|  |
| --- |
|  |

|  |
| --- |
| **Лабораторная работа  «Реализация реляционной базы данных в ACCESS»** |
| *Лабораторный практикум по базам данных* |
|  |
| 1. Создание таблиц 2. Создание схемы базы данных 3. Ввод исходных данных 4. Поддержка целостности данных 5. Анализ данных в режиме таблиц |
|  |
| **ЧувГУ, каф. Вычислительной техники** |
| **23.02.2021** |
|  |

*Этап физического проектирования начинается с создания физической структуры реляционной базы данных. Разработанная на этапе логического проектирования база данных переносится в среду конкретной СУБД – в данном случае ACCESS, что требует учета всех особенностей разработки на данной платформе.*

## Что надо знать

1. **Установка и настройка MS ACCESS (🕮** Запуск и параметры/Установка и настройка**)**
2. **Создание таблиц**
   1. Методы создания базы данных ACCESS
   2. Основные режимы работы с таблицами
   3. Способы создания таблиц
   4. Работа с полями таблицы
   5. Форматы отображения данных
   6. Маска ввода
   7. Создание и использование индексов
   8. Использование подстановок
   9. Использование списков значений
   10. Мастер анализа таблиц
   11. Улучшение производительности MS ACCESS
3. **Создание схемы базы данных**
   1. Определение связей между таблицами.
   2. Поддержание целостности данных между таблицами
   3. Задание или изменение типа объединения. Изменение типа объединения для связи в окне схемы данных не влияет на саму связь; оно устанавливает тип объединения, который будет использоваться по умолчанию при создании запросов, основанных на связанных таблицах в базе данных Microsoft Access. Предложенный по умолчанию тип объединения можно будет изменить при создании запроса.
4. **Ввод данных**
   1. Навигация по таблице
   2. Работа с записями
      1. Вставка в поле значения из предыдущей записи
   3. Контроль и проверка введенных данных
5. **Анализ данных в режиме таблицы**
   1. Форматирование таблицы:
      1. опция отображения столбцов для скрытия ключевых полей (*Отобразить столбцы…* из контекстного меню)
      2. опция «режим таблицы» (*Формат/Режим таблицы* – окно *Формат таблицы*)
      3. закрепление и освобождение столбцов в режиме таблицы
   2. Поиск и замена данных (**🕮** Работа с данными/Поиск, сортировка и группировка данных**)**
      1. по номеру нужной записи в поле номера записи (в режиме таблице нажмите F5).
      2. с помощью диалогового окна *Поиск*
      3. с помощью фильтра
      4. с помощью запроса (отбор данных)
      5. запрос на обновление (замена данных)
   3. Применение фильтра в таблице.
      1. Фильтр по выделенному (отбор записей с помощью выбора значений в форме или таблице).
      2. Обычный фильтр (отбор записей путем ввода в пустую таблицу или форму)
      3. Фильтр для **…** (отбор записей путем ввода условия в поле формы или таблицы)

Используйте клавишу Tab вместо Enter для ввода дополнительных условий

* + 1. Расширенный фильтр

Значение свойства **Фильтр (Filter)** задается в [окне свойств](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office10\1049\acmain10.chm::/html/achowSetFormReportControlProperties_EU.htm) формы или таблицы, в [макросе](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office10\1049\acmain10.chm::/html/achowSettingPropsUsingMacros_EU.htm) или в программе [Microsoft Visual Basic](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office10\1049\acmain10.chm::/html/achowSettingPropsUsingCode_EU.htm).

* 1. Использование сортировки

Значение свойства **Порядок сортировки (OrderBy)** задается в [окне свойств](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office10\1049\acmain10.chm::/html/achowSetFormReportControlProperties_EU.htm) объекта, в [макросе](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office10\1049\acmain10.chm::/html/achowSettingPropsUsingMacros_EU.htm) или программе [Microsoft Visual Basic](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office10\1049\acmain10.chm::/html/achowSettingPropsUsingCode_EU.htm).

* 1. Использование подчиненных таблиц для просмотра связанных данных. Microsoft Access автоматически создает подтаблицу в таблице, имеющей [отношение «один-к-одному»](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office10\1049\acmain10.chm::/html/aclrgSubdatasheetsWhatHow.htm##) или являющейся стороной «один» в отношении «один-ко-многим», если *свойство* таблицы **Имя подтаблицы** (SubdatasheetName) имеет значение **Авто**. Отношение определяется по совпадению полей [первичного ключа](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office10\1049\acmain10.chm::/html/aclrgSubdatasheetsWhatHow.htm##) и [внешнего ключа](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office10\1049\acmain10.chm::/html/aclrgSubdatasheetsWhatHow.htm##) в связанных таблицах. Тем не менее, подтаблицу можно добавить в любую таблицу, запрос или форму (в режиме таблицы в меню **Вставка** выберите команду **Подтаблица**).

***Примечание***. Обратите внимание, что в качестве подтаблицы можно использовать как таблицы, так и запросы. Кроме того, допускается многоуровневое вложение подтаблиц. При вставке или изменении подтаблицы можно создать самообъединение.

## Основное задание

**Изображение выглядит как текст

Автоматически созданное описание**Перед началом работы рекомендуется установить основные параметры по работе с Access в окне «Параметры Access», вызываемом из контекстного меню *Настройка ленты*. Создаваемые таблицы будут оформлены в соответствии с предварительными установками.

1. **Создайте таблицы** 
   1. При создании таблиц кроме типа данных укажите также характеристики полей. Для текстовых полей и полей даты задайте *маску ввода*, введите *значение по умолчанию*. Для ускорения поиска создайте индексы.

Если свойство Формат (Format) управляет тем, как данные отображаются в поле таблицы, то свойство Маска ввода (Input Mask) позволяет контролировать ввод данных в таблицу. И не только контролировать, но и упрощать процесс ввода. Маску используют в том случае, когда вводимые данные должны содержать определенные символы в некоторых позициях вводимой строки. Самым простым и ярким примером таких данных являются номера телефонов. Маска ввода должна обеспечить возможность вводить только цифры номера, а остальные символы (скобки вокруг кода города, дефис между цифрами номера) будут добавляться автоматически.

 Индексы могут замедлить выполнение некоторых [запросов на изменение](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office10\1049\acmain10.chm::/html/acconAboutIndexesInAccessDatabase.htm##), например, [запросов на добавление](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office10\1049\acmain10.chm::/html/acconAboutIndexesInAccessDatabase.htm##), при выполнении которых требуется обновление индексов многих полей.

* 1. Создайте поля со списком подстановок или списком значений. Активно применяйте подстановочные поля. Поле подстановок может получать список значений из таблицы или запроса, а также из постоянного набора значений, который задается пользователем. Создание в таблице поля со списком подстановок или списком значений возможно как в *режиме конструктора*, так и в *режиме таблиц*.

 При добавлении в форму поля подстановок в форме автоматически создается поле со списком или список. Однако, если после добавления в форму поля подстановок или списка значений их описание будет изменено, эти изменения не будут отражены в форме. В таком случае нужно удалить поле из формы, а затем добавить его заново.

Допускается создание в таблице поля подстановок, в котором выводятся значения из других полей этой же таблицы. Например, в таблице «Сотрудники» для каждого сотрудника можно в поле подстановки «Подчиняется» вывести имя и фамилию его начальника, которые берутся из полей «Имя» и «Фамилия» этой же таблицы, определяемых соответствующим кодом начальника из поля «КодСотрудника».

* 1. Воспользуйтесь мастером анализа таблиц для проверки разработанных таблиц на соответствие требований нормализации.

1. **Создайте схему базы данных (🕮 С**оздание БД и объектов/Работа с таблицами/Настройка связей и параметров целостности данных)
   1. Установите связи между созданными таблицами

 Таблицу можно связать саму с собой. Это полезно, когда нужно создать поле подстановок со значениями из этой же таблицы. Например, в таблице «Сотрудники» можно определить связь между полями «КодСотрудника» и «Подчиняется» и отображать в поле «Подчиняется» данные о сотруднике, из связанного поля «КодСотрудника». Чтобы связать таблицу саму с собой, в схему данных добавьте ее дважды.

Допускается определение связей не только для таблиц, но и для запросов. Однако для запросов условия [целостности данных](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office10\1049\acmain10.chm::/html/achowDefineRelationshipsBetweenTablesS.htm##) не налагаются.

Таблицы могут соединяться между собой и по нескольким полям. Для этого их следует добавить в соответствующие поля таблицы «Изменение связи»

* 1. Установите режим поддержки целостности данных между таблицами

 В поле [внешнего ключа](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office10\1049\acmain10.chm::/html/acconDetermineRelationshipsForDatabaseS.htm##) связанной таблицы невозможно ввести значение, не содержащееся в ключевом поле главной таблицы. Однако в поле внешнего ключа возможен ввод значений [Null](mk:@MSITStore:C:\Program%20Files\Microsoft%20Office\Office10\1049\acmain10.chm::/html/acconDetermineRelationshipsForDatabaseS.htm##), показывающих, что записи не являются связанными. Например, нельзя сохранить запись, регистрирующую заказ, сделанный несуществующим клиентом, но можно создать запись для заказа, который пока не отнесен ни к одному из клиентов, если ввести значение Null в поле «КодКлиента».

* 1. Задайте или измените тип объединения (внутреннее соединение, внешнее соединение, рекурсивное соединение). Изменение типа объединения для связи в окне схемы данных не влияет на саму связь; оно устанавливает тип объединения, который будет использоваться по умолчанию при создании запросов, основанных на связанных таблицах в базе данных Microsoft Access. Предложенный по умолчанию тип объединения можно будет изменить при создании запроса.
  2. Изучите в режиме *конструктора таблиц* свойства таблицы, вызвав одноименное окно свойств из панели инструментов (символ ”рука”).

Чтобы развернуть ячейку свойства в окне свойств, можно щелкнуть ее правой кнопкой и выбрать команду **Масштаб/Область ввода** (или нажмите клавишу SHIFT+F2).

1. **Введите в таблицы исходные данные**
   1. Создайте правила проверки поля, задающие условия на значение поля. Чтобы добавить правила проверки в таблицу, откройте нужную таблицу и используйте команды на вкладке **Поля** ленты.

* Создайте правила проверки поля;
* Создайте сообщения для отображения при вводе недопустимых данных.
  1. Добавьте условия на значение записи. Это условие позволяет сравнить значения нескольких полей сразу. Такая проверка предназначена для поддержки семантической целостности данных. Например, дата окончания института не может быть меньше даты окончания школы. Условное выражение, определяющее семантическую целостность, включает в себя ссылки на поля одной таблицы.
* Откройте таблицу, в которой требуется выполнить проверку записей.
* На вкладке **Поля** в группе **Проверка поля** нажмите кнопку *Проверка* и выберите пункт *Правило проверки поля*.
* Создайте правило проверки с помощью построителя выражений.

Значения свойств **Условие на значение (ValidationRule)** и **Сообщение об ошибке (ValidationText)** задаются:

* в разделе свойств поля в режиме конструктора таблицы (условие на значение поля);
* в окне свойств таблицы, вызываемом командой **Свойства** в меню **Вид** в режиме конструктора таблицы (условие на значение записи);
* в окне свойств элемента управления в форме;
* в макросе или в программе Microsoft Visual Basic. В программе Microsoft Visual Basic значения данных свойств задаются с помощью строковых выражений.
  1. Введите *исходные* данные из первичных документов.
  2. В режиме *конструктора таблиц* проверьте введенные данные на семантическую целостность *(Правка /Проверка условий).*

1. **Анализ данных в режиме таблиц** (**🕮** Работа с данными / Поиск, сортировка и группировка данных**)** 
   1. Изучите возможности поиска и замены данных, используя все возможные методы
   2. Изучите применение различных типов фильтрации в таблице.
   3. Использование сортировки.
   4. Проведите анализ данных с использованием подтаблиц. Изучите возможностипереключения подтаблиц. (**🕮** Работа с данными / Просмотр данных / Работа с таблицами / Работа с подтаблицами**).**

## Содержание отчета

 Состав разработанной базы данных, включающий перечень разработанных таблиц с атрибутами. Первичный ключ – подчеркнуть.

***Состав базы данных***

|  |  |  |
| --- | --- | --- |
| **№** | **Название таблицы** | **Названия полей** |
| 1 | СЛУЖАЩИЙ | #Сл, ФИО , ДатаР, Фото |
| 2 | СПЕКТАКЛЬ | #Сп, Назв, Жанр, Кмпз |
| 3 | РЕПЕРТУАР | # Р, Дата, Время, #Сп |
| 4 | АКТЕР | #Актер, Разряд, Звание |

 Для каждой созданной таблицы составьте описание характеристик атрибутов по предлагаемой ниже форме.

***Описание характеристик атрибутов***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Имя атрибута** | **А1** | **А2** | **А3** | **А4** | **А5** | **А6** | **А7** |
| 1 | #Сл | Сч | дц |  |  |  | да |  |
| 2 | ФИО | Текст | 20 |  |  |  | да | д/н |
| 3 | ДатаР | Д/В |  | Кфд | Дата рождения |  | да |  |
| 4 | Телеф | Текст | 8 | &&-&&-&& | Телефон |  |  |  |
| 5 | Фото | OLE |  |  |  |  |  |  |

где

* Тип данных (А1).
* Размер поля (А2).
* Формат вывода значений данного поля (А3).
* Подпись поля в форме (А4).
* Значение по умолчанию (А5).
* Обязательное поле (А6).
* Индексированное поле (А7).

 Сформулируйте ограничения (правила валидации) для полей и записей каждой таблицы.

***Правила*** ***валидации*** (ValidationRule)

***Для полей***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Таблица** | **Поле** | **Условие на значение** | **Сообщение об ошибке** |
| 1 | Репертуар | Дата | <Date() |  |
| 2 | Служащий | Дата рождения | >=#01.01.1930# And <#01.01.2020# | Ошибка! Очень молод или очень стар! |
|  |  |  |  |  |

***Для записей***

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Таблица** | **Условие на значение** | **Сообщение об ошибке** |
| 1 | Актер | Not ([Разряд]<14 And Not ([Звание]="")) | Разряд актера не соответствует его званию |
|  |  |  |  |
|  |  |  |  |

 По предлагаемой ниже форме создайте описание подстановок.

***Описание*** ***подст ановок***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Таблица | Исходное поле | Поле подстановки | Тип элемента управления | Тип источника строк | Источник строк |
| Спектакль | Жанр | Жанр | Список | Список значений | **Балет; Опера; Оперетта** |
| Роль | #Сл | ФИО актера | Поле со списком | Сохраненный запрос | **з\_Служащий** |
| Роль | #Сп | Спектакль | Поле со списком | Инструкция SQL | **SELECT Спектакль.[#Сп], Спектакль.Название FROM Спектакль;** |
| **и т.д.** |  |  |  |  |  |

 Описание связей между таблицами. Приведите схему разработанной базы данных (при открытом окне схемы *Файл/Печать схемы данных*).



## Контрольные вопросы

1. Назовите три метода создания БД в Microsoft Access

Назовите основные режимы работы с таблицами. Как перейти из одного режим в другой?

1. Какие типы объединения записей таблиц (параметры объединения) существуют? В чем их отличие?
2. Какие два инструмента помогают усовершенствовать структуру базы данных?
3. Какое отношение в Microsoft Access называется неопределенным? Как оно отображается в схеме базы данных?
4. Как поддерживается целостность данных в таблицах в Microsoft Access?
5. На примере разрабатываемой БД продемонстрируйте каскадное удаление и обновление.
6. Можно ли установить связи между несколькими полями?
7. Как создаются индексы для одного поля, составной индекс?
8. Как установить формат отображения для данных в поле или элементе управления? Что такое собственный формат?
9. Как создаются в таблице поля со списком подстановок или списком значений в *режиме таблиц*?
10. Что происходит при импортировании и связывании данных?
11. Как сохранить фильтр, чтобы при повторном открытии таблицы или формы можно снова применить сохраненный фильтр?
12. Перечислите основные свойства таблицы и объясните их назначение.
13. Как распечатать схему базы данных?
14. Как автоматически устранить ошибки, вызванные переименованием форм, отчетов, таблиц, запросов, полей, текстовых полей и других элементов управления в базах данных Microsoft Access?
15. Как предотвратить ввод в таблицу повторяющихся значений?
16. Как закрепить один или несколько столбцов (полей) объекта в режиме таблицы так, чтобы они располагались в левой части таблицы и отображались на экране независимо от прокрутки?
17. В чем состоит различие между значением *Null* и *пустой строкой*?
18. Для типа данных *Текстовый* объясните назначение таки свойств поля как *Пустые строки, Сжатие Юникод.*
19. Как отсортировать данные по нескольким полям?
20. Какие способы проверки условий на значения данных в программах вы знаете?
21. Объясните назначение пункта меню *Вид|Зависимости объектов* (при выделенном объекте). Зависимости объектов также можно вызвать из области задач.
22. Как используется автозамена имен? Назовите три режима работы автозамены.
23. В чем различие межу маской ввода и форматом поля?
24. Как перенести данные из предыдущей записи в следующую?
25. Как можно скрыть таблицу, чтобы она не отображалась в окне базы данных?
26. Знаете ли вы, что для того, чтобы при открытии файла Access не срабатывали макросы и не открывалась форма по умолчанию, необходимо …?

## Дополнительное задание

1. Автоматизировать выполнение операций импорта путем создания макроса или создания процедуры Microsoft Visual Basic для приложений. Это бывает полезно при импорте данных по определенному расписанию или при наличии нестандартных или сложных требований к импорту данных. Работа с данными /Работа с внешними данными/ Автоматизация импорта, экспорта и связывания данных
2. Разработайте спецификацию импорта/экспорта при неоднократном выполнении импорта в одну и ту же таблицу, экспорта в один и тот же файл или для автоматизации процесса импорта или экспорта.

# ПРИЛОЖЕНИЕ

**Содержание**

* **[Поля, которые должны быть в таблицах](#_Поля,_которые_должны_быть в таблица)**
* **[Повышение быстродействия Microsoft Access](#_Повышение_быстродействия_Microsoft__1)**

**Новиков Сергей**

## Поля, которые должны быть в таблицах

Каждая таблица должна иметь поле первичного ключа для уникальной идентификации каждой записи и это поле должно быть первым в таблице. Обычно это поле типа "Счет­чик" (длинное цело индексированное без повторений с авто-инкрементом). Я рекомендую использовать для поля уникального ключа именно такой тип данных (если это возможно).

Так же весьма полезно, если в таблице присутствуют следующие поля:

**ID\_StateRecord** — это целое поле, в котором по умолчанию находится 0. Определяет "состоя­ние" записи. Если в этом поле отрицательное значение, то запись считается "уда­ленной". **-1** значит "Удалено", а **-2** - "Удалено вместе с "родительской" записью". Процедура каскадного "удаления" записей будет рассмотрена ниже.

* **DateCreation** - поле типа *Дата*, в котором по умолчанию содержится дата создания записи. Удобно иметь это поле, чтобы знать, когда запись была создана.
* **UserCreation** - текстовое поле, в которое необходимо заносить имя пользователя, создав­шего запись. Удобно иметь это поле, чтобы знать, кем запись была создана. Заполнять это поле придется программно.

**Расположений полей в таблицах**

Я советую располагать поля в таблице следующим образом: первым полем должно быть **ключевое поле**, вторым - поле **ID\_StateRecord**, далее нужно располагать поля, участвую­щие в связях с другими таблицами, потом остальные поля, предпоследним полем должно быть поле **DateCreation**, и последним - **UserCreation**.

Вообще-то без разницы, в каком порядке в таблице находятся поля, но вышеперечислен­ный порядок лично для меня является самым удобным.

**"Нулевая" запись**

Каждая таблица по возможности должна иметь запись, в которой значение, содержащееся в ключевом поле (далее будем считать, что ключевое поле имеет типа "Счетчик") равно нулю. Попробую объяснить, для чего это может быть полезно. Я, например, отношусь к "нулевой" записи как к записи, несущий следующий смысл: "Не введено". Для таблицы как таковой это особо не важно, но если эта таблица участвует в связях "один-ко-многим" с другими таблицами со стороны "один", то в связных таблицах если в поле связи стоит 0, то это значит: "Не введено".

Например, если есть талицы tbl\_Cities (города) и tbl\_Countries (страны) и таблица tbl\_Countries имеет ключевое поле ID\_Country и "нулевую" запись, а эти таблицы связаны по ключевому полю ID\_Country (каждый город находиться в определенной стране), то при вводе нового города это поле в этой таблице по умолчанию должно быть равно нулю, т.е. ссылаться на "нулевую" запись в таблице стран. Это будет означать, что для этого города еще не определена страна. При этом не будет выскакивать ошибки типа "Добавление не­возможно, так как отсутствует связная запись"

Реализация этой идеи с полем типа "Счетчик" стала возможна благодаря интересной осо­бенности поля такого типа: при программном создании записи значение такого поля можно задавать самостоятельно.

**"ND"-запрос**

"ND"-запрос — это такой запрос, который отображает все неудаленные записи в таблице. ("ND" значит "Not Deleted", то есть "неудаленные"). В самой простой ситуации это SELECT по всей таблице без каких либо условий. Если таблица имеет поле ID\_StateRecord, то SQL для "ND"-запроса выглядит так: **SELECT \* FROM tbl\_MyTable WHERE ID\_StateRecord >=0.**  При использовании функции каскадного изменения состояния записей (псевдоудаления, когда при необходимости удаления запись не удаляется физиче­ски, а "помечается" как удаленная) этот запрос будет возвращать все не удаленные записи.

При наличии для таблицы "ND"-запроса ВСЯ последующая работа с данными этой табли­цы должна происходить ТОЛЬКО через этот запрос.

Каждая таблица должна иметь свой "ND"-запрос с префиксом **qryND\_**

Я вообще советую работать с таблицами ТОЛЬКО через запросы, и "ND"-запрос для этого подходит идеально.

**"List"-запрос**

"List"-запрос — это запрос, построенный на основе одного или нескольких запросов (обычно это "ND"-запросы), содержащий в себе необходимые поля для создания "List"-формы. Что из себя представляет "List"-форма будет рассказано ниже.

**"Локальная" таблица**

Что такое "локальная" таблица я довольно подробно описывал в статье «Использование ло­кальной таблицы в качестве хранилища значений элементов управления формы для усло­вий отбора в запросах, или как избавиться от ссылок на формы путем написания несколь­ких универсальных функций», так что скажу только, что в Мастере Универсального Интер­фейса "локальная" таблица и функции работы с такой таблицей претерпели некоторые из­менения. "Локальная" таблица необходима для работы "Get", "Edit" и "List"-форм. И не только. Поэтому создать для своей базы "локальную" талицу я советую в первую очередь.

**"Журнальные" таблицы**

В Access без использования архитектуры "клинт-сервер" невозможно "стандарт­ными" методами вести журнал изменения записей в таблицах. А хотелось бы. Для чего - и так понятно. Имея такой журнал, где "все ходы записаны" можно узнать, например, кто когда "срукосуйничал". У меня было не один раз были случаи чудесного излечения амне­зии у пользователей, которые кричали мне что мол "я никогда в жизни не мог эти данные сюда внести". Когда говоришь ему, что "сделал ты это такого-то числа во столько-то часов столько-то минут столько-то секунд из-под такого-то АРМа в такой-то форме, сидя за та­кой-то машиной и изменил ты такие-то поля с таких-то на такие-то значения" - сразу ам­незия проходит. Можно сделать "откат" определенных изменений. И т.д. и т.п.

Для начала нам нужна таблица, в которых будет храниться вышеописанная полезная ин­формация. Таких таблиц нам нужно 4: в одной будет вестись журнал событий (в какой таблице в какой записи кем-когда-откуда были произведены измене­ния/добавления/удаления), в другой - собственно список полей и значений этих полей (что было - что стало) для каждой записи первой таблицы. И еще две таблички будут содер­жать список событий, которые могут произойти в таблице (Создание новой записи, Изме­нение записи, Удаление записи). Рассмотрим первые 2 таблицы.

|  |  |  |
| --- | --- | --- |
| **tbl\_Journal** | | |
| **Таблица журнала событий** | | |
| **Поле** | **Тип поля** | **Описание** |
| **ID\_Journal** | Счетчик, ключ | Идентификатор записи |
| **ID\_StateRecord** | Long | см. выше про поле ID\_StateRecord |
| **ParentIDJournal** | Long | Идентификатор "родительской" записи в этой же таблице |
| **ID\_JournalType** | Long | Тип события (1 -создание, 2- удаления, 3 -изменение) |
| **TableName** | Text | Имя таблицы, для которой произошло событие |
| **KeyName** | Text | Имя ключевого поля таблицы **TableName** |
| **KeyValue** | Long | Значение поля **KeyName** таблицы **TableName**. (Поскольку поле **KeyName** являеться ключевым для таблицы **TableName**, то значение поля **KeyValue** однозначно определяет запись, для которой произошло событие). |
| **MachineName** | Text | Имя компьютера, на котором произошло событие (чтобы знать с какого компа были совершены действия). |
| **FormName** | Text | Имя формы, в которой произошло событие. |
| **AppName** | Text | Имя и путь к файлу БД, в котором произошло событие. |
| **DateCreation** | Date | Когда создана запись. (см. выше про поле DateCreation) |
| **UserCreation** | Text | Кем создана запись (см. выше про поле UserCreation) |

Например имея в таблице **tbl\_Journal** запись, где, например:   
**ID\_Journal =** 123  
**ID\_StateRecord** = 0  
**ParentIDJournal** = 0  
**ID\_JournalType** = 3  
**TableName** = "tbl\_MyTable"  
**KeyName =** "ID\_MyTable"  
**KeyValue** = 593  
**MachineName** = "NINA"  
**FormName** = "frmEdit\_MyTable"  
**AppName =** "C:\MyBase\MyBase97.mdb"  
**DateCreation** = "12.09.2002 17:43:32"  
**UserCreation** = "Vova"  
           ...а в таблице **tbl\_JournalEvents** запись:  
**ID\_JournalEvent** = 789  
**ID\_StateRecord** = 0  
**ID\_Journal** = 123  
**ID\_JournalEventType** = 3  
**ParentIDJournalEvent** = 0  
**JournalEventMemo1** = "MyField"  
**JournalEventMemo2** = "Кое-что"  
**JournalEventMemo3** = "Что-нибудь"  
**JournalEventMemo4** = ""  
**DateCreation** = "12.09.2012 17:43:32"  
**UserCreation** = "Vova"

**...** можно сказать, что **12-го сентября 2012 года в 17:43:32** пользователь **Vova**, работая за машиной **NINA** с базой **C:\MyBase\MyBase97.mdb** в форме **frmEdit\_MyTable** изменил в таблице **tbl\_MyTable** (у которой ключевое поле **ID\_MyTable**) в записи, у которой **ID\_MyTable = 593** значение поля **MyField** с **"Кое-что"** на **"Что-нибудь"**. Такая информация иногда бывает ой как нужна!

Теперь давайте разберемся, что нужно для того, чтобы такая полезная информация в этих таблицах появлялась. Для этого сначала нам необходимо создать "журнальные" таблицы и функции для работы с "журнальными" таблицами. Это умеет делать Мастер. При этом у вас в базе появится 4 вышеописанных таблицы, модуль **mdl\_Journal** с необходимым набором функций и 2 модуля класса: **class\_Database** и **class\_Recordset**, которые имеют тут же функциональность, что и объекты **Database** и **Recordset**, но "умеют" отслеживать изменения в таблицах.

Для того, чтобы в какой-либо форме происходило отслеживание изменений в ее модуль надо добавить блок кода из модуля **mdl\_Journal** (там он находится в начале модуля и закомментирован. Комментарии, понятно дело, в модуле формы надо убрать). Этот код универсален и будет работать для любой формы независимо от ее источника данных.

С формами разобрались. Теперь надо разобраться с программным изменением записей. Для этого нам потребуеться класс **class\_Recordset**. Работать с ним надо следующим образом: создавать экземпляр класса, инициализировать его и вызывать методы **AddNew**, **Edit**, **Update, MoveFirst** и т.п. уже для этого класса, а не для рекордсета, которым он был инициализирован. Вот как должен выглядеть код **ДО** и **ПОСЛЕ** внесения необходимых изменений:

|  |
| --- |
| **ДО** |
| Dim rst As Recordset  Set rst = CurrentDb.OpenRecordset("qryND\_MyTable", dbOpenDynaset)  rst.AddNew      rst!MyField = "Кое-что"      rst!MyFieldAmount = 0      rst!UserCreation = AsCurrentUser()  rst.Update |
| **ПОСЛЕ** |
| Dim rst As Recordset  Dim clrst As New class\_Recordset  Set rst = CurrentDb.OpenRecordset("qryND\_MyTable", dbOpenDynaset)  clrst.Initialize rst  clrst.AddNew      rst!MyField = "Кое-что"      rst!MyFieldAmount = 0      rst!UserCreation = AsCurrentUser()  clrst.Update |

Вот пока и все. Сейчас работаю над процедурой автоматического "отката" и над тем, чтобы класс **class\_Database** использовать для отвлеживания изменений при программном выполнении запросов на обновление/добавление/удаление. Чтобы вместо **CurrentDb.Execute "DELETE \* FROM tbl\_MyTable"** можно было написать **clDb.Execute "DELETE \* FROM tbl\_MyTable"** - и это уже бы поддерживало отслеживание изменений.

[](#_Основное_задание)

## Повышение быстродействия Microsoft Access

**Оптимизация работы с таблицами**

Существует несколько возможностей оптимизировать работу с таблицами. Помимо перечисленных ниже советов, можно использовать анализатор быстродействия для анализа определенных таблиц в базе данных:

**Советы по повышению производительности**

* Проектируйте таблицы, не содержащие избыточных данных. Хорошая структура таблицы является необходимым исходным условием для эффективного получения и обновления данных. Если все же таблицы содержат слишком много данных, используйте мастер анализа таблиц для разделения их на связанные таблицы, что повысит эффективность хранения данных. Для получения дополнительных сведений нажмите кнопку .
* Выбирайте подходящий тип данных для полей. Это поможет уменьшить размеры базы данных и увеличит скорость выполнения операций связи. При описании поля следует задать для него тип данных наименьшего размера, позволяющий хранить нужные данные. Для получения дополнительных сведений нажмите кнопку.
* Создавайте индексы для сортируемых и объединяемых полей, а также для полей, используемых при задании условий отбора. Существенное повышение скорости выполнения запросов приносит индексирование полей, расположенных по обе стороны отношения, или создание связи между этими полями, а также индексирование всех полей, используемых для заданий условий отбора в запросе. Поиск записей при помощи диалогового окна *Поиск* также выполняется намного быстрее, если он ведется по индексированному полю. Индексы занимают место на диске и замедляют операции добавления, изменения и удаления записей. Однако в большинстве случаев выигрыш в скорости получения данных перевешивает неудобства при обновлении данных. Если приложение обновляет данные очень часто, или если на жестком диске мало свободного места, использование индексов можно ограничить; во всех остальных случаях их следует использовать везде, где это возможно. Если данные в поле являются однозначными, обязательно используйте однозначный индекс.
* В составном индексе, включающем несколько полей, используйте ровно столько полей, сколько для него необходимо.

**Повышение производительности при работе со связанной таблицей**

Связанные таблицы используются как обычные таблицы Microsoft Access, но следует помнить, что на самом деле они не находятся в базе данных Microsoft Access. Для каждого просмотра данные связанной таблицы извлекаются из другого файла. Это требует времени, особенно если связанная таблица находится в сети или в базе данных SQL.

Если связанная таблица хранится в сети или в базе данных SQL, следующие правила помогут получить наилучшие результаты:

Если оставлять открытыми связанные базы данных, то можно повысить производительность при открытии основной базы данных и при открытии таблиц и форм. Для этого необходимо создать пустую таблицу в связанной базе данных и связать ее с основной базой данных. Для открытия связанной таблицы используется метод OpenRecordset. Это предотвращает постоянное открытие и закрытие связанных баз данных, а также создание и удаление связанных файлов .ldb ядром базы данных Microsoft Jet.

* Просматривайте только необходимые данные. Не перемещайтесь без необходимости по экранам в режиме таблицы. В больших таблицах избегайте перехода к последней записи. Если требуется добавить в большую таблицу новые записи, выполняйте команду Ввод данных из меню Записи, чтобы не загружать в память существующие записи.
* Применяйте фильтры или запросы, чтобы ограничить количество записей, просматриваемых в режиме формы или таблицы. В этом случае по сети передается меньше данных.
* В запросах, включающих связанные таблицы, старайтесь не использовать функции в условиях отбора в запросе. Особенно избегайте в любых местах запроса применения статистических функций по подмножеству, таких как Dsum. При вызове статистической функции по подмножеству, Microsoft Access для выполнения запроса загружает все данные из связанной таблицы.
* Если приходится часто добавлять данные в связанную таблицу, создайте для добавления записей форму, у которой в свойстве Ввод данных (Data Entry) задано значение "Да". В этом случае в форме, открываемой для ввода данных, не выводятся существующие записи. Это экономит много времени, позволяя избежать загрузки всех записей из связанной таблицы.
* Помните, что другие пользователи могут пытаться получить доступ к внешней таблице одновременно с вами. Если база данных Microsoft Access используется в сети, старайтесь не блокировать записи дольше, чем это необходимо.

[Изображение выглядит как текст

Автоматически созданное описание](#_Основное_задание)