**29.10.2020 ИС** [**Основы проектирования БД**](http://www.vartet.narod.ru/do/2210-is2-bd.doc)

Самостоятельная работа

1. Написать программный код формирования таблицы для СУБД FoxPro c помощью команды

|  |
| --- |
| **CREATE TABLE - SQL Command** |
|  |

|  |
| --- |
|  |

Создает таблицу, с использованием указанных полей или массива.

|  |
| --- |
|  |
| CREATE TABLE | DBF *TableName1* [NAME *LongTableName*] [FREE]  [CODEPAGE = nCodePage]  ( *FieldName1* *FieldType* [( *nFieldWidth* [, *nPrecision*] )] [NULL | NOT NULL]  [CHECK *lExpression1* [ERROR *cMessageText1*]]  [AUTOINC [NEXTVALUE *NextValue* [STEP *StepValue*]]] [DEFAULT *eExpression1*]  [PRIMARY KEY | UNIQUE [COLLATE *cCollateSequence*]]  [REFERENCES *TableName2* [TAG *TagName1*]] [NOCPTRANS]  [, *FieldName2* ... ]  [, PRIMARY KEY *eExpression2* TAG *TagName2* |, UNIQUE *eExpression3* TAG *TagName3*  [COLLATE *cCollateSequence*]]  [, FOREIGN KEY *eExpression4* TAG *TagName4* [NODUP]  [COLLATE *cCollateSequence*]  REFERENCES *TableName3* [TAG *TagName5*]] [, CHECK *lExpression2* [ERROR *cMessageText2*]] )  | FROM ARRAY *ArrayName* |

**Параметры**

**CREATE TABLE | DBF *TableName1***

Создает таблицу или .dbf. Параметр *TableName1* указывает имя таблицы. Опции **TABLE** и **DBF** идентичны.

***nCodePage***

Указывает кодовую страницу, которая будет использоваться для вновь создаваемой таблицы. Список поддерживаемых кодовых страниц вы можете посмотреть в описании [Кодовые страницы, поддерживаемые Visual FoxPro](http://www.foxclub.ru/html/a3d7b0e0-8320-44b1-8983-17c30a78c6c4.htm).

**NAME *LongTableName***

Указывает длинное имя для таблицы. Вы можете указать длинное имя для таблицы только в том случае, когда открыта база данных, поскольку длинные имена таблиц хранятся в базе данных. Длинные имена таблиц могут содержать до 128 символов в длину и могут быть использованы в базе данных вместо коротких имен.

**FREE**

Определяет, что таблица не будет ассоциирована с открытой базой данных. Если у вас нет открытой базы данных, то ключевое слово **FREE** можно не использовать.

***FieldName1*, *FieldType*, *nFieldWidth,nPrecision***

Указывает соответственно имя поля, тип поля, его длину и точность (число десятичных мест). Одиночная таблицы может содержать до 255 полей. Если одно или более полей позволяют использование null-значений, допустимое число полей снижается до 254.

Параметр *FieldType* представляет собой одну букву или длинное имя, указывающее тип данных поля. Вы можете указать *nFieldWidth*, *nPrecision* или оба для некоторых типов полей. В приведенной ниже таблице перечислены значения для *FieldType* и указано, можете ли вы указать *nFieldWidth* и *nPrecision*.

|  |  |  |  |
| --- | --- | --- | --- |
| ***FieldType – тип поля*** | ***nFieldWidth – размер поля*** | ***nPrecision – точность(зн.после зап.)*** | **Data type – тип данных** |
| W, Blob | - | - | **Blob** |
| C, Char, Character | n | – | Поле **Character** длиной в *n* символов |
| Y, Currency | – | – | **Currency** |
| D, Date | – | – | **Date** |
| T, DateTime | – | – | **DateTime** |
| B, Double | – | d | **Double** |
| G, General | – | – | **General** |
| I, Int, Integer | – | – | **Integer** |
| L, Logical | – | – | **Logical** |
| M, Memo | – | – | **Memo** |
| N, Num, Numeric | n | d | Поле **Numeric** с длиной *n* и *d* десятичными местами |
| F, Float | n | d | Поле **Numeric** с плавающей запятой длиной *n* с *d* десятичными местами |
| Q, Varbinary | n | - | Поле **Varbinary** с длиной *n* |
| V, Varchar | n | - | Поле **Varchar** с длиной *n* |

Параметры *nFieldWidth* и *nPrecision* игнорируются для полей типов W, Y, D, T, G, I, L и M. Если *nPrecision* не указана для полей типов N или F, то параметр *nPrecision* по умолчанию устанавливается равным нулю (без десятичных мест). Если *nPrecision* не указана для поля типа B, то параметр *nPrecision* устанавливается по умолчанию в число десятичных мест, определенных установкой команды **SET DECIMALS**.

**NULL | NOT NULL**

Определяет - будут ли разрешены null-значения для хранения в поле. **NULL** разрешает хранение null-значений, а **NOT NULL**, соответственно, запрещает. Если одно или более полей могут хранить null-значения, максимальное число полей на таблицу ограничивается значением 254.

**CHECK *lExpression1***

Указывает правило проверки для поля. Параметр *lExpression1* должен оцениваться в логическое выражение и может представляет собой пользовательскую функцию или хранимую процедуру. Visual FoxPro проверяет правило проверки, указанное в опции **CHECK** при добавлении пустой записи.

**ERROR *cMessageText1***

Указывает сообщение об ошибке. Visual FoxPro будет отображать это сообщение, когда правило проверки, указанное в опции **CHECK** будет генерировать ошибку. Сообщение отображается только тогда, когда данные изменяются в окне Browse или окне Edit.

**AUTOINC [NEXTVALUE *NextValue*[STEP *StepValue*]]**

Разрешает автоувеличение для поля. *NextValue* указывает начальное значение и может представлять, как положительное, так и отрицательное значение, лежащее в диапазоне от -2,147,483,647 до 2,147,483,647. Значением по умолчанию является единица (1). Вы можете установить значение *NextValue,* используя счетчик **Next Value** на закладке **Fields** дизайнера таблиц.

*StepValue*указывает значение увеличения для поля и может быть положительным, ненулевым значением в диапазоне от 1 до 255. Значением по умолчанию является единица (1). Вы можете установить значение *StepValue* с помощью счетчика **Step** на закладке **Fields** дизайнера таблиц.

Варианты таблиц:

1. посетители библиотеки, поля:
   1. номер по пор.
   2. Фамилия
   3. Имя
   4. Отчество
   5. Дата рождения
   6. Н.пункт адреса
   7. Ул. Адреса
   8. Дом
   9. Квартира
2. Книги библиотеки
   1. Номер по пор.
   2. Автор
   3. Кол.экз.
   4. Год издания
   5. Ном.хранилища
3. клиенты автомойки
   1. номер по пор.
   2. Фамилия
   3. Имя
   4. Отчество
   5. Ул. Прож.
   6. Дом прож
   7. КВ.
   8. Марка автом
   9. Год выпуска автом.
   10. Цвет автом
   11. Тех. Состояние(испр.-неисправ)

Условия:

- таблицы «свободные»(Free) – если вы для прошлой работы делали с БД с подобной темой, то включение таблицы в такую БД приветствуется;

- имена таблиц и полей пишите английским шрифтом.

- ответы прислать на эл. Почту [vas2498@yandex.ru](mailto:vas2498@yandex.ru)

- попробуйте на скачанном VFP9 свою программку выполнить. Если получится – результат присылайте. Вопросы присылайте по эл.почте. У кого не получилось скачать VFP9 – пишите, будем искать выход. Всем успехов.

Ваш преподаватель Шабашов Василий Валентинович.

# Создание и изменение таблиц в Microsoft Visual FoxPro

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Команда CREATE [*FileName* | ?] открывает проектировщик таблицы VFP для ее создания.  Параметр (см. также табл. 1.6):  *FileName* – имя или полное имя создаваемой таблицы. Расширение может быть опущено.  Если таблица создается, когда имеется текущая база данных, то таблица будет добавлена в эту базу данных.  Имя таблицы не должно содержать дефиса. Имена устройств MS-DOS, такие, как CON, NUL, PRN и COM1, не должны использоваться в качестве имени таблицы.  Команда CREATE TABLE – SQL  CREATE TABLE | DBF TableName1 [NAME LongTableName]                    [CODEPAGE = *nCodePage*]                    [FREE] (FieldName1 FieldType [(nFieldWidth [, nPrecision])]                    [NULL | NOT NULL] [CHECK lExpression1 [ERROR cMessageText1]]                    [AUTOINC [NEXTVALUE NextValue [STEP StepValue]]]                    [DEFAULT eExpression1] [PRIMARY KEY | UNIQUE                    [COLLATE cCollateSequence]] [REFERENCES TableName2                    [TAG TagName1]] [NOCPTRANS] [, FieldName2 ...]                    [, PRIMARY KEY eExpression2 TAG TagName2,                    | UNIQUE eExpression3 TAG TagName3 [COLLATE cCollateSequence]]                    [, FOREIGN KEY eExpression4 TAG TagName4 [NODUP]                    [COLLATE cCollateSequence] REFERENCES TableName3                    [TAG TagName5]] [, CHECK lExpression2                   [ERROR cMessageText2]]) | FROM ARRAY ArrayName  создает таблицу по списку полей или из массива *ArrayName*. Если таблица создается, когда имеется текущая база данных, и опущена опция FREE, то таблица будет добавлена в эту базу данных.  Опции и параметры:  CREATE TABLE | DBF*TableName1* – создает таблицу с именем DBF-файла*TableName1*. Опции TABLE и DBF равнозначны. Имя таблицы может включать путь к DBF-файлу. Расширение DBF в имени файла может быть опущено. Если путь не указан, то таблица создается в директории по умолчанию.  Имя *TableName1* без пути и расширения называется *коротким именем*таблицы базы данных.  *nCodePage* – номер используемой кодовой страницы; список кодовых страниц приведен в разд. 3.7.6.  NAME*LongTableName* – длинное имя таблицы. Оно может быть задано, только когда открыта база данных (таблица создается как часть базы данных), поскольку именно в ней хранятся длинные имена. Длинное имя может содержать до 128 символов (буквы, цифры, знаки подчеркивания) и употребляться взамен короткого имени таблицы.  Длинное имя не должно совпадать с длинными именами других таблиц.  Если опция NAME опущена, то длинное и короткое имена таблицы совпадают.  VFP отображает длинное имя, если оно задано, всякий раз, когда таблица появляется в интерфейсе, например в Project Manager, Database Designer, Query Designer, View Designer или в заголовке Browse-окна.  Длинное имя употребляется в других оперирующих с таблицами баз данных командах и функциях, например в команде RENAME TABLE и функции DBGETPROP( ).  Пример:  create table d:\VendIntl name VendorsInternational (Company C(40))  *Замечание*. Параметр *LongTableName*, как, впрочем, и имя поля, может быть указано в виде символьной строки, например:  create table d:\VendIntl name "VendorsInternational" ("Company" C(40))  FREE – указывает, что таблица не включается в открытую базу данных (создается свободная таблица). Если база данных не открыта, использование FREE необязательно. Если опция FREE задана, а файл с именем *TableName1* уже существует, то поведение приложения зависит от установки SET SAFETY.  *FieldName1*,*FieldType*,*nFieldWidth,* *nPrecision*– имя поля, тип поля, длина поля и точность. Число полей в таблице не должно быть более 255 или 254, если хотя бы одно поле таблицы поддерживает NULL.  Параметр*FieldType* – это короткое или длинное имя типа данных поля. С полями некоторых типов должны задаваться параметры *nFieldWidth* и *nPrecision* (вместе или порознь). Всевозможные варианты задания параметров описаны в табл. 15.2.  Таблица 15.2  Варианты задания параметров *FieldType*,*nFieldWidth* и *nPrecision*   |  |  |  |  | | --- | --- | --- | --- | | *FieldType* | *nFieldWidth* | *nPrecision* | Тип данных | | W, Blob | – | – | Blob | | C, Char, Character | n | – | Character (символьное поле длины *n*) | | Y, Currency | – | – | Currency | | D, Date | – | – | Date | | T, DateTime | – | – | DateTime | | B, Double | – | d | Числовое поле типа Double c *d* десятичными знаками | | G, General | – | – | General | | I, Int, Integer | – | – | Integer | | L, Logical | – | – | Logical | | M, Memo | – | – | Memo | | N, Numeric | n | d | Числовое поле типа Numeric дины *n* с *d* десятичными знаками | | F, Float | n | d | Числовое поле типа Float дины *n* с *d* десятичными знаками | | Q, Varbinary | n | – | Поле типа Varbinary длины *n* | | V, Varchar | n | – | Поле типа Varchar длины *n* |   Параметры *nFieldWidth* и *nPrecision* игнорируются для W, D, T, I, Y, L, M и G типов данных. Если параметр *nPrecision* не включен для полей типа N и F, то он считается равным нулю. Если параметр *nPrecision* не включен для поля типа B, то он принимается равным установке команды SET DECIMALS.  NULL | NOT NULL – указывает, допустим ли NULL для поля или нет. Если опция NULL | NOT NULL опущена, то берутся установки команды SET NULL, но не всегда: если включена опция PRIMARY KEY или UNIQUE, то независимо от установки SET NULL берется NOT NULL.  В то же время VFP не генерирует сообщения об ошибке, если создается первичный или кандидатный индекс и одновременно разрешается NULL, несмотря на то, что NULL не допустим в полях, используемых для индексов такого рода. Сообщение об ошибке, однако, возникнет при работе программы, если будет выполнена попытка установить в это поле NULL.  Если действует опция NOT NULL, запрещающая ввод NULL, то при попытке ввести в поле NULL генерируется ошибка.  CHECK*lExpression1* – правило проверки поля. Выражение *lExpression1* является логическим и может быть пользовательской или хранимой процедурой. Проверка завершается с положительным результатом, если *lExpression1* вычисляется со значением .T.  VFP вычисляет *lExpression1* и в том случае, когда добавляется чистая запись.  Опция CHECK может быть употреблена только для таблицы, включаемой в базу данных.  ERROR*cMessageText1* – сообщение об ошибке. VFP генерирует это сообщение, когда *lExpression1* вычисляется со значением .F.; если опция ERROR опущена, то генерируется системное сообщение об ошибке.  Сообщение об ошибке генерируется VFP, в том числе и когда данные изменяются в Edit (Change) или Browse-окне.  *Пример*. При добавлении чистой записи нарушается правило TypeId > 0, заданное командой CREATE TABLE для поля TypeId таблицы WorkT.  && Должна быть открыта и выбрана база данных.  && В противном случае употреблении опции CHECK недопустимо  if not Dbused('Homelibrary') then                                              open database d:\Homelibrary\Homelibrary  else                                              set database to Homelibrary  endif  && Таблица WorkT не должна быть частью базы данных  if Indbc('WorkT', 'TABLE') then                                              MessageBox('Таблица WorkT уже имеется в базе данных!')                                              return  endif  create table d:\WorkT.dbf ;                                              (TypeId Int check TypeId > 0 ;                                                 error "Значение поля TypeId должно быть положительно", ;                                              WorkType Char(20))  && Добавляем чистую запись  && Возникающее при этом сообщение об ошибке приведено на рис. 15.1  append blank    Рис. 15.1. При выполнении команды APPEND BLANK нарушено правило TypeId > 0  AUTOINC [NEXTVALUE*NextValue* [STEP*StepValue*]] – устанавливает автоматическое увеличение значения поля типа Integer на величину шага *StepValue*: значение поля вновь добавленной записи будет равно значению поля предшествующей записи +*StepValue*. Для поля первой записи берется значение *NextValue*, которое может быть целым числом в диапазоне от –2,147,483,647 до 2,147,483,647; значение по умолчанию *–* число 1.  Параметр *StepValue* по умолчанию также равен 1 и может изменяться в диапазоне от 1 до 255. Опция STEP не может быть задана без опции NEXTVALUE.  Попытка задать опцию AUTOINC полю, тип которого отличен от Integer, приведет к генерации ошибки.  *Замечание*. Поле, имеющее тип Integer (AutoInc), доступно только для чтения. Следующее значение поля может быть задано в поле Next Value вкладки Fields проектировщика таблиц. Там же в поле Step устанавливается и шаг приращения значения. Программно эти величины регулируются SQL-командой ALTER TABLE.  Пример:  close databases  open database d:\HomeLibrary\HomeLibrary.dbc  create table d:\WorkT.dbf (TypeId I autoInc nextValue 1 step 2, WorkType C(20))  for k = 1 to 5  insert into WorkT (WorkType) values ('Type' + Transform(k))  next  browse last  DEFAULT*eExpression1* – значение поля по умолчанию, используемое при добавлении записи. Тип выражения *eExpression1* должен совпадать с типом поля.  Опция DEFAULT может быть задана только для таблицы базы данных.  Если употреблена опция AUTOINC, то значение по умолчанию не используется.  PRIMARY KEY | UNIQUE – создает первичный (опция PRIMARY KEY) или кандидатный (опция UNIQUE) ключ, использующий в качестве выражения *FieldName1* и имеющий то же имя.  COLLATE *cCollateSequence* – задает collate-последовательность. По умолчанию для *cCollateSequence* используется MACHINE. Опция может быть использована, если указана опция PRIMARY KEY | UNIQUE.  *Пример.* Создается таблица WorkT, имеющая 2 поля, кандидатный ключ TypeId и collate-последовательностью "RUSSIAN".  if CPCurrent( ) = 1251 then  create table d:\WorkT.dbf free (TypeId I unique collate "RUSSIAN", WorkType C(20))  else  MessageBox('Установите кодовую страницу 1251')  cancel  endif  REFERENCES*TableName2*[TAG*TagName1*] – указывает родительскую таблицу *TableName2*, с которой устанавливается постоянная связь. Родительская таблица не должна быть свободной.  Параметр *TagName1* – это индекс родительской таблицы. Если опция TAG опущена, то в устанавливаемой связи используется первичный ключ таблицы *TableName2*. Если такового не имеется, то генерируется ошибка.  Устанавливаемая таким образом связь, поскольку индексы дочерней и родительской таблиц являются уникальными, имеет вид 1:1.  NOCPTRANS – запрещает перевод поля типа Character, Memo или Varchar в другую кодовую страницу; приводит к появлению поля Character (binary), Memo (binary) или Varchar (binary).  *FieldName2 ...* – задает одно или более дополнительных полей и их атрибутов.  PRIMARY KEY*eExpression2*TAG*TagName2* – создает первичный индекс, в котором *eExpression2*может содержать произвольное правильно построенное выражение, включающее комбинации полей. Опция PRIMARY KEY может быть использована только единожды.  В опции TAGпараметр*TagName2* – это имя первичного индекса; оно не должно превышать 10 символов.  UNIQUE*eExpression3*TAG*TagName3* – создает кандидатный индекс с именем *TagName3*. Индексное выражение *eExpression3*может содержать произвольное правильно построенное выражение, включающее комбинации полей. Однако если поле уже указано для первичного индекса, оно не может входить в *eExpression3*. Кандидатных индексов может быть несколько.  FOREIGN KEY*eExpression4*TAG*TagName4*[NODUP] – создает регулярный индекс *TagName4*с ключевым выражением *eExpression4*. Опция NODUP, если задана, означает, что создается не регулярный, а кандидатный индекс.  REFERENCES*TableName3*TAG*TagName5* – задает имя *TableName3*родительской таблицы, с которой устанавливается связь, и имя индекса *TagName5*, на котором эта связь основана. Если опция TAG опущена, то по умолчанию для связи будет использован первичный ключ таблицы *TableName3*. Если такового нет, то возникнет ошибка.  *Пример.* Создается таблица BookCont; между таблицей BookCont и таблицей WorkType, имеющей первичный индекс TypeId устанавливатся связь.  close databases  open database d:\HomeLibrary\HomeLibrary.dbc  && Изображение установленной связи см. на рис. 15.3  create table d:\BookCont.dbf (BookId Int, TypeId Int, ;                                              foreign keyTypeId tagTypeId references WorkType tag TypeId, ;                                              Work Char(50))    CHECK*lExpression2* – правило проверки таблицы. Тип выражения *lExpression2* – логический. Проверка выполняется и при добавлении чистой записи.  ERROR*cMessageText2* – сообщение, выдаваемое, если *lExpression2* вычисляется со значением .F.; если опция ERROR опущена, то генерируется системное сообщение об ошибке.  *Пример*. При добавлении чистой записи нарушается правило TypeId > 0, заданное командой CREATE TABLE для всей таблицы WorkT.  create table d:\WorkT.dbf ;                                              (TypeId I, WorkType C(20),                                                 check TypeId > 0 ;                                                 error "Значение поля TypeId должно быть положительно")  && Добавляем чистую запись  append blank  && Возникающее при этом сообщение об ошибке приведено на рис. 15.1  *Замечание*. Проверка для поля в отличие от проверки для таблицы не отделяется от имени поля и описания его типа запятой.  FROM ARRAY*ArrayName* – массив, содержимое которого используется для создания таблицы. Массив может быть сформирован функцией AFIELDS( ).  Опция поддерживается только для таблиц, входящих в базу данных, то есть не может употребляться с опцией FREE.  С опцией FROM ARRAY нужно употреблять опцию NAME, поскольку в противном случае возникнет неуникальность длинного имени таблицы.  *Пример*. Создается массив, содержащий структуру открытой таблицы WorkType, и затем по его данным создается новая таблица WorkT, входящая в текущую базу данных.  if not Used('WorkType') then  wait 'Откройте таблицу WorkType' window  return  endif  && Копируем структуру таблицы WorkType в массив  Afields(arrayWithTableStructure*,*'WorkType')  && Создаем новую таблицу WorkT  create table d:\WorkT.dbf name WorkT from array arrayWithTableStructure  После выполнения команды CREATE TABLE новая таблица открывается в первой неиспользуемой рабочей области. Таблица независимо от установки SET EXCLUSIVE открывается для монопольного доступа.  Опции CHECK, DEFAULT, FOREIGN KEY, FROM ARRAY, NAME, PRIMARY KEY и REFERENCES могут употребляться, если создаваемая таблица включается в базу данных, которая в момент выполнения CREATE TABLE должна быть открыта. В противном случае генерируется ошибка.  Название опции UNIQUE употребляется для ANSI-совместимости. Кандидатный индекс, создаваемый, если в команде CREATE TABLE – SQL (или ALTER TABLE – SQL) присутствует эта опция, не то же самое, что индекс, создаваемый командой INDEX, использующей опцию UNIQUE. Индекс, создаваемый командой INDEX с опцией UNIQUE разрешает дублирование ключей, в то время как кандидатный индекс это запрещает.  Команда ALTER TABLE – SQL ALTER TABLE TableName1 ADD | ALTER [COLUMN] FieldName1                    FieldType [(nFieldWidth [, nPrecision])] [NULL | NOT NULL]                    [CHECK lExpression1 [ERROR cMessageText1]]                    [AUTOINC [NEXTVALUE NextValue [STEP StepValue]]]                    [DEFAULT eExpression1] [PRIMARY KEY | UNIQUE                    [COLLATE cCollateSequence]]                    [REFERENCES TableName2 [TAG TagName1]]                    [NOCPTRANS] [NOVALIDATE] или ALTER TABLE TableName1 ALTER [COLUMN] FieldName2                    [NULL | NOT NULL] [SET DEFAULT eExpression2]                    [SET CHECK lExpression2 [ERROR cMessageText2]]                    [DROP DEFAULT] [DROP CHECK] [NOVALIDATE] или ALTER TABLE TableName1 [DROP [COLUMN] FieldName3]                    [SET CHECK lExpression3 [ERRORcMessageText3]] [DROP CHECK]                    [ADD PRIMARY KEY eExpression3 [FOR lExpression4] TAG *TagName2*                    [COLLATE cCollateSequence]] [DROP PRIMARY KEY]                    [ADD UNIQUE eExpression4 [[FOR lExpression5] TAG TagName3                    [COLLATE cCollateSequence]]] [DROP UNIQUE TAG TagName4]                    [ADD FOREIGN KEY [eExpression5] [FOR lExpression6] TAG TagName4                    [COLLATE cCollateSequence] REFERENCES TableName2 [TAG TagName5]]                    [DROP FOREIGN KEY TAG TagName6 [SAVE]]                    [RENAME COLUMN FieldName4 TO FieldName5] [NOVALIDATE] позволяет программно изменять структуру таблицы.  Если команда ALTER TABLE применяется для свободной таблицы, то включение опций CHECK, DEFAULT, FOREIGN KEY, NOVALIDATE, PRIMARY KEY, REFERENCES и SET приведет к генерации ошибки.  Опции и параметры:  Смысл опций NULL | NOT NULL, CHECK *lExpression1*, ERROR *cMessageText1*, AUTOINC [NEXTVALUE*NextValue* [STEP*StepValue*]], DEFAULT *eExpression1*, PRIMARY KEY | UNIQUE, COLLATE *cCollateSequence*, REFERENCES *TableName2* [TAG *TagName1*], NOCPTRANS, ERROR *cMessageText2*, REFERENCES *TableName2* [TAG *TagName5*] см. в описании команды CREATE TABLE.  ALTER TABLE*TableName1* – задает имя модифицируемой таблицы. Имя *TableName1* может содержать путь. Расширение может быть опущена. Если таблица открыта, то достаточно указать ее псевдоним. Если таблица закрыта, то она будет открыта и станет текущей.  ADD | ALTER [COLUMN] *FieldName1* – имя добавляемого (ADD) или изменяемого (ALTER) поля.  Применение ALTER COLUMN, если не использована опция COLLATE, устанавливает заданную по умолчанию последовательность сортировки (collate-последовательность). Чтобы восстановить нужную последовательность, нужно обновить индексы таблицы (команда REINDEX).  *FieldType* [(*nFieldWidth* [, *nPrecision*])] – тип поля, его длина и точность. Варианты задания параметров *FieldType*,*nFieldWidth* и *nPrecision* см. в табл. 15.2.  *Пример*. Добавляется символьное поле в таблицу Friends, а затем запись со значением в этом поле. После просмотра таблицы тип поля меняется на числовой. Затем добавленное поле удаляется.  &&  && Добавляем поле NewField  alter table d:\Friends add column NewField C(15)  insert into Friends (NewField) values ("23.5 градуса")  browse last  &&  && Изменяем тип поля NewField  alter table Friends alter column NewField N(10, 2)  browse last                          &&  && Удаляем поле NewField  alter table Friends drop column NewField  browse last  NOVALIDATE – позволяет при изменениях нарушать ссылочную целостность данных таблицы. По умолчанию опция не действует.  ALTER [COLUMN] *FieldName2* – задает имя модифицируемого поля. Чтобы изменить более одного свойства поля, ALTER COLUMN включается в команду неоднократно.  SET DEFAULT *eExpression2* – задает новое значение по умолчанию для существующего поля. Тип *eExpression2* должен совпадать с типом поля.  Значение по умолчанию не может быть задано для поля Integer (AutoInc).  SET CHECK *lExpression2* – задает новое правило проверки существующего поля.  DROP DEFAULT – удаляет значение по умолчанию существующего поля.  DROP CHECK - удаляет правило проверки существующего поля.  DROP [COLUMN] *FieldName3* – удаляет поле с именем *FieldName3* из таблицы. Одновременно удаляются и значение по умолчанию поля и его правило проверки.  Если удаляемое поле участвует в индексном выражении или в триггере, то они становятся ошибочными. Однако ошибки при удалении поля не генерируются, они, если не исправлены, обнаружатся на этане исполнения.  SET CHECK *lExpression3* – правило проверки таблицы.  ERROR *cMessageText3* – сообщение, выдаваемое, если *lExpression3* вычисляется со значением .F.; если опция ERROR опущена, то генерируется системное сообщение об ошибке.  DROP CHECK – удаляет правило проверки таблицы.  ADD PRIMARY KEY *eExpression3* [FOR *lExpression4*] TAG *TagName2* – добавляет первичный индекс, в котором *eExpression2*– это индексное выражение. Опция ADD PRIMARY KEY может быть использована только единожды.  Выражение *lExpression4* задает фильтр – доступны будут только те записи, для которых *lExpression4* вычисляется со значением .T. Опцию FOR, однако, не следует использовать при задании первичного индекса, поскольку уникальность первичного ключа обеспечивается только для тех записей, которые удовлетворяют условию *lExpression4*. Взамен для фильтрации записей следует употреблять опцию FOR с командой INDEX.  Чтобы увеличить производительность за счет использования рашмор-технологии, в *lExpression4* следуют включать оптимизируемые выражения.  В опции TAGпараметр*TagName2* – это имя первичного индекса; оно не должно превышать 10 символов. Если опция опущена и *eExpression3* – это поле таблицы, то имя этого поля будет использовано в качестве имени первичного индекса.  *Пример*. Создается первичный ключ Cust\_id таблицы Customer.  alter table Customer add primary key Cust\_id tag Cust\_id  alter table Customer alter column Cust\_id C(5) primary key  DROP PRIMARY KEY – удаляет первичный индекс таблицы. Удаление первичного индекса приводит к разрушению связей, на нем основанных.  ADD UNIQUE *eExpression4* [TAG *TagName3* [FOR *lExpression5*]] – добавляет кандидатный индекс с именем *TagName3*. Смысл опции FOR тот же, что и в ADD PRIMARY KEY.  DROP UNIQUE TAG *TagName4* – удаляет кандидатный индекс с именем *TagName4*.  ADD FOREIGN KEY [*eExpression5*] TAG *TagName4* [FOR *lExpression6*] – добавляет регулярный индекс *TagName4*с ключевым выражением *eExpression5*. Смысл опции FOR тот же, что и в ADD PRIMARY KEY.  DROP FOREIGN KEY TAG *TagName6* [SAVE] – удаляет регулярный индекс *TagName6*. Если использована опция SAVE, то индекс останется в CDX-файле, или будет из него удален – в противном случае.  RENAME COLUMN *FieldName4* TO *FieldName5* – изменяет имя поля *FieldName4* на имя *FieldName5*.  Если поле *FieldName4* участвует в индексных выражениях, правилах проверки и пр., то выполнение RENAME COLUMN может не привести к замене *FieldName4* на *FieldName5* во всех выражениях.  Команда ALTER TABLE может перестраивать таблицу путем создания нового заголовка таблицы и добавления в него записей. Например, изменение типа или длины поля ведет к перестройке таблицы.  После перестройки выполняются правила проверки полей для каждого поля, тип или длина которого изменены. При изменении типа или длины поля также выполняется правило проверки таблицы. При обнаружении первой ошибки будет выдано сообщение.  Для удаления свойства AUTOINC команда ALTER TABLE используется с опцией ALTER COLUMN, в которой опция AUTOINC опущена.  При добавлении свойства AUTOINC поля типа Integer автоматическое увеличение значения поля начинается при добавлении новой записи. Значения поля, получившего свойство AUTOINC, в прежних записях таблицы не обновляются.  При употреблении ALTER TABLE для курсоров VFP команда использует правила, применяемые к свободным таблицам, и если курсор обладает свойствами, которые поддерживаются только для таблиц базы данных, например длинные имена полей, то поведение ALTER TABLE может быть непредсказуемым. Поэтому применять ALTER TABLE с курсорами следует, лишь когда результат изменения очевиден.  Если изменяется существующее поле и выражения первичного или кандидатных индексов состоят из полей таблицы, то VFP проверяет эти поля на предмет наличия в них NULL или неуникальных значений. При обнаружении таковых VFP генерирует ошибку; таблица не модифицируется.  Команда MODIFY STRUCTURE открывает проектировщик текущей таблицы. Если в текущей рабочей области таблица не открыта, то возникнет диалог Open.  *Замечание*. Изменение типа поля может привести к потере данных из-за невозможности преобразования старого типа в новый.  При изменении структуры таблицы создается ее копия с расширением BAK и копия memo-файла (если таковой имеется) с расширением TBK. При принятии изменений данные переносятся из копий в измененную таблицу.  Если после принятия изменений процесс копирования данных прерван, то новая таблица будет содержать не все записи таблицы-оригинала.  Перед изменением структуры можно установить новое значение для размера блока memo-данных (команда SET BLOCKSIZE). Этот размер будет использован для файла с измененной структурой.  Команда COPY STRUCTURE TO TableName [FIELDS FieldList]                    [[WITH] CDX | [WITH] PRODUCTION]                    [DATABASE cDatabaseName [NAME LongTableName]] создает новую пустую таблицу с той же структурой, что и текущая таблица.  Опции и параметры:  *TableName* – имя создаваемой таблицы. Может включать путь. Расширение необязательно.  FIELDS *FieldList* – список полей, которые переносятся в структуру новой таблицы. Имена полей в списке разделяются запятыми. Если опция FIELDS *FieldList* опущена, то в структуре новой таблицы будут присутствовать все поля таблицы-оригинала.  [WITH] CDX | [WITH] PRODUCTION – создает структурный индексный файл для новой таблицы, идентичный индексному файлу таблицы-оригинала. При этом, однако, если создаваемая таблица является свободной, то первичный индекс таблицы-оригинала конвертируется в кандидатный индекс новой таблицы. Если создается таблица базы данных, то статус индекса не меняется.  Опции CDX и PRODUCTION равнозначны.  *Пример*. Создается свободная таблица NewAuthors; в качестве таблицы-оригинала берется таблица Authors.  close databases  close tables all  open database d:\HomeLibrary\HomeLibrary.dbc  use Authors in 0  copy structure to d:\NewAuthors with cdx  use d:\NewAuthors in 0  select NewAuthors  browse last  DATABASE *cDatabaseName* – задает имя существующей базы данных, в которую добавляется создаваемая таблица.  NAME *LongTableName* – длинное имя новой таблицы базы данных; может содержать до 128 символов. Если опция опущена, то будет использовано имя, заданное параметром *TableName* (берется имя файла без пути и расширения).  *Пример*. Создается таблица NewAuthors как таблица базы данных HomeLibrary с длинным именем OneMoreAuthors; в качестве таблицы-оригинала берется таблица Authors.  close databases  close tables all  open database d:\HomeLibrary\HomeLibrary.dbc  use Authors in 0  if not InDBC('OneMoreAuthors', 'TABLE') then                                              copy structure to d:\NewAuthors with cdx dataBase HomeLibrary name OneMoreAuthors                                              use OneMoreAuthors in 0                                              select OneMoreAuthors                                              browse last  else                                              wait 'Таблица OneMoreAuthors уже включена в базу HomeLibrary' window  endif  Команда COPY STRUCTURE EXTENDED TO TableName [FIELDS FieldList]                    [DATABASE DatabaseName [NAME LongTableName]] создает новую таблицу *TableName* с записями, содержащими структуру текущей таблицы. Каждому полю таблицы-оригинала отвечает одна запись создаваемой таблицы с информацией об этом поле.  Опции параметры:  FIELDS *FieldList* – список полей, отображаемых в виде записей создаваемой таблицы. Если опция опущена, то отображаются все поля.  Смысл прочих опций и параметров см. в описании команды COPY STRUCTURE TO.  В записи новой таблицы 18 полей, перечисленных в табл. 15.3.  Таблица 15.3  Поля таблицы, создаваемой COPY STRUCTURE EXTENDED   |  |  |  | | --- | --- | --- | | Имя поля | Тип | Что содержит | | FIELD\_NAME | Character | Имя поля таблицы-оригинала | | FIELD\_TYPE | Character | Символ, характеризующий тип поля: W – Blob, C – Character, Y – Currency, N – Numeric, F – Float, I – Integer, B – Double, D – Date, T – DateTime, L – Logical, M – Memo, G – General, Q – Varbinary, V – Varchar и Varchar (Binary) | | FIELD\_LEN | Numeric | Длина поля | | FIELD\_DEC | Numeric | Число десятичных позиций (для числовых полей) | | FIELD\_NULL | Logical | Поддержка NULL | | FIELD\_NOCP | Logical | Флаг запрета изменения кодовой страницы поля (для символьных и memo-полей) | | FIELD\_DEFA | Memo | Значение поля по умолчанию | | FIELD\_RULE | Memo | Правило проверки поля | | FIELD\_ERR | Memo | Error-текст правила проверки поля | | TABLE\_RULE | Memo | Правило проверки таблицы (только первая запись) | | TABLE\_ERR | Memo | Error-текст правила проверки таблицы (только первая запись) | | TABLE\_NAME | Character | Длинное имя таблицы (только первая запись) | | INS\_TRIG | Memo | Выражение триггера вставки (только первая запись) | | UPD\_TRIG | Memo | Выражение триггера обновления (только первая запись) | | DEL\_TRIG | Memo | Выражение триггера удаления (только первая запись) | | TABLE\_CMT | Memo | Комментарий таблицы (только первая запись) | | FIELD\_NEXT | Numeric | Следующее значение для AutoInc-поля | | FIELD\_STEP | Numeric | Шаг для AutoInc-поля. Нуль означает, что поле не имеет свойства AUTOINC |   Данные таблицы *TableName* могут быть использованы командой CREATE FROM для создания новой таблицы. Сама же таблица *TableName* может быть перед употреблением в CREATE FROM отредактирована либо интерактивно, либо программно.  *Пример*. Создается таблица d:\AuthorsStructure со структурой таблицы Authors, а затем команда CREATE FROM использует d:\AuthorsStructure для получения копии таблицы Authors.  close databases  close tables all  open database d:\HomeLibrary\HomeLibrary.dbc  use Authors in 0  copy structure extended to d:\AuthorsStructure  && Фрагмент таблицы AuthorsStructure приведен на рис. 15.5  create d:\NewAuthors from d:\AuthorsStructure  use d:\NewAuthors in 0  select NewAuthors  browse last  Рис. 15.5. Фрагмент таблицы AuthorsStructure  Команда CREATE [FileName1 [DATABASE DatabaseName                    [NAME LongTableName]]] FROM [FileName2] создает таблицу по данным таблицы, построенной командой COPY STRUCTURE EXTENDED.  Созданная таблица открывается и становится текущей.  Опции и параметры:  *FileName1* – имя файла создаваемой таблицы. Может включать путь и не содержать расширения.  FROM [*FileName2*] – имя таблицы, созданной COPY STRUCTURE EXTENDED.  Таблица *FileName2* может быть также построена интерактивно или отредактирована после создания COPY STRUCTURE EXTENDED. При этом для создания таблицы *FileName1* берутся все записи таблицы*FileName2*, включая удаленные.  Смысл опций DATABASE и NAME см. в описании команды COPY STRUCTURE TO. |