агрегатную функцию COUNT.

Запросы позволяют эффективно извлекать и анализировать данные из базы данных, предоставляя пользователям возможность получать нужную информацию без непосредственного взаимодействия с таблицами. SQL-запросы дают дополнительную гибкость и возможности для работы с данными, позволяя выполнять более сложные операции, чем стандартные запросы, создаваемые с помощью мастера.

3.3. Процесс формирования отчетов

Отчеты в MS Access - это объекты базы данных, которые позволяют представить данные из таблиц и запросов в удобном для чтения и печати формате. Отчеты могут включать в себя группировку, сортировку, вычисления и форматирование данных. Они особенно полезны для создания структурированных документов, таких как списки, сводки или аналитические отчеты.

Для создания отчетов в MS Access можно использовать Мастер отчетов, который предоставляет пошаговый процесс создания отчета на основе выбранных таблиц и запросов.

Алгоритм создания отчета с помощью Мастера отчетов:

1. В главном окне базы данных перейти на вкладку "Создание" и выбрать "Мастер отчетов".
2. Выбрать таблицу или запрос, на основе которого будет создан отчет, и нажать "Далее".
3. Выбрать поля, которые будут включены в отчет, и нажать "Далее".
4. При необходимости добавить уровни группировки и нажать "Далее".
5. Выбрать порядок сортировки данных и нажать "Далее".
6. Выбрать макет отчета (ступенчатый, блок, структура) и ориентацию страницы, затем нажать "Далее".
7. Выбрать стиль отчета и нажать "Далее".
8. Задать имя отчета и выбрать "Просмотреть отчет", затем нажать "Готово".

Пример создания отчетов:

1. Отчет "Список пациентов":

* Источник данных: таблица "Пациент"
* Поля: Код пациента, Фамилия, Имя, Отчество, Дата рождения, Телефон
* Группировка: нет
* Сортировка: по полю "Фамилия" по возрастанию
* Макет: ступенчатый
* Стиль: по выбору пользователя

1. Отчет "Талоны по врачам":

* Источник данных: запрос "Талоны по врачам" (предполагается, что такой запрос уже создан и включает необходимые поля из таблиц "Талон" и "Врач")
* Поля: Фамилия врача, Имя врача, Отчество врача, Дата приема, Время приема, Номер кабинета
* Группировка: по полям "Фамилия врача", "Имя врача", "Отчество врача"
* Сортировка: по полю "Дата приема" по возрастанию
* Макет: блок
* Стиль: по выбору пользователя

После создания отчетов можно дополнительно настроить их внешний вид в режиме конструктора отчетов, добавить заголовки, примечания, вычисляемые поля и другие элементы оформления.

Отчеты позволяют наглядно представить данные из базы данных, облегчая их восприятие и анализ. Они являются важным инструментом для создания информативных и профессионально оформленных документов на основе данных, хранящихся в базе данных.