5. Блокировка неправильных данных

Необходимо задать правила проверки, которые отвергают подозрительные данные в момент их ввода, после того как неверные данные попали в БД, найти их труднее.

В этом параграфе описан основной набор средств проверки, имеющийся у программы Access:

- *основные*, включая совпадения, обязательные поля и значения по умолчанию;
- *маски ввода*, форматирующие, с помощью образцов обычный текст, такой как почтовые коды и телефонные номера;
- *правила верификации* – строгие правила для полей, не подчиняющиеся никаким законам;
- *подстановки*, ограничивающие возможные значения списком заранее заданных вариантов.

О целостности данных

Все средства проверки программы Access реализованы в режиме *Конструктора*. Для их применения выбирается поле, и настраиваются его свойства. Единственная сложность – знать, какие свойства наиболее полезны.

Программа Access предоставляет три варианта переключения в режим *Конструктора*:

- щелкнуть правой кнопкой мыши на заголовке вкладки таблицы и выбрать из меню команду *Конструктор*;
- использовать кнопку *Режим* на вкладке ленты *Главная;*
- воспользоваться крошечными кнопками режима в правом нижнем углу окна программы Access.

Запрет незаполненных полей

Для того чтобы каждая запись имела какой-то смысл, в ней должен быть хотя бы абсолютный минимум информации. Без помощи пользователя программа Access не может отделить важную информацию от необязательных деталей. По этой причине все поля в новой таблице определены как необязательные, за

исключением поля первичного ключа (которое обычно содержит идентификационный номер записи).

Этот недостаток легко исправить - просто выбрать в *Конструкторе* поле, которое обязательно должно быть заполнено и задать в свойстве *Обязательное поле* значение *Да* (рис. 5.1).

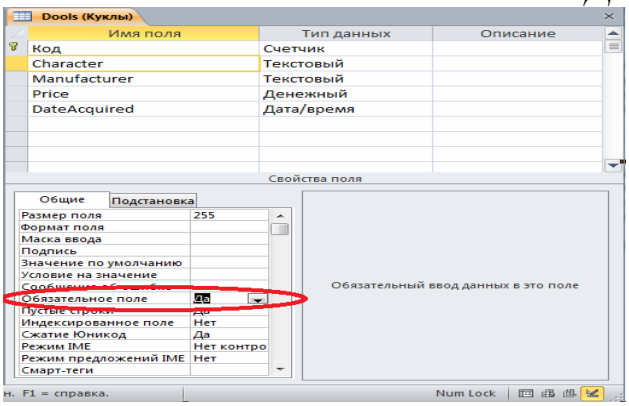

Рис.5.1. Свойство Обязательное поле сообщает программе Access о запрете пропущенных значений

Access проверяет свойство *Обязательное поле* при каждом добавлении новой записи или исправлении поля в уже имеющейся записи. Но если в таблице уже есть данные, нет гарантии, что они соответствуют установленным правилам.

Когда сохраняется таблица, возвращаясь в *Режим таблицы* или закрывая таблицу, программа Access дает возможность проверить записи, уже внесенные в таблицу (рис. 5.2).

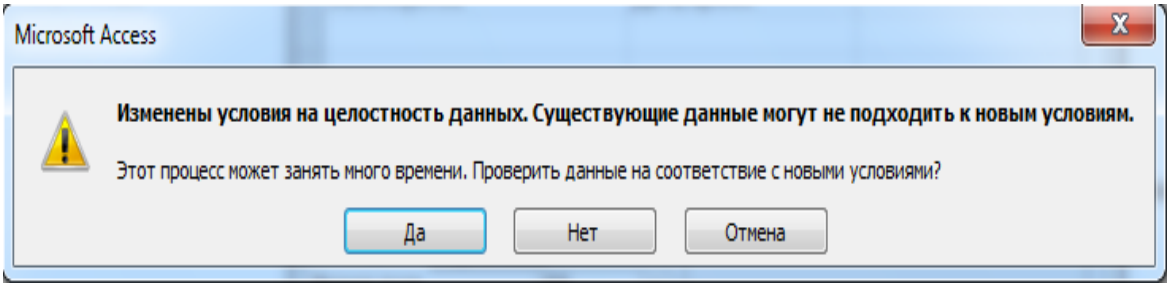

Рис.5. 2. Проверка условия заполненности таблицы

Если выбирается выполнение проверки, и Access обнаруживает проблему, программа позволяет отменить внесенные корректировки (рис. 5.3)

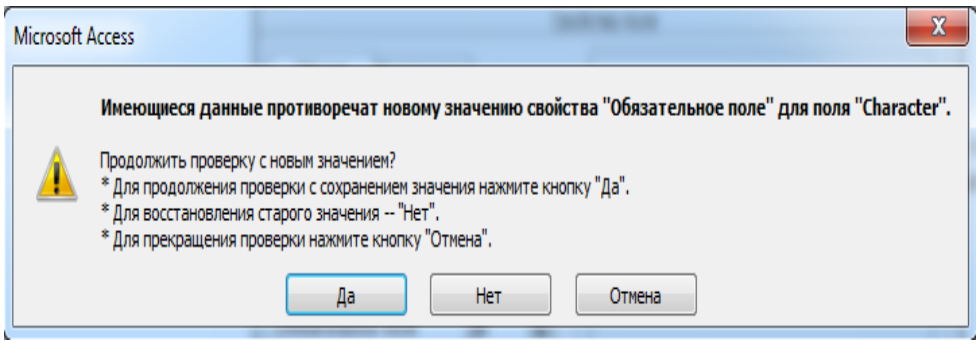

Рис. 5..3. Обнаруженные ошибки

Это хорошая идея проверить таблицу на соответствие новым требованиям, которые были установлены, иначе некорректные данные могут остаться в БД.

Если программа Access находит пропущенное значение, она останавливает поиск и спрашивает, что делать. Можно сохранить изменения даже если они конфликтуют хотя бы с одной записью – в итоге, по крайней мере, новые записи не будут порождать подобную проблему. Другой возможный вариант – вернуть прежнее, более терпимое значение свойству поля. В любом случае можно найти пропущенные данные, отсортировав данные с помощью вопроса, выводящего незаданные значения в верхние строки таблицы.

Нужно хорошенько подумать, какой минимум данных нужен для создания записи.

Следует принять за правило применение необязательного поля в том случае, когда данные в момент ввода записи для него необязательны или недоступны.

Пропущенные значения и пустые строки

Программа Access поддерживает свойство *Обязательное поле* для всех типов данных. Но, возможно, для некоторых типов данных понадобятся дополнительные проверки. Это объясняется тем, что свойство *Обязательное поле* запрещает только незаполненные поля – поля, в которых нет совсем никаких данных. Но программа Access, что кажется несколько странным, различает пропущенные значения и пустые строки.

Пропущенное значение (null) означает отсутствие данных. Пустая строка свидетельствует о том, что значение поля было введено, но оказалось пустым. Разница существует, т. к. БД, такие

как Access, должны распознавать пропущенные данные. Для того, чтобы проверить эту разницу в таблице, нужно создать текстовое поле со значением свойства *Обязательное поле*, равным *Да*, попытаться вставить новую запись и оставить ее пустой. Access остановит это действие. Если попробовать вставить новую запись, но поместить единственный пробел в поле, происходит странная вещь: Access автоматически обрезает пробелы и, делая это, превращает единственный пробел в пустую строку. Но сообщения об ошибке не будет, поскольку пустая строка – это не то же самое, что пропущенное значение.

Задание значений по умолчанию

До сих пор поля в таблицах заполнялись пользователем, вставлявшим запись или пропускавшим ее. Но есть еще одна возможность – можно определить значение по умолчанию. Теперь, если кто-то вставляет запись и пропускает поле, программа Access использует в нем значение по умолчанию.

Задается значение по умолчанию в свойстве поля *Значение по умолчанию*. Access вставляет значение по умолчанию, когда создается новая запись, но всегда можно изменить это значение. Также можно во время редактирования поля вернуться к значению по умолчанию с помощью сочетания клавиш *<Ctrl>+<Alt>+<Пробел>*.

Очень удобно использовать значение по умолчанию как отправную точку для новой записи. Например, когда создается новая запись в таблице, можно редактировать значение по умолчанию, а не заменять его полностью другим значением. Программа Access оценивает их, когда вводится новая запись, что означает зависимость выбранного значения по умолчанию от других данных записи. Динамические значения по умолчанию используют выражения (специальные формулы БД), способные выполнять вычисления или извлекать другие подробности. Полезная функция *Date()* извлекает текущую дату, установленную на компьютере. Если применить эту функцию как значение по умолчанию для поля с датой (как показано на рис. 5.4), программа Access автоматически вставляет текущую дату при вводе новой записи.

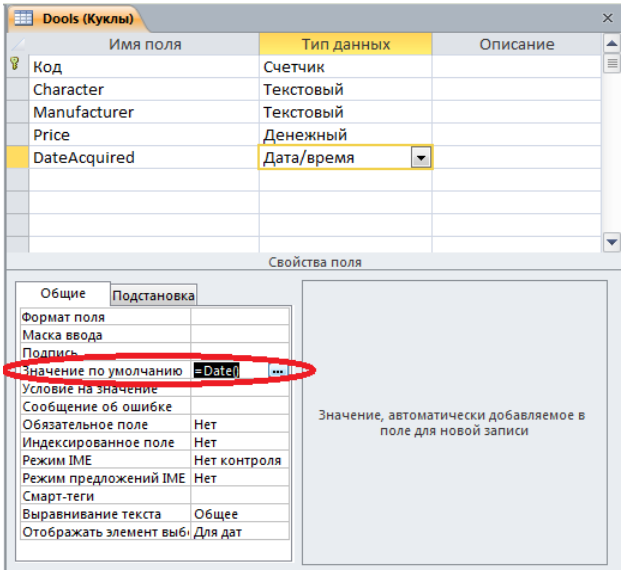

Рис.5.4. Установка текущей системной даты по умолчанию

Если применяется функция *Date()*, как значение по умолчанию в поле *Значение по умолчанию* в таблице, то при каждой вставке новой записи программа Access вставляет текущую дату.

Предотвращение дублирования значений с помощью индексов

Первое правило любой таблицы – каждая включенная в нее запись должна быть уникальна. Для соблюдения этого требования нужно выбрать первичный ключ, одно или несколько полей, которые не должны дублироваться в разных записях.

Как уже известно, самый надежный способ – создание идентификационного поля (поля Код) для первичного ключа. Можно заставить поле требовать уникальных значений с помощью индекса. Индекс БД похож на предметный указатель в книге – это список значений из поля с перекрестной ссылкой, которая указывает на соответствующий раздел (полную запись).

Применение индекса с полем – хитрость, заключающаяся в применении свойства *Индексированное поле*, которое доступно для данных всех типов за исключением типа *Вложение* и типа *Объект OLE*. Когда добавляется поле, у его свойства *Индексированное поле* указано значение *Нет*. Для вставки индекса и предупреждения дублирования значений можно изменить в *Конструкторе* значение свойства *Индексированное поле* на *Да* (*Совпадения не допускаются*). При выборе третьего варианта – *Да* (*Допускаются*

совпадения) – создается индекс, но разрешается нескольким записям иметь одинаковые значения поля. Этот вариант не поможет поймать повторяющиеся записи, но его можно применить для ускорения поиска.

Когда закрывается *Конструктор* после изменения свойства *Индексированное поле*, программа Access напоминает о необходимости сохранить сделанные корректировки. В этот момент она создает любые нужные ей новые индексы. Нельзя создать индекс, запрещающий совпадения, если в таблице уже есть дублирующая информация. В данной ситуации Access выводит сообщение об ошибке, когда закрывается *Конструктор* и программа пытается добавить индекс.

Индексы – это средство предотвращения ввода неверных данных или средство, повышающее производительность.

Индексы не только препятствуют дублированию значений. Они также незаменимы, когда нужно увеличить скорость обычного поиска.

Важно знать, что индексы улучшают производительность только в случае очень больших и сложных таблиц.

Индексы для нескольких полей

Также можно применять индексы для предотвращения повторений комбинации значений. Пусть создается таблица для хранения списка ваших друзей и их контактной информации. Могут встретиться одинаковые имена и фамилии. Если есть желание помешать включению в две записи одинаковых и имени, и фамилии, то такой запрет избавит от случайного ввода сведений об одном и том же человеке дважды.

Маски ввода

БД ценят непротиворечивость данных. Например, если есть поле *Рост*, лучше использовать в нем значения, заданные в одних и тех же единицах измерения. Аналогично, если есть поле *Номер телефона*, лучше убедиться в том, что у всех номеров один и тот же формат. Если одни телефонные номера записаны с дефисами, пробелами и скобками (например, (844) 547-1123), в то время как другие несколько отличаются (скажем, 847-547-1123), а третьи вообще пропускают междугородний код (547-1123), тогда появится

небольшая проблема. Из-за недостатка согласованности будет трудно обрабатывать такие данные, например, искать конкретный телефонный номер или отсортировать телефонные номера в зависимости от междугороднего кода.

Для облегчения обработки значений, имеющих фиксированный шаблон, – например, телефонных номеров – можно воспользоваться маской ввода. *Маска ввода* (или маска для краткости) предоставляет возможность сообщить программе Access, какой шаблон или образец должны использовать вводимые данные. Основываясь на этом образце, Access изменяет способ ввода и редактирования значений, делая их более понятными и менее подверженными ошибкам.

Можно добавить маску для любого поля с текстовым типом данных. По сравнению с обычным текстом маски обладают рядом достоинств:

- *Маски управляют элементом ввода*. Будучи пустым, шаблон маски отображает символы-заполнители, на место которых должны попасть значения. В пустой маске телефонного номера отображается текст (_ _ _)_ _ _ - _ _ _ , ясно обозначающий вид необходимых данных.
- *Маски помогают понять смысл данных*. Гораздо легче читать множество значений, представленных определенным образом. Большинство людей быстрее найдут нужные номера социального обеспечения, если они будут представлены как (012-86-7180), а не как (012867180).
- *Маски предупреждают ошибки*. Они отбрасывают символы, не соответствующие шаблону. Если используется маска для ввода номеров телефонов, то нельзя ввести буквы.
- *Маски устраняют путаницу*. Одни и те же данные многих типов можно представить несколькими способами. Можно ввести номера телефонов с междугородним кодом и без него. Используя маску с символами-заполнителями для междугороднего кода, следует понимать, что эта информация обязательна и показывается, где она должна располагаться. Очевидно, что не нужно набирать скобки

или дефисы для разделения номеров, поскольку эти детали уже стоят в нужном месте. Такие же преимущества маски дают при вводе дат, которые можно ввести разнообразными способами (*Год/Месяц/День, Месяц-День-Год и т. д.*).

Маски подходят как нельзя лучше для сортировки числовой информации в текстовом поле. Этот сценарий реализуется с самыми разными данными, включая номера кредитных карт, почтовые индексы и номера телефонов. Данные этих типов не следует хранить в числовых полях, поскольку они не должны интерпретироваться как единый номер. Их следует воспринимать как последовательность цифр.

Маски поддерживают только типы данных *Текстовый* и *Дата/время*.

Создание собственной маски

Мастер создания масок ввода предоставляет очень ограниченный набор вариантов масок. Если применяется маска для данных особого вида, например, специальный код клиента, применяемый на предприятии, то нужно создать собственную маску.

Создать маску очень легко, но придется потратить немного времени, прежде чем получится желаемый результат. Есть два варианта:

- набрать или отредактировать маску непосредственно в поле свойства *Маска ввода*;
- запустить *Мастер создания масок ввода*, выбрать одну из масок как отправную точку (как описано в предыдущем разделе) и затем перейти во второе окно мастера. Достоинство этого варианта в возможности тестирования создаваемой маски в поле *Проба* до того, как она будет сохранена как часть своей таблицы.

В любой маске есть три типа символов:

- заполнители указывают, куда вводить символ;
- специальные символы сообщают программе Access о способе интерпретации части маски;

- литералы и любые другие символы служат элементами оформления, которые облегчают трактовку значения.

Рассмотрим предыдущий пример о маске номера телефона – !(999) 000-000. Символы 9 и 0 – заполнители: они указывают, куда вводить цифры номера телефона. Скобки, пробел и дефис – просто средства форматирования – литералы. И всего один специальный символ – восклицательный знак. Он сообщает Access о том, что символы должны вводиться в маску слева направо, стандартный и единственный имеющий смысл в случае телефонного номера вариант.

Для того, чтобы разложить все по полочкам, предложены следующие таблицы: в табл. 5.1 приведены все заполнители, которые можно использовать в масках ввода, в табл. 5.2 перечислены другие специальные символы. Все остальное автоматически относится к литералам.

Таблица 5.1.

Символы-заполнители для масок ввода

Символ	**Описание**
0	Обязательная цифра (от 0 до 9).
9	Необязательная цифра (от 0 до 9).
#	Необязательная цифра, знак плюс (+) или знак минус (–).
L	Обязательная буква.
?	Необязательная буква.
A	Обязательная буква или цифра.
a	Необязательная буква или цифра.

Таблица 5.2.

Специальные символы для масок ввода

Символ	**Описание**
&	Обязательный символ любого типа (включая буквы, цифры, знаки пунктуации и т. д.).
C	Необязательный символ любого типа (включая

	буквы, цифры знаки пунктуации и т. д.).
!	Обозначает направление заполнения маски слева направо при вводе. Это направление выбрано по умолчанию, поэтому данный символ не требуется (но во встроенные маски он включен).
<	Преобразует все следующие за ним символы в строчные.
>	Преобразует все следующие за ним символы в прописные.
\	Указывает на следующий символ как литерал. Например, у символа # специальное назначение в масках. Поэтому если вы хотите обычный символ # включить в маску, следует ввести \#.

Далее приведено несколько примеров масок:

- (000) 000-000. В телефонный номер обязательно должны быть включены цифры междугороднего кода. Эта маска отличается от маски телефонного номера предлагаемой *Мастером создания масок*. В последней первые три 0 заменены 9, что делает междугородний код необязательным.
- 99:00:00 >LL. Маска для ввода времени в поле типа *Дата/время*. Она формируется из двух цифр для часов и двух цифр для минут. Последние два символа благодаря наличию символа > всегда отображаются, как прописные, и предназначены для обозначения половины суток AM или PM.
- 099.099.099.099. *IP-адрес*, идентифицирующий компьютер в сети. Он записан как четыре значения, разделенные точками. В каждой части адреса должна быть, как минимум, одна цифра, а как максимум – три. Такой шаблон в маске отображается комбинацией 099 (одна обязательная цифра, за которой следуют две необязательные).
- *Пароль*. Маска, допускающая ввод обычного текста разной длины с одной лишь разницей, все символы отображаются звездочками (*) и скрыты от любопытных глаз.

- Маски могут заканчиваться двумя необязательными элементами, разделенными точкой с запятой (;).

Вторая составляющая маски – число, сообщающее программе Access, должна ли она сохранять литеральные символы маски в записи БД. Это последний вопрос, который задает *Мастер создания масок*. Если этот фрагмент маски пропустить или использовать цифру 1, Access сохраняет только символы, которые вводит пользователь. Если же применить цифру 0, программа сохранит весь текст вместе с литералами.

В третьей составляющей маски содержится символ-заполнитель. Если этот компонент маски пропустить, программа Access применяет знакомый знак подчеркивания.

Далее приведена маска, в которую включены оба дополнительных компонента:

(000) 000-000;1;#

Во второй части стоит 1, а в третьей – #. Маска предназначена для ввода телефонных номеров и сохранения их в БД вместе с литералами маски, в данном случае двумя скобками, пробелом и дефисом. В данной маске вместо знака подчеркивания в качестве заполнителя используется знак номера (#).

Для добавления собственной маски нужно использовать кнопки переходов между записями, расположенные у нижнего края этого окна, для перехода в конец. Это окно можно использовать и для изменения маски. Например, встроенная маска телефонного номера не требует обязательного включения междугороднего кода.

Иногда удается создать маску, которая невероятно полезна, и хочется ее использовать в разных таблицах БД. Несмотря на то, что можно скопировать маску в каждое поле, которое нуждается в ней, у программы Access есть более удачное средство – можно хранить маску в списке масок программы.

Для вставки маски в список нужно перейти к свойству поля *Маска ввода* (любого поля) и щелкнуть мышью кнопку со скругленными углами для запуска *Мастера создания масок*. Затем щелкнуть мышью кнопку *Список*, раскрывающую удобное окно, в котором можно редактировать маски, предоставляемые Access, и вставить собственную.

Правила верификации или условия на значения

Маски ввода – замечательное средство, но они применяются лишь с информацией нескольких определенных типов, обычно с текстом фиксированной длины, имеющим единственный неизменный шаблон. Для создания действительно "пуленепробиваемой" таблицы следует применять более сложные ограничения, такие как гарантия попадания числа в определенный диапазон, проверка еще не наступивших дат или первой буквы текста. Правила верификации или условия на значения могут помочь сформировать все эти ограничения, используя всю мощь языка *SQL (Structured Query Language*, язык структурированных запросов).

Суть правила верификации проста. Задается ограничение, сообщающее программе Access, какие значения разрешены в поле, а какие нельзя считать правильными. Когда кто-нибудь добавляет новую запись или редактирует имеющуюся, Access проверяет, удовлетворяют ли данные вашим условиям на значения. Если нет, программа выводит сообщение об ошибке, заставляет откорректировать ошибочные данные и попробовать еще раз.

Применение условия на значение поля

У каждого поля может быть одно условие на значение или правило верификации. Далее приведены действия, необходимые для задания такого правила. Начнем с простого условия, запрещающего вводить в числовое поле 0 или любое отрицательное число, а в следующих разделах можно отшлифовать навыки создания правил верификации настолько, что будет можно защитить и данные других типов.

Для вставки нужного *условия на значение* следует выполнить указанные действия:

- В *Конструкторе* выбрать поле, к которому применяется условие. Данные всех типов, кроме: *Поле MEMO*, Счетчик и Объект OLE, поддерживают *условие на значение*.
- В свойстве поля *Условие на значение* ввести проверочное выражение (рис.5.5).

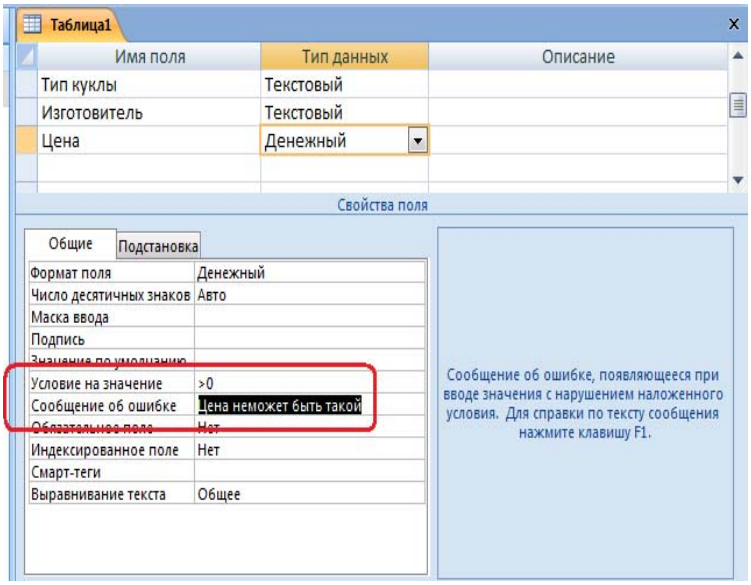

Рис.5.5. Установка условия на значение и текст при возникновении ошибки

Выражение представляет собой фрагмент на языке *SQL*, выполняющий проверку введенных данных. Программа Access проверяет данные на значение, когда введена порция данных и будет выполняться переход к другому полю или другой записи. Например, условие >0 – это правило верификации, требующее ввода в поле только положительных значений.

В данном примере свойство *Условие на значение* препятствует вводу недопустимых значений, а свойство *Сообщение об ошибке* содержит текст сообщения. Если введено значение, не прошедшее проверку, программа Access отвергнет его и выведет этот текст сообщения об ошибке в диалоговом окне. Если не будет предложено никакого текста, программа отобразит *условие на значение* для данного поля, которое было введено Как проверочное выражение.

Запись условия на значение поля

Как видите, применять *условие на значение* к полю достаточно легко. Но формирование правила требует больших умственных усилий. Для получения желаемого результата вы должны сделать первый шаг в мир языка *SQL*.

Несмотря на то, что условия на значения ограничены только вашим воображением, разработчики, профессионально работающие о программе Access, возвращаются к нескольким основным шаблонам снова и снова. В следующих разделах

представлены базовые знания для быстрой и легкой разработки правил верификации данных разных типов.

Программа Access применяет правило верификации, если в поле есть данные. Если поле остается пустым, программа принимает его без всякой проверки. Если не нравится такой подход, нужно задать в свойстве *Обязательное поле* значение *Да*, чтобы добиться обязательного заполнения поля.

Проверка допустимости числовых значений

Для числовых данных самый распространенный вариант проверки – это принадлежность введенного значения определенному диапазону. Другими словами, проверяется, больше или меньше введенное число другого значения. Такими инструментами в этом случае – знаки операций сравнения < и >. В табл. 5.3 приведено несколько часто используемых примеров.

Таблица 5.3

Условия на значение для чисел

Сравнение	**Пример условия**	**Описание**
Меньше чем	<100	Значение должно быть меньше 100
Больше чем	>0	Значение должно быть больше 0
Не равно	< >42	Значение может быть любым, но не равным 42
Меньше или равно	<=100	Значение должно быть не больше 100
Больше или равно	>=0	Значение должно быть не меньше 0
Равно	=42	Значение должно быть 42.
Между	Between 0 and 100	Значение должно быть 0,100 или любое промежуточное значение

Проверка допустимости дат

Как и в случае числовых данных, проверка допустимости дат, как правило, включает проверку принадлежности даты определенному диапазону. Задача – убедиться в том, что у даты формат, подходящий для условия на значение. Если используется условие >Jan 30, 2012 (> 30 Янв, 2012), программа Access приходит в крайнее замешательство, т. к. не понимает, что текст (Jan 30, 2012) предназначается для представления даты. Точно так же, если проверяется условие >1/30/12, Access предполагает, что числа справа от знака сравнения – часть выражения с последовательными операциями деления.

Для решения этой проблемы используется универсальная синтаксическая форма представления дат программы Access, которая выглядит следующим образом: #1/30/2012#

В универсальную синтаксическую запись для представления дат компоненты включаются в порядке месяц/день/год и обрамляются с обеих сторон символами #. С помощью этого синтаксиса можно использовать условие, такое как >#1/30/2012#, требующее, чтобы вводимая дата была больше (наступала позже).

Универсальная синтаксическая запись может включать и время, например: #1/30/2012 5:30PM#

При сравнении двух дат программа Access принимает во внимание сведения о времени. Дата #1/30/2012# не содержит данных о времени, поэтому она интерпретируется как наступившая в самую первую секунду суток. В результате Access считает, что значение #1/30/2012 8:00 AM# больше, поскольку наступает на 8 часов позже.

Теперь, зная об универсальной синтаксической записи для дат, можно использовать любые операции сравнения, которые применяются для сравнения чисел. Можно применять и следующие удобные функции для получения информации о текущих дате и времени:

- *Date()* – вычисляет текущую дату (без какой-либо информации о времени, поэтому она вычисляется как первая секунда текущего дня);
- *Now()* – вычисляет текущий момент времени, включая дату и время.

В табл. 5.4 приведено несколько примеров.

Таблица 5.4

Условия на значения для дат

Сравнение	Пример условия	Описание
Меньше чем	<#1/30/2012#	Дата до 30 января 2012 г.
Больше чем	>#1/30/2012 5:30 PM#	Любая дата после 30 января 2012 г., или 30 января 2012 г. после 17:30
Меньше или равна	<=#1/30/2012#	Дата до 30 января 2012 г. или первая секунда 30 января 2012 г.
Больше или равна	>=#1/30/2012#	30 января 2012 г. или любая более поздняя дата
Больше текущей даты	>Date()	Сегодня или более поздняя дата
Меньше текущей даты	<Date()	Вчера или более ранняя дата
Больше текущей даты (и времени)	>Now()	Сегодня после текущего времени или любая дата в будущем
Меньше текущей даты (и времени)	<Now ()	Сегодня до настоящего момента или любая дата в прошлом

Проверка допустимости текста

В случае текста условие на значение позволяет задать начальный или конечный символ текста или наличие определенных символов в строке. Все эти задачи решаются с помощью оператора *Like*, сравнивающего введенный текст с образцом. Следующее условие требует начинать поле с буквы "*П": Like "П**"

Звездочка обозначает отсутствие символов или присутствие нескольких символов. Таким образом, полное условие заставляет программу Access проверять, начинается ли строка с буквы "П" (или "п"), за которой могут следовать символы или нет.

Очень похожее условие можно применять для проверки завершающих символов фрагмента текста: *Like "*ая"*

Это условие считает корректными значения: белая, черная и 34z%($)#ая.

Не столь распространенный прием – использование нескольких звездочек. В следующем выражении требуется наличие букв "a" и "b" (именно в таком порядке, но не обязательно сразу друг за другом) в любом месте строки текста:

*Like "*a*b*"*

Наряду со звездочкой в операторе *Like* могут использоваться и некоторые другие символы. Можно применять ?, соответствующий единичному символу, что очень удобно, если известна длина текста или позиция определенной буквы в тексте.

Далее приведено условие на значение для восьмисимвольного кода изделия, заканчивающегося *0ZB*:

Like "?????0ZB"

У символа # аналогичная роль, но он представляет цифру. Таким образом, следующее правило верификации определяет код изделия, заканчивающийся комбинацией символов 0ZB, которой предшествует пять цифр:

Like "#####0ZB"

И, наконец, можно ограничить значение любого символа набором определенных букв или символов. Для этого нужно заключить допустимые символы в квадратные скобки.

Предположим, что ваша компания использует восьмисимвольный код изделия, который всегда начинается с "A" или "E". Далее приведено необходимое условие на значение:

Like "[AE]???????"

Фрагмент [AE] представляет один символ, который может принимать значение A или E. Если нужно разрешить символы A, B, C, D, следует написать в условии [ABCD] или воспользоваться удобной сокращенной формой [A-D], означающей разрешение любого символа от A до D, включая A и D.

Далее приведено условие на значение, разрешающее ввод семибуквенного слова и запрещающего цифры и другие символы. Оно формируется семикратным повторением кода [A- Z], который разрешает любую букву:

Like [A-Z] [A-Z] [A-Z] [A-Z] [A-Z] [A-Z] [A-Z]

Комбинирование условий на значения

Независимо от типа данных можно комбинировать условия двумя способами. Используя ключевое слово *And*, можно создать правило верификации, содержащее два условия. Этот прием очень полезен, т. к. у поля может быть только одно условие на значение.

Для использования ключевого слова *And* просто записать два правила верификации и вставить между ними слово *And*. Неважно, какое правило указано первым. Далее приведено условие на значение даты, которая должна была наступить до текущей даты, но после 1 января 2012 г.:

<Date( ) And >#l/l/2012#

Можно также использовать ключевое слово *Or* для принятия значения, удовлетворяющего одному из условий. В следующем правиле разрешены числа, большие 1000 или меньшие -1000: >1000 Or < -1000.

Создание условия на значение для таблицы

Условия на значения всегда применяются к отдельному полю. Но проектировщики БД часто нуждаются в средствах сравнения значений разных полей. Поскольку это правило верификации включает в себя два поля, единственный способ вставить его – создать условие на значение для всей таблицы. Табличные правила верификации могут применять все уже известные приемы и удалять значения из любого поля текущей записи.

При создании условия на значение для таблицы в *Конструкторе* следует выбрать на ленте *Работа с таблицами* | *Конструктор* → *Показать или скрыть* → *Страница свойств*. Справа в окне программы появляется страница с дополнительными параметрами (рис. 5.6).

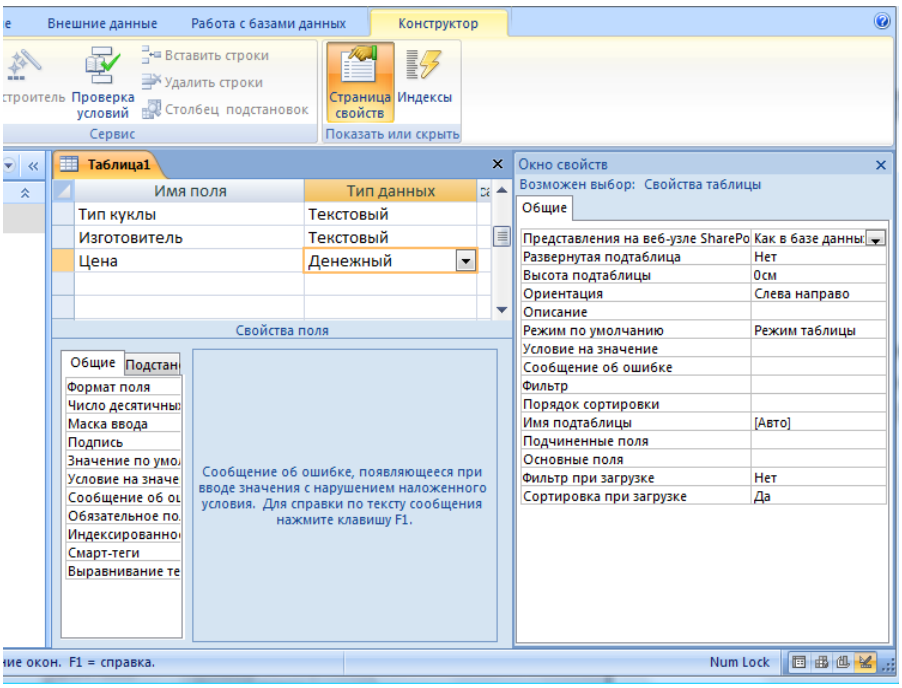

Рис.5.6. Окно свойств

В окне свойств отображена некоторая информация о таблице в целом, включая параметры сортировки и фильтрации, примененные на листе данных, а также условие на значение.

Подстановки

Подстановки – еще одно средство, позволяющее стандартизировать данные. Подстановки, в первую очередь, позволяют вставить значение в поле из подготовленного списка возможных вариантов.

У программы Access два основных типа списков подстановок: списки с набором фиксированных значений, и списки, полученные из связанной таблицы.

Подстановки не поддерживают следующие типы данных: *Поле MEMO, Дата/время, Денежный, Счетчик, Логический, Объект OLE, Гиперссылка и Вложение*.

Создание простого списка подстановок из констант

Простые списки подстановок имеют смысл, если короткий простой список не нуждается в частых корректировках.

Для создания списка можно воспользоваться следующими действиями:

- Открыть таблицу в *Конструкторе*.

- Найти поле, в которое нужно вставить список подстановок.
- Убедиться в том, что у поля корректный тип данных.

Подстановки применяются чаще всего для данных *Текстового* и *Числового* типов. Для этого следует:

- Выбрать в списке типов данных *Мастер подстановок*. Это действие на самом деле не изменяет заданный тип данных. Оно лишь сообщает программе Access о том, что нужно запустить мастер *Создание подстановки*, базирующийся на текущем типе данных. Как только выбран описанный вариант, на экране появляется окно мастера *Создание подстановки* (рис. 5.7).

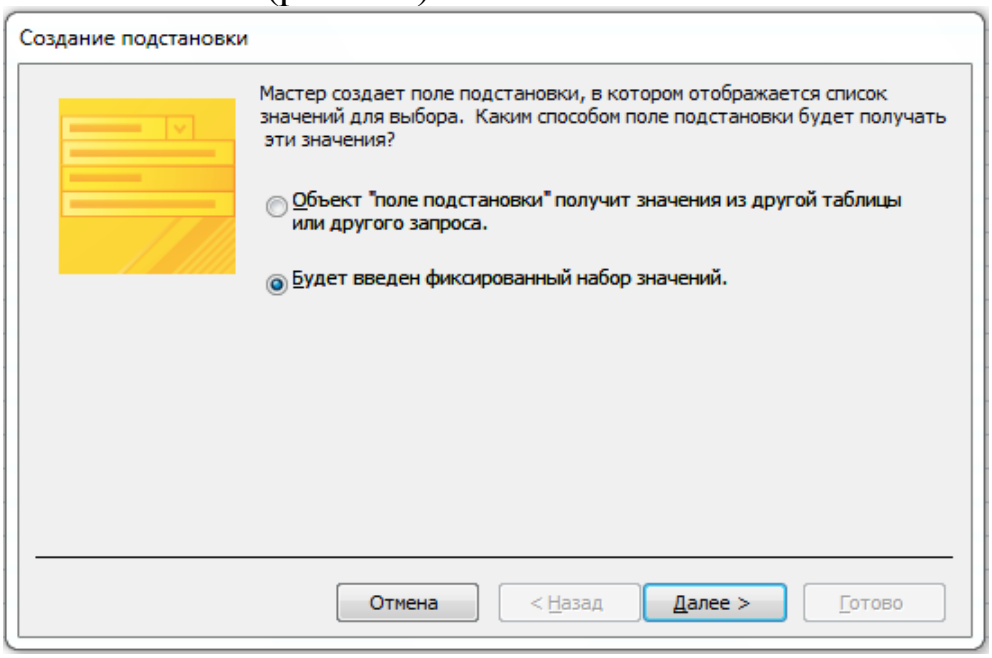

Рис.5.7. Выбор источника подстановок

- Выбрать переключатель *Будет введен фиксированный набор значений*.
- Щелкнуть мышью кнопку *Далее*.
- В следующем окне *Создание подстановки* можно ввести список значений, которые следует использовать по одному в каждой строке. В списке можно заполнить несколько столбцов данных.
- Щелкнуть мышью кнопку *Далее*.
- На экране появляется последнее окно мастера *Создание подстановки*.

- Указать, можно ли хранить в столбце подстановки множественные значения.
- Если разрешено наличие множественных значений, список подстановок отобразит флажок рядом с каждым элементом. Можно выбрать несколько значений для одной записи, установив несколько флажков.
- Щелкнуть мышью кнопку *Готово*.
- Перейти в *Режим таблицы*: щелкнуть правой кнопкой мыши заголовок вкладки, выбрать *Режим таблицы* и сохранить изменения.

При переходе в поле со списком подстановок справа можно увидеть стрелку, направленную вниз. Если щелкнуть ее кнопкой мыши, то на экране появится раскрывающийся список со всеми введенными вариантами. Далее следует выбрать один из вариантов для вставки в поле.

Создание списка подстановки из другой таблицы

Эффективно хранить список подстановок в отдельной таблице.

Далее приведено несколько доводов в пользу применения отдельной таблицы:

- Можно вставлять, редактировать и удалять элементы, просто корректируя таблицу подстановок.
- Можно многократно использовать один и тот же список подстановок в нескольких разных полях как в одной таблице, так и в разных. Такой подход устраняет бесконечные операции копирования и вставки.
- Можно хранить дополнительную информацию.

Списки подстановок в виде таблиц немного сложнее, поскольку они включают связь таблицы – ссылку, объединяющую две таблицы вместе и иногда порождающую новые ограничения.

Добавление новых значений в список подстановок

Когда создается подстановка, использующая константы, список предоставляет лишь перечень предложений. Можно игнорировать список подстановок и ввести совершенно другое значение, даже если его нет в списке. Такой подход позволяет

применять список подстановок как удобное экономящее время средство, которое при этом не ограничивает выбор.

В этой ситуации хотелось бы, чтобы список подстановок был так же средством проверки ошибок и верификации, препятствующим вводу посторонних значений.

В этой ситуации нужно выполнить следующие действия:

- В *Конструкторе* перейти в поле, содержащее список подстановок.
- В области *Свойства поля* щелкнуть кнопкой мыши вкладку *Подстановка*.
- На вкладке *Подстановка* представлены параметры для тонкой настройки списка подстановки, большинство из которых легче задать в *Мастере создания подстановки*. В поле *Источник строк*, например, можно откорректировать список предлагаемых значений. (Все значения расположены в одной строке, заключены в кавычки и отделяются друг от друга точкой с запятой.)
- Задать значение *Да* в поле *Ограничиться списком*. Это действие защитит от ввода значений, не включенных в список.
- Можно выбрать значение *Да* в поле *Разрешить изменение списка значений*.

Это действие позволит корректировать значения списка в любое время. Если в списке подстановок что-то пропущено, можно вставить новое значение на лету.