Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный университет
имени Франциска Скорины»

Д. С. КУЗЬМЕНКОВ

LOTUS DOMINO/NOTES

ПРАКТИЧЕСКОЕ РУКОВОДСТВО

по выполнению лабораторных работ

В 2-х частях
Часть 1

Гомель
УО «ГГУ им. Ф. Скорины»
2011

Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный университет
имени Франциска Скорины»

Д. С. КУЗЬМЕНКОВ

LOTUS DOMINO/NOTES

ПРАКТИЧЕСКОЕ РУКОВОДСТВО

по выполнению лабораторных работ

для студентов математического факультета специальностей
1-31 03 03-02 «Прикладная математика (научно-педагогическая деятельность)»
1-31 03 06-01 «Экономическая кибернетика (математические методы
и компьютерное моделирование в экономике)»
1-40 01 01 «Программное обеспечение информационных технологий»

В 2-х частях
Часть 1

Гомель
УО «ГГУ им. Ф. Скорины»
2011

УДК 004.43 (075.8)
ББК 32.973–018.1я73
К 893

Рецензенты:
А.И. Рябченко – заведующий кафедрой информационных технологий учреждения образования «Гомельский государственный технический университет им. П.О. Сухого», кандидат физико-математических наук;
кафедра вычислительной математики и программирования учреждения образования «Гомельский государственный университет им. Ф. Скорины»

Рекомендовано к изданию научно-методическим советом учреждения образования «Гомельский государственный университет им. Ф. Скорины»

Кузьменков, Д. С.
К 893 Lotus Domino/Notes: практическое руководство для студентов математического факультета специальностей 1-31 03 03-02 «Прикладная математика (научно-педагогическая деятельность)», 1-31 03 06-01 «Экономическая кибернетика (математические методы и компьютерное моделирование в экономике)», 1-40 01 01 «Программное обеспечение информационных технологий»: в 2 ч. Ч.1 / Д. С. Кузьменков; М-во образования РБ, Гомельский государственный университет им. Ф. Скорины. – Гомель: ГГУ им. Ф. Скорины, 2011. –48 с.

Практическое руководство содержит описание современной компьютерной системы управления документооборотом Lotus Domino/Notes, приводятся алгоритмы решения наиболее типовых практических задач. Оно адресовано студентам математического факультета специальностей 1-31 03 03-02 «Прикладная математика (научно-педагогическая деятельность)», 1-31 03 06-01 «Экономическая кибернетика (математические методы и компьютерное моделирование в экономике)», 1-40 01 01 «Программное обеспечение информационных технологий» специализации «Компьютерные системы и Internet технологии». Руководство призвано оказать помощь студентам в овладении и закреплении базовых знаний и умений в проектировании приложений в среде Lotus Domino/ Notes.

УДК 004.43 (075.8)
ББК 32.973–018.1я73

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Большинство современных компьютерных систем, в том числе и компьютерных систем управления документооборотом, работают по технологии клиент-сервер. Одной из таких систем и является Lotus Domino/ Notes. Система Lotus является одной из самых современных и мощных систем управления документооборотом. Кроме того, Lotus тесно интегрирован с почтовой системой, Lotus Domino является HTTP сервером, также Lotus – интегрирующая платформа, позволяющая организовать обмен данными с множеством различных систем, функционирующих на базе других платформ. Система ключей, id-файлов и шифрование данных позволяет обеспечить полную безопасность информации в разрабатываемом приложении. Система Lotus позволяет эффективно разрабатывать современные бизнес-приложения. Все вышеперечисленное говорит о необходимости изучения системы Lotus Domino/ Notes специалистами в области программного обеспечения информационных технологий, прикладной математики и информатики.

Практическое руководство составлено в соответствии с учебными программами спецкурса «Lotus Domino/Notes» для студентов 4 курса специальности 1-31 03 03-02 «Прикладная математика (научно-педагогическая деятельность)», 4 курса специальности 1-31 03 06 01 «Экономическая кибернетика (математические методы и компьютерное моделирование в экономике)», 3 курса специальности 1-40 01 01 «Программное обеспечение информационных технологий» специализации «Компьютерные системы и Internet технологии», утвержденными научно-методическим Советом Учреждения образования «Гомельский государственный университет».

Первая часть практического руководства структурно содержит пять тем. В первой теме приводится краткое описание среды Lotus, ее преимущества и недостатки. Вторая тема посвящена работе с элементами структуры (форма, поля, таблицы, разделы). В третьей теме описываются представления – основное средство отображения информации в Lotus. Четвертая тема посвящена организации навигации в базе данных. В последней теме приводится подробное описание языка формул, его основных команд и функций. В конце каждой темы приводится список вопросов для самоконтроля и задание, направленное на закрепление приведенной в теме информации (кроме первой темы).

Практическое руководство по курсу «Lotus Domino/ Notes» направлено на формирование умений и навыков в проектировании приложений в среде Lotus Domino/ Notes, на усвоение современной компьютерной технологии разработки информационных систем.

Практическое руководство может быть использовано преподавателями при проведении практических занятий и студентами в их самостоятельной работе над предметом.

Тема 1 КРАТКОЕ ОПИСАНИЕ LOTUS DOMINO

Lotus – это клиент-серверная система, то есть существует сервер, который хранит информацию и авторизует пользователей, выполняет серверную логику приложений (запускает запрашиваемые клиентским приложением службы); есть клиенты которые выполняют бизнес-логику серверных приложений.

Основная парадигма Lotus – это документарный подход. Если говорить о современных подходах в реляционной технологии, то какой-то аналогией является хранение данных в виде XML, который описывает документ в виде набора «поле-значение» и хранится в одном поле таблицы. Lotus работает не с реляционными данными, его основной объект – это документ. Lotus работает с документами совершенно различной структуры (финансовые отчеты, распоряжения, докладные записки и т.д.). Он позволяет эффективно работать с такими данными.

Слабые стороны Lotus:

1) Из-за документарного подхода Lotus не удобен для построения отчетов; на Lotus очень сложно и долго строить многомерный параметризуемый отчет; но есть связка Lotus – RDBMS, где Lotus используется в качестве системы сбора и обработки информации, RDBM-Storage в качестве системы построения отчета;

2) В Lotus отсутствует понятие транзакции, то есть если вам необходимо выполнить логическую совокупность действий, как одно целое и в случае падения операции в середине все откатить, то у вас это стандартными методами не получится. Поэтому на Lotus не рекомендуется строить финансовые системы, которые рассчитаны не на учет, а именно на перемещение средств.

Достоинства Lotus:

1) **Репликация** – это синхронизация данных между копиями одного и того же приложения.

Пример. Есть компания, у которой куча филиалов в разных городах. В компании функционирует система, в которую вбиваются и в которой согласуются финансовые проводки. Есть центральный офис, в который эти проводки отовсюду поступают и в котором они либо подтверждаются и отправляются в филиалы на выполнение, либо отклоняются.

Как решается задача территориальной распределенности в Lotus?

В каждом филиале ставится сервер domino, на котором в каждом филиале ставится реплика базы. Пользователи каждого филиала работают с базой на своем сервере. А сервера по расписанию, например раз в час, обмениваются всеми поступившими изменениями и новыми документами. При этом парамет-

ры репликации можно задать таким образом, что множество документов поступающих с сервера на сервер будет ограничено – например, в центральный офис будут поступать документы со всех филиалов, а в филиале будут присутствовать только его документы.

При такой схеме, в Гомеле бухгалтер вносит планируемую проводку, в состоянии черновик и отправляет ее на рассмотрение, через час она появляется в главном офисе, там он утверждается или отклоняется и информация об этом изменении еще через час достигает сервера в Гомеле. Можно и не раз в час реплицировать обновления, при кластерной репликации обновления распространяются практически мгновенно.

2) **Накат дизайна**. Дизайн базы – свой бизнес-логики, который хранится в виде специфичных документов. Все обновления дизайна так же реплицируются между серверами. Эта парадигма сильно облегчает задачу перехода к новым версиям. База с новой версией просто объявляется источником дизайна, а база со старой версией дизайна – наследником. После чего запускается накат дизайна, и все обновления поступают в работающее приложение.

3) **Клиент Lotus Notes является облегченной версией сервера**, потому при отсутствии постоянной связи с сервером можно сделать себе локальную реплику базы, работать в ней и отреплицировать их с сервером при появлении связи. Это дает широкие возможности для создания единой информационной системы для компаний, специфика которых подразумевает частые командировки сотрудников.

4) **Защита информации.** При регистрации пользователя сервер создает для него id-файл, в котором хранятся публичные и приватные ключи, а так же электронная почта этого пользователя. Для обращения к серверу пользователь должен авторизоваться, используя этот id-файл. Все изменения, которые пользователь вносит в документы, подписываются им. Даже администратор не сможет эмулировать, что документ был сохранен от имени конкретного пользователя. Весь трафик между серверами и клиент-сервером шифруется. Локальные реплики баз так же могут быть зашифрованы;

5) Lotus тесно интегрирован с **почтовой системой**. Domino сам является SMTP-сервером, который дополнен рядом возможностей для работы с внутренней корреспонденцией. Любой документ внутри Lotus может быть отправлен по почте, как виде письма, так и в виде той формы, по которой он был создан.

6) Domino так же является **HTTP-сервером**. Согласно бизнес-логике приложения сервер генерирует HTML для данных, которые на нем хранятся. Поэтому для большинства приложений возможно реализация как интерфейса в Lotus клиенте, так и через web-браузер;

7) Lotus является **интегрирующей платформой**, то есть позволяет организовывать обмен данными с множеством различных систем, функционирующих на базе другой платформы. Он поддерживает связку OLE-объектами, что позволяет хранить в Lotus-документах объекты Microsoft Office и пользуются всей их функциональностью.

Система управления документоориентированой базой данных

С задачей поиска нужных документов так или иначе связаны 30% перемещений сотрудников по офису, в общей сложности этот процесс отнимает у них около одного месяца в год, причем 15% бумажных документов безвозвратно теряются. На согласование документов уходит 60-70% рабочего времени. В свете вышеуказанного, 20-30% поставленных задач вообще не решаются. Все эти проблемы призвана решить система управления документооборота, организованная с помощью Lotus Notes.

В среде Lotus Notes можно разработать любую базу данных, позволяющую работать с документами. Прежде всего это нефинансовый документооборот. Большая часть документооборота на Lotus Notes признана эквивалентной статусу бумажных документов.

Если отталкиваться от реальной практики использования, то можно выделить 4 основных сферы применения среды Lotus:

1) организации, которым нужна современная, надежная инфраструктура электронной почты, передачи сообщений и коммуникаций;

2) организации, которые используют Lotus в качестве платформы и инфраструктуры для бизнес-приложений, автоматизации деловых процедур, документооборота и.т.д.;

3) организации, выбирающие Lotus Domino в качестве уникальной технологии для создания инфраструктуры Web;

4) организации, выбирающие Lotus в качестве интегрирующего программного обеспечения, способного интегрировать информацию и данные практически из произвольных источников информации – реляционных СУБД, систем управления ресурсами предприятий (ERP), среды Internet и т.д.

Вопросы для самоконтроля

1. Что представляет собой среда *Lotus*?
2. Какие слабые стороны есть у среды *Lotus?*
3. Перечислите достоинства Lotus Domino/Notes?
4. Что такое репликация?
5. Что такое накат дизайна?
6. Перечислите 4 основные сферы применения Lotus Domino/Notes?

Тема 2 РАБОТА С ЭЛЕМЕНТАМИ СТРУКТУРЫ

2.1 Формы

Формы являются одним из основных элементов дизайна приложений Domino. Они служат средствам добавления, редактирования и просмотра документов в БД Lotus. Формы служат механизмом ввода информации в БД Notes, а поля – средствам хранения данных в документах. Поля служат основными строительными блоками форм Notes. Их можно редактировать, вычислять, скрывать или отображать. Форма Notes может содержать множество полей.

Объекты форм Notes

1) Статический текст;
2) Графика;
3) Таблицы;
4) «Горячие» ссылки, могут служить ссылками на документы Notes и web-страницы, показывать всплывающие подсказки, выполнять формулы или сценарии.
5) Кнопки (buttons);
6) Действия (actions), доступны для форм и представлений, могут присутствовать в меню и /или в панели действий;
7) Подчиненные формы (Subforms), могут вставляться в формы в зависимости от формулы или постоянно, являются повторно используемыми элементами проекта;
8) Разделы (Sections), служат средством представления, организации и управления информацией;
9) Области компоновки – специальные области форм, которые позволяют разработчику создавать формы, где легко осуществляется комбинирование текста и графики;
10) Вычисляемый текст;
11) Ресурсы изображений (Image Resource), реализована возможность внедрения общедоступных изображений, хранимых в БД;
12) Внедренные элементы (Embedded Elements). Можно внедрять следующие объекты: схемы, представления, навигаторы и т.д.;
13) HTML код может вводиться непосредственно в формы для отображения в web-браузерах;
14) Java аплеты, могут внедряться непосредственно в формы Notes;
15) JavaScript. Сценарии JavaScript могут внедряться непосредственно в формы, могут применяться для написания обработчиков событий различных объектов в Notes (OnKeyDown, OnKeyPress, OnKeyUp, PostSave, PostSend и.т.д.);

16) Горизонтальные линейки, отображают горизонтальные линии на страницах во время их просмотра в web;

17) OLE-объекты, могут использоваться в формах для ссылки на объекты, созданные в других приложениях.

Типы форм

Тип формы задается в опции type вкладки Form Info (первая вкладка) окна Form Properties. Существует три типа форм, основанных на иерархии документов:

1. Документ.
2. Ответ (response).
3. Ответ на ответ (response of response).

Тип формы «документ» называется главным или главной тематической формой (main topic form). Он находится на вершине иерархии. Главные документы могут иметь несколько связанных с ними ответных документов. Точно так же ответные документы могут иметь несколько связанных с ними документов типа «ответ на ответ». Эта иерархия документов использована в шаблоне БД Discussion и может использоваться в приложениях для сбора информации. В Lotus Domino поддерживается до 32 уровней подчиненности.

Для *создания формы* необходимо сделать следующее:

1) открыть Domino Designer;
2) открыть БД в окне дизайна (Design Pane);
3) в списке Design List выбрать опцию Forms;
4) в рабочей панели щелкнуть на кнопке New Form;
5) открыть окно свойств Form Properties, ввести имя формы;
6) в окне свойств выбрать тип формы и при необходимости установить дополнительные параметры;
7) вставить в форму текст, поля, подчиненные формы, внедренные объекты, графику и т.д.;
8) ввести заголовок окна в нижнем левом окне в свойстве формы Window Title (заголовок указывается в кавычках) и программный код для программируемых событий формы (QuerySave, PostSave и.т.д.).

Текст можно вписать практически в любое место формы. Для выделенного текста можно изменить различные свойства – шрифт, размер, стиль, выравнивание и т.д. (закладки окна Text Properties). На предпоследней закладке (Paragraph Hide When) можно скрыть объект в различных режимах просмотра документа, а также написать на языке формул предикат, который позволяет скрывать документ, если формула истина.

Hide paragraph from (скрыть параграф из):

1) Notes R 4.6 or later;

2) Web browsers;
3) Mobile.

Hide paragraph when document is… (скрыть абзац, когда документ…) :

1) Previewed for reading (предварительный просмотр для чтения);
2) Previewed for editing (предварительный просмотр для редактирования);
3) Opened for reading (открыть для чтения);
4) Opened for editing (открыть для редактирования);
5) Printed (печатается);
6) Copied to the clipboard (копировать в буфер обмена);
7) Embedded (внедряется).

2.2 Предварительный просмотр

Проверка формы (предварительный просмотр) в процессе проектирования очень удобна и позволяет разработчику постепенно достраивать и проверять ее. Перед проверкой изменений их необходимо сохранить. Можно тестировать только формы главных документов, но не формы ответных документов.

Возможна проверка форм в двух версиях:

1) Notes Preview – предварительный просмотр формы в среде клиента Notes;

2) Preview in Web Browser:

а) Preview in Default Browser – предварительный просмотр формы в окне Web-браузера по умолчанию;

б) Preview in Internet Explorer;

в) Preview in Notes Browser – предварительный просмотр в окне Web-браузера Notes.

Предварительные просмотры в Notes выполняются с помощью команды Design → Preview In Notes, Preview In Web Browser или при выборе таких же пунктов меню из контекстного меню формы.

2.3 Работа с таблицами

Таблицы можно вставлять в формы, страницы и поля форматированного текста. Для создания таблицы необходимо щелкнуть на пиктограмме Insert Table или выбрать в основном меню команду Create → Table. При этом откроется окно, содержащее настройки количества строк, столбцов, ширины таблицы и ее типа.

Существует 5 типов таблиц:

1. Основные (Basic);

2. С вкладками (Tabled);
3. Анимационные (Animated);
4. Таблицы заголовков (Caption);
5. Программируемые (Programmed).

Можно создавать вложенные таблицы, для этого следует поместить курсор в одну из ячеек и выполнить команду Create → Table.

У таблицы есть параметры, относящиеся ко всей таблице и параметры, относящиеся к выделенной курсором зоне.

Кратко рассмотрим закладки свойств таблиц.

1) Первая закладка (Table Layout) посвящена позиционированию таблицы на экране и ее размеру по ширине – ее можно вписать в экран (fit to window – по ширине окна) или другую таблицу (fit with margins – настройка по границам), можно задать фиксированную длину (fixed width), состоящую из длин ее столбцов.

Ниже на этой закладке задается ширина ячейки, которая распространяется на весь столбец, к которому относится вся ячейка; высота ячейки, отступ между строками, отступ между столбцами и позиционирование текста в отдельной ячейке.

2) Вторая закладка (Cell Borders) посвящена заданию стиля и цвета разделительных полос.

3) На третьей закладке (Table/Cell Background) задаются параметры графики выделенных ячеек.

4) Четвертая закладка (Table Borders) посвящена параметрам границы таблицы.

5) Пятая закладка (Table Margins) устанавливает обтекание таблицы и параметры отступа таблицы.

6) Шестая закладка(Table Rows) – параметры показа ячеек.

7) Последняя закладка (Table Programming) – возможность задания атрибутов и стилей для html тегов (<tr>, <td>, <table>).

Для работы со столбцами, строками и ячейками можно использовать пункты меню Table (Insert, Append, Delete) или соответствующие пиктограммы.

2.4 Работа с полями

Чтобы создать поле необходимо поставить курсор в то место, где вы хотите его расположить и в меню выбрать Create → Field или пиктограмму  .

Кратко опишем смысл закладок свойств поля.

1) На первой закладке задается тип и вид поля, а так же опции отображения и обхода полей табуляцией.

2) Вторая закладка имеет разный вид в зависимости от того, какой тип и вид поля выбран на первой закладке. На ней будут задаваться параметры отображения чисел для числового формата, дат, параметры задания диалога для

диалоговых окон, опции автоматического обновления документа при изменении значения поля для checkbox и radiobutton и др.

3) На третьей закладке задаются подсказка, опции отображения и ввода списковых значений в поля, а так же опции защиты информации в поле – например, сохранения в поле цифровой подписи или запрещения редактирования данного поля всем персонам в ACL (Access Control List) ниже, чем Editor.

4) Четвертая закладка посвящена параметрам стиля для текста.

5) На пятой закладке задаются параметры стиля для позиционирования поля влево – вправо, интервала между строками и т.д.

6) На шестой закладке определяются параметры скрытия поля (в режиме чтения, редактирования или от какой-либо роли).

7) На последней закладке задаются параметры при генерации html для доступа из web.

В седьмой версии 17 типов данных поля: текстовый, числовой, даты/времени, флажок, переключатель, список и т.д. Типы полей приведены в таблице 1.1.

Таблица 1.1 – Типы полей в Lotus Domino/Notes

Тип поля	Назначение
Editable (редактируемое)	Устанавливается по умолчанию для полей, редактируемых пользователями и хранимых в документе.
Computed(вычисляемое)	Значение задается с помощью формулы; хранится в документе.
Computed for display (вычисляемое для отображения)	Значение задается с помощью формулы; не хранится в документе.
Computed when composed (вычисляемое при составлении)	Значение задается с помощью формулы, когда документ создается вновь; хранится в документе.

Текстовые поля системы Notes отличаются от текстовых полей других баз данных тем, что не имеют фиксированного размера. Они могут хранить до 32 кбайт данных. Текстовые поля не могут применяться в вычислениях (операциях с числами), но функция @ TextToNumber преобразует текстовые значения в числа, которые затем могут участвовать в вычислениях. Текстовые поля также можно конвертировать в даты, с помощью функции @ TextToTime.

Поля даты–времени могут содержать значение даты, времени или то и другое.

Пример. Отобразим дату в формате типа: Monday, April_8, 2007. Для этого необходимы следующие действия:

1. Щелкнуть на вкладке Control.
2. Выбрать значение Custom в поле Use preferences from.

3. В поле Show установить значение all.
4. В поле Format оставить стандартное значение WMDY (день недели, месяц, число, год).
5. Ввести запятую и пробел в первом поле группы Separators (разграничители). Остальные поля оставить без изменения.
6. В поле Month установить значение mm.
7. В поле Day установить значение dd.
8. В поле Year оставить значение yyyy.
9. В поле Week Day установить значение ww.

В разделе Display Time можно настроить формат отображение времени, часовой пояс, 12 или 24 часа и т.д. Для доступа к элементу управления календарем, когда форма находится в режиме редактирования, необходимо щелкнуть по кнопке Calendar рядом с полем.

Числовые поля могут содержать числа, знаки «+» или «-» и экспоненту для научного формата. Рассмотрим вкладку Control для числового поля. Переключатели раздела Number Format задают четыре числовых типа: Decimal, Percent (процентный), Scientific (научный), Currency (денежный). Если в поле Use preferences from выбрана опция Custom, то можно устанавливать десятичный символ и разделитель разряда чисел. Раздел Additional display formatting содержит опции, позволяющие заключать отрицательные числа в ( ) и отделять разряды тысяч. Элементы управления раздела Currency symbol активны, когда в разделе Number Format установлен переключатель Currency, а в поле Use preferences from значение Custom.

2.5 Разделы (секции)

Разделы могут вставляться в документы, страницы, подчиненные формы и формы Notes. В форме разделы представляют собой свертываемые области, которые могут содержать текст, поле, графику и т.д. Разделы очень удобны для форм, которые содержат большой объем данных, т.к. они могут применяться для оптимизации отображения формы путем объединения информации по категориям и ее свертывания.

Существует 2 типа разделов

1. Стандартные (Standard);

2. С управляемым доступом (Controlled Access).

Стандартные разделы просто объединяют текст и поля в свертываемую область формы. Разделы с управляемым доступом могут ограничивать доступ к полям и данным раздела.

Для создания стандартного раздела необходимо выделить область, содержащую текст и поля, а затем выбрать команду меню *Create→Section→Standard*. Для удаления раздела необходимо выбрать команду *Remove Section*. Но она удаляет **только сам раздел, а не содержащиеся в нем поля или текст**.

Рассмотрим основные вкладки окна свойств Section:

1) Вкладка *Section Title and Border* содержит поле заголовка. Можно ввести заголовок в виде текста или использовать формулу. При щелчке на переключателе Formula появится кнопка отображения формул. Для заголовка раздела может использоваться только язык формул. Эта вкладка также содержит настройки стиля и цвета границ раздела.

2) Вкладка *Expand/Collapse* позволяет разработчику вводить правила развертывания и свертывания раздела в зависимости от вида документа. Кроме того, вкладка содержит 2 флажка:

а) *Hide Title When Expanded*. Если этот флажок установлен, то заголовок не отображается при развертывании, но виден когда раздел свернут.

б) *Show As Text When Not Previewing* задает отображение текста в режиме предварительного просмотра. В других режимах содержимое раздела отображается в виде текста, но сам раздел отсутствует.

Для создания управляющего раздела необходимо выделить область, содержащую текст и поля, а затем выбрать команду меню: *Create→Section→Controlled Access.*

Кратко рассмотрим основные вкладки окна свойств раздела.

Первая вкладка позволяет ввести текст заголовка и имя поля раздела

Вторая Вкладка содержит две кнопки: первая позволяет переключаться между опциями Editors и Non Editors; вторая служит для применения созданных для одной группы параметров к другой группе.

Третья вкладка *Formula* позволяет вводить формулы доступа и их тип. Выбор любого значения, кроме Editable требует ввода формулы перед сохранением формы. Если формула доступа содержит значение Editable, то имеющие доступ к разделу пользователи могут изменять список доступа.

Команда меню: *Section→Define Editors* (только для управляющих разделов с формулами доступа Editable) отображает диалоговое окно, которое позволяет указать редакторов раздела. В список редакторов можно включать имена пользователя, группы и роли.

Вопросы для самоконтроля

1. Какие объекты можно добавлять на формы?
2. Какие типы форм используются в Lotus Designer?

3. Порядок создания формы.
4. Где возможен предварительный просмотр формы?
5. Какие типы таблиц существуют в Lotus Designer?
6. Какие типы полей можно использовать в Lotus Domino/Notes?
7. Каковы особенности использования полей различных типов данных?
8. Для чего применяются разделы?
9. Какие типы разделов применяются при проектировании форм?

Задание

Создайте в учебной базе данных форму в соответствии с заданной прикладной областью. Форма должна содержать область заголовка с названием формы и область данных. В области данных должна быть таблица с двумя вкладками: основные данные и история работы с документом. Создайте необходимые поля для ввода данных. Тип полей должен соответствовать типу вводимых данных, так запрещается использовать текстовые поля для числовых данных или даты. Желательно использовать поля таких типов как: ComboBox, ListBox, CheckBox, Radiobutton и.т.д., для размещения на форме полей данных типов необходимо использовать разделы. Поля с обязательной для заполнения информацией необходимо пометить «звездочкой», которая должна отображаться лишь в режиме редактирования.

Добавьте кнопки-действия: Закрыть (видима всегда), Редактировать (видима только в режиме чтения), Сохранить (видима только в режиме редактирования).

Предметные области

1) Страховое агентство;
2) Фирма комиссионной продажи автомобилей;
3) Банк;
4) Отдел сбыта предприятия;
5) Киоск;
6) Научно-исследовательский центр;
7) Книжный магазин;
8) Склад товаров;
9) Общежитие;
10) Аптечный киоск;
11) Кассы ж/д вокзала;
12) Гаражный кооператив;
13) Фирма-продавец компьютерных комплектующих;
14) Магазины города;
15) Медперсонал больниц города.

Тема 3 ПРЕДСТАВЛЕНИЯ

3.1 Создание представлений. Типы представлений

Представления (*view*) – это способы отображения документов для пользователя. В представлении задается:

1) Какое множество документов отображать (множество документов задается языком формул как некоторый предикат на множестве всех документов);

2) Какие поля документов отображать в списке документов, каким образом сортировать и т.д.

Представление состоит из трех основных областей: панели Navigation, панели View и необязательной панели Preview. Только панели Navigation и View содержат элементы, которые может изменять разработчик. Панель Preview только отображает содержимое документов в режиме предварительного просмотра.

Каждое представление должно иметь хотя бы один столбец. Столбцы определяют содержимое строк. Формулы отбора определяют документы, отображаемые в представлении. Формулу можно задать при создании представления, затем изменить ее в свойствах *view selection* в нижнем левом окне. Стандартная формула отбора Select@All отображает все документы базы данных. Ключевое слово Form задает, какие формы должны использоваться для отображения документов в представлении.

Пример. *select form = "Main_form" & year="2011"* – выбирает все документы, созданные по форме Main_form, в поле year которых содержится "2011".

В свойстве *view selection* можно задать условие выбора (которое равносильно некоторой формуле). Для этого необходимо

1) Открыть представление в режиме конструктора. В свойстве *view selection* в нижнем левом окне выбрать опцию Simple Search.

2) Щелкнуть на Add Condition и выбрать в поле со списком Condition опцию By Form.

3) В появившемся окне выбрать форму, которая будет использоваться.

Для просмотра формулы нужно щелкнуть на переключателе Formula, после чего увидим: *Select((Form="Main_form"))*.

Перед созданием любого представления необходимо продумать план будущего представления, например, какие документы будут отображаться, порядок сортировки и разбивки по категориям документов, какие поля (элементы) должны выводиться в столбцах и т.д. Можно создать, так называемый, контрольный список проекта представления, который позже станет частью документации дизайна.

Для создания представления в дизайнере перейдем в пункт *Views* и нажмем кнопку *New View*; в появившемся окне создания представления задаются начальные параметры: название, формула, по которой отбираются документы, здесь можно скопировать стиль уже существующего представления, а также задается его тип.

Типы представлений

1. *Shared* (общие) – это представление, которое все видят одинаково. В настройках доступа можно отрегулировать группы и роли, которые имеют и не имеют доступ к этому представлению. К тому же, в зависимости от прав доступа, разные пользователи будут видеть в этом представлении разное количество и список документов. Но в таких представлениях нельзя использовать колонки для подсчета, например, суммарных значений по разным категориям. Даже если пользователю не доступна половина финансовых документов, в таком представлении он увидит суммарное значение по всем документам. Это является основным минусом таких представлений.

2. *Shared, contains documents not in any folder* – такое представление на множестве документов, которые задаются формулой выбора, задает множество тех из них, которые не лежат ни в одной из папок.

3. *Shared, contains deleted documents* – если в базе разрешено *soft deletion* (задается на последней закладке свойств базы данных), то в такое представление будут попадать документы, которые были удалены и их оттуда можно восстановить.

4. *Shared, private on first use* – это общее представление, которое станет «приватным», т.е. видимым и принадлежащим только конкретному пользователю после того, как он в первый раз откроет это представление. При этом создастся копия этого представления, принадлежащая пользователю. Такие представления удобно использовать для решения задач показа, например, документов, созданных пользователем. Место хранения приватного варианта представления определяется самим пользователем при его создании.

5. *Shared, desktop private on first use* – аналог предыдущего типа, только представление будет храниться на рабочем столе пользователя (в файле desktop.dsk)

6. *Private* – это представление, доступное только самому пользователю. Он сам его создает, если не хватает стандартных представлений базы. При наличии прав в ACL на создание приватных представлений, такое представление будет храниться в базе на сервере. Если прав нет – то в файле desktop.dsk.

3.2 Свойства представления

Для открытия свойств выполним команду меню *Design→View Properties*. На первой закладке указывается название представления, его alias, комментарии, а также основной вид представления: **стандартное** представление, в котором документы упорядочены линейно; и **календарное**, когда отображается календарь с датами и каждый документ привязан к какой-то дате. Если указать стиль представления – календарь, то в списке свойств появятся настройки отображения календаря.

Рассмотрим основные флажки второй закладки окна свойств представления:

– *Default when database is first opened*. Объявляет представление первым открываемым при открытии базы данных. Если не задан frameset, и в нем не указано открытие другого представления, то будет открываться именно это.

– *Default design for new folders and views*. Эта опция объявляет, что по умолчанию дизайн вновь создаваемого представления наследуется из этого представления.

– *Collapse all when database is first opened*. Опция включает свертывание всех категорий представления при первом открытии базы. Эту опцию нельзя включить для представлений, которые используются в диалогах выбора, и являются категоризированными (если в диалоге выбора включить опцию выбора из конкретной категории, а категории при этом свернуты, то в окне выбора не будет документов).

– *Show response documents in a hierarchy*. Опция используется для представлений, в которых отображаются не только документы, но и документы ответов.

– *Show in View menu*. Показывает представление в меню клиента view.

В нижней части второй закладки находятся две опции, определяющие, куда ставить курсор при открытии и что показывать для обновления.

На третьей закладке свойств задается цветовая гамма представления и стили. В разделе Other данной закладки есть несколько флажков:

– *Extend last column to window width* – последняя колонка визуально растягивается до конца экрана.

– *Show selection margin* – показывает слева от всех колонок полосы, в которых документы можно пометить галочками и они становятся выбранными.

Четвертая закладка показывает представление в отдельном множестве фреймов, пятая посвящена параметрам доступа через web, шестая – опциям, кому показывать представление.

3.3 Отображение информации в колонках представлений

Первоначально представление имеет одну колонку. Чтобы отредактировать список колонок представления, надо щелкнуть правой кнопкой мыши на колонке – там есть опции вставки колонки, добавления, удаления; или щелкнуть на соответствующей пиктограмме панели пиктограмм.

Пример. Создадим колонки «ФИО» и «контактная информация».

Теперь в созданных колонках укажем, что в каждой будет отображаться. Для этого поставим курсор на колонку, и в нижнем левом окне загрузится ее значение. В значении пишется формула, которая будет подсчитана на каждом документе, и будет выдавать некоторое значение. Значение может быть строкой, числом или датой. Существуют колонки специального вида, отображающие иконки и картинки.

В первой колонке напишем формулу:

firstname +" "; + @if (@Trim (middlename)!=""; middlename+ " "; "") +
+lastname

Функция @Trim удаляет лишние пробелы из начала и конца строки или из каждого элемента текстового списка.

В колонке представления можно отображать поле (или выбрав его из списка при включенном переключателе *Field,* или назвав его по имени при включенном переключателе *Formula* окна *Column Value*) или результат выполнения формул на языке формул (переключатель *Formula* окна *Column Value*). Есть также набор *Simple Function* (одноименный переключатель), который выдает на документе некоторые его атрибуты, такие как дата создания, дата модификации, размер и т.д.

На первой закладке свойств колонки представления указывается:

– заголовок колонки;

– ее ширина;

– *multi-value separator* – если в колонке отображается поле со списком значений – то в этом свойстве можно указать, что использовать при отображении в качестве разделителя элементов списка;

– *resizable* – разрешение пользователю изменять размер колонки;

– *show responses only* – опция, используемая в иерархических представлениях. Она включает отображение в колонке именно документов-ответов;

– *display values as icons* – отображение значений иконками. Эта опция позволяет отображать встроенные в Lotus иконки в строках представления. Иконки имеют номера и при подстановке в формулу номера иконки и включении этой опции, в колонке будет отображаться иконка.

– *do not display title in column header* – не отображать текст заголовка колонки;

– ***show twistie when row is expandable*** – показ свертки/развертки для раскрывающейся категории. При включении на второй закладке опции категоризации по колонке, эта опция включает “уголки”.

На второй закладке свойств колонки задаются сортировки (*Ascending* – по возрастанию, *Descending* – по убыванию; *Secondary sort column* – столбец, по значениям которого будут отсортированы документы, если значения поля, отображаемого в данном столбце, одинаковы у нескольких документов), категоризации, параметры суммирования и вычисления и т.д.

На закладках 3-5 свойств колонки задаются свойства отображения данных в этой колонке для различных типов этих данных, а на последней закладке хранится информация об уникальном имени этой колонки и параметры генерации ссылок на документы из значений этой колонки при просмотре через web.

При копировании и вставке колонки иногда Lotus сам не уникализирует имя, поэтому необходимо изменять его вручную.

3.4 Добавление действий к представлению

Для отображения действий необходимо щелкнуть по кнопке s*how / Hide Action Pane*, перетащить панель *Action* в представление мышью, или выбрать команду *View → Action Pane*.

Существует 6 стандартных действий:

1. *categorize* – распределить по категориям;
2. *edit document*– редактировать документ;
3. *send document* – отправить документ;
4. *forward* – отправить;
5. *move to folder* – переместить в папку;
6. *remove from folder* – удалить из папки.

Эти действия можно отключать, но не удалять из панели. Для отключения действия необходимо сделать следующее:

1. Дважды щелкнуть на действии (например, Categorize), появится окно свойств действия.

2. В центре вкладки *Action Info* находятся 3 флажка: *Include action in Action menu*, *Include action in Action bar*, *Include in right mouse button menu*. Необходимо отменить установку всех флажков и действие будет отключено.

Выберем команду *Create Action* в контекстном меню, откроется окно свойств, в котором можно присвоить действию имя, выбрать его тип (*Button,*

CheckBox, Menu Separator), задать настройки отображения действия и выбрать для него иконку. Действие может быть простым (*Simple Action*), представлять собой формулу, составленную из набора команд @Commands или функций языка формул, либо сценарии Lotus Script, Java Script, Common Java Script.

Если при задании команд форма не определена, то отображается окно доступных форм, в котором пользователь может сделать выбор. Если для пользователя должна быть открыта определенная форма, то необходимо указать ее имя в формуле. Лучше указать не полное имя формы, а ее псевдоним. Например, если «Reguest for Information» – полное имя формы, а ее псевдоним «RI», то формула должна иметь следующий вид: *@ Command([compose]; "RI")*.

Чтобы действие отображалось в панели кнопок необходимо установить флажок *Include Action in action bar*. Чтобы действие можно было вызвать из других представлений, необходимо установить флажок *Share this Action* (общедоступное действие). При этом действие будет невозможно модифицировать из текущего представления. Для изменения общедоступного действия необходимо открыть окно *shared Actions* с помощью меню *Resources*.

В отличие от действий форм, обладающих предварительно установленными условиями скрытия кнопок Action Bar, кнопки действий представлений могут скрываться только с помощью языка формул. Для этого необходимо установить флажок *Hide Action If Formula Is True* и ввести формулу

Окно свойств панели Action доступно только тогда, когда панель Action видима. Для доступа к окну можно выбрать команду *Design → Action Bar Properties*, или выбрать опцию *Action Bar* из поля со списком окна свойств.

На первой вкладке поле со списком *Alignment* позволяет задать порядок размещения кнопок. Настройки Web Access позволяют определить, будет ли использоваться Java апплет для отображения представлений для пользователей Web. Если апплет применяется, то Web-клиенты могут прокручивать панель Action и использовать поля со списком. Панель Action можно отображать также посредством стандартного кода HTML.

Вторая вкладка определяет размер панели Action. На третьей закладке можно изменить стандартный серый цвет фона, или загрузить картинку. На четвертой закладке задаются параметры границы панели действий: цвет, толщина границы и т.д. Пятая закладка управляет фоном кнопки: задается размер кнопки, когда отображается ее граница, применение фонового изображения или цвета. На последней закладке задается шрифт надписей кнопок.

Вопросы для самоконтроля

1. Что такое представление и зачем оно применяется?
2. Где и как задается условие выбора документов для представления?

3. Какие типы представлений можно создавать в Lotus Designer?
4. Перечислите основные свойства представления?
5. Какие существуют особенности отображения информации в колонках представления?
6. Что такое *Twistie* и зачем он используется?
7. Какие стандартные (системные) действия можно задавать в Action Pane?
8. Какие существуют особенности добавления действий к представлениям?

Задание

Создайте в учебной базе данных два представления: общий список и список по категориям. Представления должны отображать основные поля документов и иметь возможность сортировки столбцов. При создании представлений необходимо применять различные свойства представления и столбцов: шрифты, фоновый цвет, графику и т.д.

Добавьте к каждому представлению кнопки-действия: Новый документ, Редактировать, Удалить.

Тема 4 ОРГАНИЗАЦИЯ НАВИГАЦИИ В БАЗЕ ДАННЫХ

4.1 Схемы навигации (Outlines)

Схема навигации (Outline) – это элемент дизайна, который содержит структурную информацию для обеспечения навигации по базе данных. Outline позволяет задать удобное для пользователя расположение ссылок в представлении.

4.1.1 Создание схемы навигации

Схемы (схемы навигации) служат средством навигации по узлам и базам данных. Схема выглядит как гибрид навигатора и панели папок вместе взятых. Схемы могут содержать ссылки на представления, формы, наборы фреймов, другие базы данных узла и т.д.

Схема позволяет добавлять к записи картинку. Схемы могут быть иерархическими, а записи могут раскрываться и свертываться (как категории в представлении). Схема может внедряться как апплет, в страницу или форму. База данных может содержать несколько схем.

Схема состоит из одной или нескольких записей. Каждая запись может служить ссылкой на объект базы данных или URL-адрес, а также запускать действие, запрограммированное на языке формул. После создания схемы ее можно внедрить в форму, страницу или документ (чаще всего схемы используются в страницах). Когда схема вставлена в страницу, ее можно представить пользователям путем помещения страницы в набор фреймов.

Так как схемы описывают структуру приложения или сайта, то существует два метода работы с ними:

1. Можно создать схему до построения всех остальных элементов, а затем добавить ссылки или действия по мере разработки остальных составляющих приложения.
2. Можно добавить схему к базе данных с элементами, которые уже существуют.

Для создания схемы необходимо выбрать базу данных в панели Design в Domino Designer, затем выбрать *Shared Code→Outlines* в списке Design и щелкнуть на кнопке *New Outline*.

Рассмотрим кнопки панели Design Action:

1. *New Entry* (Новая запись) – создает новую запись, которая следует за выделенной записью схемы;

2. *Save Outline* – сохранение схемы;

3. *Use Outline* – внедрение схемы в страницу (для просмотра схемы можно пользоваться режимом предварительного просмотра страницы). Эту страницу можно присоединить к набору фреймов;

4. *Generate Default Outline* – создание схемы для представлений и папок, уже существующих в базе данных. В ней допускается переименование вновь созданных записей, добавление, удаление и перемещение записей. Все ссылки создаются автоматически. При внесении изменений в структуру представления схема не будет автоматически обновлена. Для этого необходимо открыть схему и снова щелкнуть на кнопке Generate Default Outline, но при этом будут отменены все внесенные в схему изменения;

5. *Indent Entry* (сместить запись во внутрь) – перемещает запись вправо относительно записи, находящейся уровнем выше. Служит для построения иерархических схем. В иерархической схеме записи с отступом могут раскрываться и свертываться (иерархические схемы функционируют во многом подобно иерархическим представлениям). Если иерархическая схема внедрена в страницу или форму, то вложенные записи могут раскрываться и свертываться щелчком на маркерах развертываемых строк (twisties);

6. *Outdent Entry* (сместить запись наружу) – перемещает запись влево.

После создания схемы достаточно добавить к ней записи, отражающие структуру приложения или узла.

4.1.2 Работа с записями схемы

Записи выполняют действия, служат ссылками на другие БД или web-сайты, открывают представления и папки и выполняют много других функций. Окно свойств записи содержит вкладки *Entry Info* и *Entry Hide When*. Вкладка Entry Info позволяет задать имя, содержащее изображение и опцию сохранения фокуса вкладки. Раздел Content содержит поля Type, Value, Frame, а также ряд кнопок и поле со списком. В зависимости от значения поля Type элементы раздела Content изменяются. Существует 5 опций поля *Type*:

1) *None* – может быть использовано как заполнитель, например, как заголовок категории.

2) *Action* – создает действие на базе набора @Commands и языка формул.

3) *Link* – вставляет в запись ссылку, ранее скопированную в буфер обмена.

4) *Named Element* – список, позволяющий выбрать один из следующих объектов БД:

– страница;

– форма;

– множество фреймов;

– представление;

– папка;

– навигатор.

5) *URL* – ввод допустимого URL-адреса в качестве ссылки.

Тип *None* применяется как запись, содержащая подчиненные записи.

Кнопка *Browse* доступна только для типа *named element* и позволяет выводить список элементов проекта. Кнопка *Formula* позволяет открыть окно Formula и доступна для 2,4,5 типов.

Запись типа action предоставляет доступ к языку формул и набору команд @Commands. Действия могут использоваться для запуска программы-агента.

Для создания нового документа из записи можно обойтись без использования действия. Можно выбрать тип *named element* и выбрать опцию *Form*, при помощи кнопки browse найти нужную форму. Присвоим записи имя. Когда пользователь щелкнет на записи, будет создан новый документ на базе указанной формы.

Чтобы создать запись типа *link*, нужно скопировать ссылку (на документ, представление, или БД) в буфер обмена в среде notes, установить тип *link* и нажать клавишу *paste*. В поле *Value* отображается ссылка.

Для добавления изображения к записи необходимо сначала добавить его в набор *image resources*, затем выбрать картинку с помощью кнопки *Browse Images* или путем ввода формулы. Результатом вычисления формулы должно быть имя ресурса изображения.

Флажок *do not display an image* позволяет отключить отображение картинки. Этот флажок в основном служит для отмены вывода стандартных изображений, которые предусмотрены в Domino для различных элементов.

Отключение фокуса выделения выполняется в разделе Options, находящемся в нижней части вкладки *info* окна свойств записи схемы. При этом фокус выделения остается на предыдущей записи схемы. Этот эффект незаметен, если не изменен цвет выделения на вкладке *Font* окна свойств *Embedded Outline*. Данная опция удобна для записи, которая создает документ или выполняет какие-либо другие действия.

4.1.3 Внедрение схем. Вкладка info окна Embedded Outline

Схему можно внедрить в страницу, щелкнув по кнопке *use outline* рабочей панели Outline, или можно вставить схему в готовую страницу, форму или документ командой меню *create→embedded element→outline.* Чаще всего схема внедряется в страницу.

Страницы (Pages) – элементы дизайна, используемые для отображения информации, но не для ввода. Как следствие есть ограничение по сравнению с формами – **на странице нет полей**, все остальные элементы, свойственные формам (таблицы, секции, кнопки, встроенные объекты, ссылки и т.д.) можно размещать на страницах.

После внедрения для изменения объекта *outline* будут использоваться свойства внедренной схемы (*embedded outline*). Окно свойств внедренной схемы состоит из 7 вкладок.

Свойства первой вкладки *info* управляют отображением схемы. Вкладка info содержит 5 разделов: Первый раздел позволяет присвоить объекту имя, тип, заголовок и фрейм назначения, загрузить картинку. Во втором указывается корневая запись, а также режим отображения объекта: *Display as Saved* (отображать, как было при сохранении), *Display as Expanded* (отображать раскрытой), *Display as Collapsed* (отображать свернутой). Третий раздел позволяет устанавливать размеры внедренной схемы, а четвертый – режим доступа к WEB, пятый – режим отображения непрочитанной информации.

Поле *type* содержит две опции:

1) *tree style* – древовидный стиль;
2) *flat style* – плоский стиль.

В режиме *flat style* отображается только верхний уровень иерархии. При щелчке на раскрываемой записи под ней отображается следующий уровень. Возможно как вертикальное, так и плоское горизонтальное отображение. Это свойство устанавливается с помощью поля со списком, которое появляется, когда выбрана опция *tree style*.

В поле *target frame* указывается фрейм, в котором будет отражена схема, если она используется в наборе фреймов.

Раздел *root entry* (корневая запись) содержит поле ввода записи для категорий схемы. Следует использовать псевдоним записи, а не ее надпись. Если указана корневая запись, то отображаются только записи данного и более низкого уровней. Поле со списком, находящееся справа, позволяет задать режим отображения иерархической схемы. Для *flat style* в этом поле по умолчанию ус-

танавливается *collapse all* (свернуть все). Для режима *tree style* можно выбрать одну из следующих опций:

1) *expand all* – раскрывает все записи на всех уровнях;
2) *expand first* – раскрывает только записи первого уровня;
3) *display as saved* – схема отображается в таком виде, как было при сохранении;
4) *collapse all* – свертывает схему до самых верхних записей иерархии.

4.1.3 Вкладки Font, Background и Layout окна Embedded Outline

Вкладка *Font* позволяет устанавливать шрифт для трех различных уровней: *Title* (заголовок), *Top level* (верхний уровень), *Sub level* (следующий уровень). Вкладка Font содержит стандартные настройки шрифта, размера и стиля. В нижней части окна находятся дополнительные средства настройки цветового выделения текста. Для уровня *Title* возможна настройка цвета для двух состояний: *Normal* (обычный) и *Moused* (при установке курсора мыши). Поскольку заголовок не может быть выделен, то состояние *Selected* (выделенный) не отображается. Для *Top level* и *Sub level* возможна настройка цветов всех трех состояний.

При щелчке на каждом из этих трех полей открывается цветовая палитра Domino. Поле *Normal* задает цвет текста, который не выделен и на нем пока не установлен курсор мыши. После щелчка на записи схемы ее цвет изменяется в соответствии с установкой поля *Selected*. При наведении на текст курсора мыши используется настройка *Moused*. Эти свойства не проявляются в Web, если в поле *Web Access* не выбрана опция *Using Java Applet*.

Элементы управления свойств заголовка отображаются только в том случае, когда во вкладке *Info* в поле *Title style* выбрана опция *Simple*.

Вкладка *Background* содержит элементы настройки цвета и фонового изображения. Эти свойства можно установить для элемента управления в целом, заголовка, записей верхнего уровня, записей следующих уровней. В зависимости от элемента можно выбрать цвет фона для трех состояний: *Normal*, *Moused*, *Selected*. Для фона элемента управления доступна лишь одна опция цвета.

Допускается добавление фонового изображения к любому уровню. Для этого изображение должно быть сохранено как *image resources*. После добавления изображения к уровню можно установить для него один из следующих режимов отображения:

1) *repeat once* – картинка отображается в единственном экземпляре;

2) *repeat vertically* – копии изображения располагаются в вертикальном ряду до заполнения области отображения;

3) *repeat horizontally* – копии изображения располагаются в горизонтальном ряду до заполнения области отображения;

4) *repeat both ways* –оба варианта (2, 3);

5) *size to fit* (привязка к размеру) – изменяется размер картинки, чтобы она заполнила область отображения.

Настройки вкладки *Layout* определяют местоположение записи, заголовок записи и связанной с ней изображение. Для уровня *title* доступно 2 раздела: *Entry* (запись) и *Entry Label* (заголовок записи). Для *Top level* и *Sub level* кроме вышеперечисленных разделов доступен раздел *Entry Image*.

Каждый раздел содержит 2 типа настроек: *alignment* (выравнивание) и *offset* (смещение).

В разделе *Entry* содержится только один параметр выравнивания – *Height*, т.к. этот раздел управляет расположением записи в целом как блока. Поле со списком этого раздела содержит опции *fit to content* (по размеру содержимого) и *fixed* (фиксированная). Во всех трех разделах имеются 2 поля *Offset* – для вертикального и горизонтального смещения в дюймах записи, ее заголовка и изображения. Как указывают стрелки рядом с полями, при вертикальном смещении объект сдвигается вниз, а при горизонтальном – вправо.

4.2 Наборы фреймов (Framesets)

Наборы фреймов (Framesets, фреймсеты) – это элементы дизайна, задающие логику расположения окон, в которых отображаются навигация, заголовки, представления, формы и другие элементы дизайна.

Для создания набора фреймов необходимо:

1. В дизайнере перейдем в пункт *Framesets* и нажмем кнопку *New Framesets*;

2. В появившемся окне создания множества фреймов выбираем число фреймов (2–4) и их расположение, нажимаем клавишу OK;

3. Для созданного множества фреймов необходимо указать имя, алиас;

4. Необходимо указать имена каждого из фреймов и выбрать их содержимое.

Для добавления схемы к набору фреймов необходимо

1) Выбрать фрейм для страницы;

2) Открыть окно свойств фрейма и присвоить ему имя;

3) В поле *Type* установить значение *Named Element*. В качестве *Named Element* справа выберем *Page*, а затем щелкнем на кнопке *Browse*;

4) в диалоговом окне *Locate object* выберем страницу со внедренной схемой, щелкнем на кнопке OK;

5) присвоим набору фреймов имя и сохраним его.

Страница будет добавлена к набору фреймов, и будет отображаться для клиентов web и notes.

Структуру уже созданного набора фреймов можно изменить с помощью кнопок, расположенных над проектируемым множеством фреймов:

- *Split into Columns* – текущий фрейм разбивается на два фрейма, равных по ширине;
- *Split into Rows* – текущий фрейм разбивается на два фрейма, равных по высоте;
- *Delete Frame* – текущий фрейм удаляется;
- *Remove Frame Contents* – удаляется содержимое текущего фрейма;
- *Flip Horizontally* – меняются местами два рядом расположенных фрейма, находящиеся на одной высоте.

Для того чтобы, при выборе элемента в одном фрейме, он отображался во втором фрейме, а не в новом окне, необходимо на вкладке *Basics* первого фрейма в свойстве *Default target for links in frame* указать имя второго фрейма, например *Right_frame*.

4.3 Создание общего изображения

В проекте базы данных часто повторно используются изображения (например, логотип компании). Начиная с пятой версии Lotus, путем создания ресурса изображений графический файл хранится в базе данных, что позволяет избегать вызова файлов изображений извне файловой системы. Ресурсы изображений используются в страницах, кнопках действия, формах и элементах схем. Их также можно помещать в действия, ячейки таблиц, страницы и формы в количестве фоновых иллюстраций.

Общее изображение создается путем сохранения файла в формате *.gif* или *.jpg* в базе данных. При сохранении в качестве файла общего изображения файла с расширением *.bmp*, он будет автоматически преобразован в файл с расширением *.gif*.

Для создания общего изображения обратимся к подразделу *Images* раздела *Shared Resources* в окне проекта и нажмем *New Image Resource* для добавления общего изображения. После выбора и выделения графических файлов общее изображение создается и вносится в список. Затем можно установить свойства изображения.

На первой закладке окна свойств изображения можно задать имя файла, псевдоним и комментарий. Раздел *Advanced* позволяет описать набор изображений, который представляет собой графический файл, содержащий до четырех изображений. Каждое из них может отображаться в зависимости от состояния элемента управления. Если выбран горизонтальный набор изображений (устанавливаем количество изображений в поле *Image across)*, то первое из них будет отображаться в обычном состоянии, а каждое из трех остальных – в зависимости от события установки курсора мыши, выделения, выполнения щелчка мышью.

Окно свойств *Image Resource* позволяет установить свойства запрета замены или обновления элемента, а также распространения элемента. Дополнительные параметры изображения определяются элементом проекта, в который будет помещено данное изображение. Если изображение помещено в форму, страницу или подчиненную форму, то его свойства становятся доступными для внедренного изображения.

На вкладке *Info* задается источник изображения, его масштаб, обтекание текстом, подпись картинки и указатель заменяющий текст (он выводится вместо картинки, когда она не отображается). На вкладке *Picture Boarder* задается тип и цвет рамки, тень и толщина каждой стороны рамки.

Вопросы для самоконтроля

1. Что представляет собой *Outline*?
2. Какие кнопки расположены на панели *Design Action* при создании схемы?
3. Перечислите значения поля *Type* вкладки *Entry Info* окна свойства записи схемы?
4. Какие объекты базы данных можно добавлять по имени к записи схемы?
5. Что представляют собой страницы? В чем их отличие от форм?
6. Перечислите основные свойства вкладки *Info* окна *Embedded Outline*?
7. Назовите 3 уровня записей схем, для которых можно по-разному установить настройки шрифта?
8. Какие 3 состояния могут быть у записей схемы?
9. Что представляют собой наборы фреймов?

10. Как добавить схему к набору фреймов?
11. Что такое общие изображения и зачем они создаются?

Задание

Создайте в учебной базе данных дерево выбора для своих представлений (схему навигации), разместите его на странице. Создайте набор фреймов, содержащий название базы данных, страницу со списком представлений (дерево выбора) и выбранное представление. При выборе представления в левом фрейме, оно должно открываться в правом фрейме, а не в новом окне.

Добавьте изображение в общие ресурсы и используйте его в дизайне базы данных.

Тема 5 ЯЗЫК ФОРМУЛ

5.1 Синтаксис языка формул. Константы и операторы

Поля в форме обрабатываются слева направо и сверху вниз. Таким же образом формулы всегда вычисляются слева направо и сверху вниз (исключения смотрите в разделе 5.2). Формулы могут содержать константы, поля, ключевые слова, операторы, функции, команды и переменные.

Работа с константами

Существует 3 типа констант: 1) Числовые; 2) Текстовые; 3)Даты/ Времени.

Текстовые константы всегда заключаются в кавычки, числовые вводятся обычным образом, а константы даты/времени заключаются в квадратные скобки.

Пример. ODateTime:=[08/14/08 11:30 PM]. Сначала указывается **месяц**, затем число.

Операторы

Существует 6 основных типов операторов: арифметические, присвоения, сравнения, конкатенации (сцепления) списка, логические и унитарные (положительные и отрицательные). Между двумя переменными обязательно должен стоять оператор.

Арифметические операторы (+,-,*,/) реализуют соответствующие математические функции.

Операции сравнения: =, >, <, <> или ><, <= или =<, >= или =>. В результате их применения возвращаются логические величины. Т.к. логического типа переменных в языке формул нет, то число 1 заменяет истину, а 0 – ложь.

Оператор присваивания (:=) присваивает значение переменной или полю. Область действия переменной ограничивается временем выполнения формулы. Даже если в форме несколько полей и события во всех полях содержат формулы, используются переменные с одинаковыми именами, то значение переменной остается в той формуле, к которой эта переменная относится. Изменение ее значений в одном поле никак не повлияет на значение переменной в другом поле. Можно использовать переменную, одноименную с полем, и если поле имеет значение, то и переменная будет иметь такое значение. Затем можно изменить значение переменной с помощью формулы. Но пока вы явно не сохраните величину, используя ключевое слово FIELD или функцию @SetField(), значение

переменной будет возвращено в исходное состояние сразу же после того, как формула завершит вычисления, даже если имена переменной и поля совпадают.

Обычно в качестве имени переменной используют j и имя поля, например, jcSubject.

Списки конкатенированы с помощью двоеточия (:).

Пример. Создадим 2 списка cTextList и dDateList:

cTextList:=”Item One”: ”Item Two”: ”Item Three”;
dDateList:=[10/26/08]: [12/13/08]”: @Today;

Существует 3 логических оператора: **1) И – &, 2) Не – !, 3) Или – |.**

Унитарными операциями являются знаки плюса и минуса. Они изменяют знак числа.

Использование точки с запятой, регистра и круглых скобок

Пробелы необходимы только после ключевых слов. Каждый оператор формулы должен завершаться «;» (кроме последнего). Все аргументы в функциях и командах также разделяются «;».

В большинстве случаев формулы не чувствительны к смене регистра, но регистр важен для текстовых констант, т.к. **TEXT, Text, text** – разные константы. Ключевые слова всегда вводят с помощью заглавных букв (*так принято, но можно и строчными буквами*).

Используя () можно изменить приоритет выполнения логических и арифметических операторов.

Пример. *SELECT Form=”Main_form”&! (dDate1=““ & dDate2= ““ & dDate3= ““)* – будут отображены все документы, созданные с помощью формы Main_form, содержащие поля dDate1, dDate2, dDate3, за исключение тех, где все 3 поля даты не заполнены.

Пример. *SELECT Form=”Main_form” &!dDate1=““ & dDate2=““ & dDate3=““* – будут отображены документы, составленные с помощью формы Main_form с заполненным полем dDate1 и пустыми полями dDate2, dDate3.

5.2 Работа с @Function, @Commands

Язык формул содержит 3 основные конструкции – **@Function, @Commands**, и 5 ключевых слов (**REM, SELECT, FIELD, DEFAULT, ENVIRONMENT**).

В языке формул не предусмотрена возможность создания циклов, нет возможности вызвать одну формулу из другой – т.е. не предусмотрено ветвление, оно возможно в пределах одной формулы. Если что-то невозможно реализовать с помощью языка формул, то можно обратиться к языкам Lotus Script,

Java Script или Java. Но процесс кодирования средствами языка формул почти всегда выполняется быстрее, чем эквивалентный процесс, запрограммированный с помощью языка Lotus Script.

Функции **@Function** и команды **@Commands** используются для создания формул, которые бы возвращали результат или выполняли действие. Функции **@Functions** всегда возвращают результат. Команды **@Commands** действуют только в пользовательском интерфейсе и связаны с командным меню. Например, команда **@Command([FileCloseWindow])** – закрывает текущее окно, а команда @**Command([EditDocument])** открывает документ в режиме редактирования. Эквивалентные этим командам пункты меню: *File →Close; Actions → Edit*. Команды @Commands не обязательно должны возвращать какое-либо значение.

За некоторыми исключениями (например, команда **@Command([FileCloseWindow])** всегда выполняется в последнюю очередь), команды **@Commands** выполняются в том порядке, в котором они заданы в формуле.

Синтаксис функции @Function:

@Functions(arguments).

Но есть функции, не требующие наличия аргумента, например, **@All (select @All –** возвращает все документы БД).

Команда: **@SetField ("Noom_koop", "12") –** присваивает полю Noom_koop значение 12.

Синтаксис команд @Commands:

@Command([Keyword];arguments);

Команды могут использоваться в действиях, пиктограммах, при работе с кнопками и «горячими» ссылками. Их можно применять в программах-агентах, обрабатывающих текущий документ. Но команды невозможно применить в программах – агентах, запускаемых по расписанию, в формулах полей и других объектах, которые не взаимодействуют с пользователем.

Пример. Формула кнопки-действия, которая сохраняет и закрывает документ:

@Command([FileSave]);
@Command([FileCloseWindow]);

Многие команды не работают в Web–приложениях, так как они приспособлены к особенностям Lotus и не имеют своих аналогов в браузерах. Но есть несколько команд, которые работают в Web [7, с. 407, табл.18.2]. Например:

@Command([NavigateNext]) можно применить при работе с кнопкой действием формы, тогда пользователь сможет переходить к следующему документу.

5.3 Работа с ключевыми словами

Ключевые слова – это специфические функции, используемые в формулах. Они записываются заглавными буквами. Ключевые слова должны быть перечислены в строке.

1) **DEFAULT** – если поле не существует, то DEFAULT создает временный экземпляр и присваивает ему значение; если поле существует, то в формуле используется текущее значение этого поля.

2) **ENVIRONMENT** – присваивает значение переменной среды, хранимой в поле настроек пользователя в файле Notes.ini.

3) **FIELD** – присваивает значение полю, создавая поле, если оно не существует.

4) **REM** – используется для создания комментария в формулах.

5) **SELECT** – используется в формулах выделения представления, формулах репликации и программах – агентах, чтобы определить включен ли текущий документ в выборку.

При использовании ключевого слова **REM**, текст комментария должен быть заключен в **двойные кавычки**. Если в текст необходимо вставить ‘ или “ кавычки, то перед символом кавычек ставится символ \.

Пример. REM “Вставить \”Новую главу \”и ее номер”;

Ключевое слово SELECT содержит один аргумент: формулу, выделяющую документы. Если результирующий аргумент возвращает истинное значение для документа, то он будет выделен.

Пример. Выберем все документы, составленные по форме Main Topic, а также все документы «ответа» и «ответа на ответ»

Form = “Main Topic” | @Alldescendants

Если вместо символа «|» поставить «&», то будут отображаться документы, составленные по форме Main Topic и имеющие тип «ответ» или «ответ на ответ».

Синтаксис ключевых слов DEFAULT и FIELD

DEFAULT variablename:=value
FIELD fieldname:=value

Пример. FIELD cName: = cName – с помощью такой формулы можно проверить существует ли поле **cStatus**, и если существует, то проверить, не изменилось ли его значение. А ключевое слово DEFAULT следует использовать, если необходимо убедиться в том, что поле **cStatus** не существует и оно инициализировано, как new.

Пример. DEFAULT Avto:= "BMW" – если поле Avto не существует, то необходимо создать его экземпляр и присвоить ему значение BMW. Если поле Avto существует, то следует использовать текущее значение поля.

Пример. FIELD Avto: = @If (Avto) = "" ; "Daewoo" ; Avto) – проверяем значение поля, и если оно пустое, то ему присваивается значение Daewoo, иначе используется текущее значение поля.

Пример. Если необходимо сохранить последний используемый навигатор, то его следует сохранить в среде с помощью формулы:

ENVIRONMENT LastNav: = navigatorname

где navigatorname – текстовая величина, содержащая имя или псевдоним навигатора. Значение переменной можно получить с помощью @Environment**.**

5.4 Применение оператора присваивания и управляющих операторов

Кроме рассмотренного выше оператора присваивания (:=) существует еще 3 способа присвоения значений: **DEFAULT, FIELD,** функция **@SetField().** Присвоить значение переменной можно только посредством ключевого слова DEFAULT. Другими двумя способами присваиваются значения полям документа.

Чтобы добавить новое поле, ему необходимо присвоить значение с помощью 3-х вышеприведенных способов. Чтобы удалить поле используют формулу: **FIELD fieldname:= @DeleteField.**

Значения полей можно присваивать переменным, а значение переменных – полям. По умолчанию, любое значение, присваиваемое переменной, является временным и область видимости переменной ограниченно формулой.

К операторам, управляющими ходом выполнения формулы (управляющим операторам) относят: **@If(), @Do(), @Return().**

Основная форма оператора @If ():

@If(условие; оператор, если условие истинно; оператор, если условие ложно);

Операторы для истинного и ложного условий могут быть любого типа, за исключением оператора присваивания.

Пример. В событие *Input Validation* (подтверждение ввода) поля Avto введем формулу: **@If(@IsDocBeing Saved & Avto=””“; @Failure(“Вы должны ввести значение поля Avto!”); @Success);**

Эта формула не выполняется до тех пор, пока документ не будет сохраняться. Если при этом поле Avto не заполнено, то функция *@Failure* возвращает пользователю окно, содержащее взятый в кавычки текст. Затем курсор будет помещен в поле Avto. Эта формула является типичной для поля, требующего обязательного заполнения.

Расширенная форма условного оператора:

@If(1-е условие; оператор, если 1-е условие истинно; 2-е условие; оператор, если второе условие истинно; … ; оператор, если все условия ложны);

Если 1-е условие ложно, то производится проверка 2-го условия, и т.д. Оператор @If() всегда должен иметь нечетное число аргументов, независимо от количества условий в операторе.

В оператор @If() можно вкладывать другие операторы @If().

Функция *@Return()* прекращает выполнение формулы. Функцию *@Do()* можно использовать для выполнения слева направо ряда операторов, расположенных в одной строке.

В операторе *@If()* невозможно выполнить больше одного оператора, если проверяемое условие выполняется. Но применение оператора *@Do()* позволяет расположить ряд операторов в одной строке и выполнить их.

Пример. REM “Отправить документ для рассмотрения”;
jvYesNo:=@Prompt([YesNo]; “Вы уверены”; “Вы хотите отправить документ?”);
@If(jvYesNo; @Success; @Return(“”));
@SetField(“DocStatus”; “Отправка”);
@SetField(“StatusSort”;4);

Функция **@Prompt** отображает диалоговое окно с заголовком (второй параметр), двумя кнопками Yes, No и текстом в окне (третий параметр). Она возвращает 1, если пользователь нажал Yes, и 0 – если No.

Когда прользователь отвечает Yes, формула продолжает выполняться, сохраняя при этом соответствующие значения в полях DocStatus и StatusSort. Если пользователь отвечает No, то функция @Return прекращает выполнение формулы.

5.5 Логические функции @Functions, работа со строками, функции даты/ времени

Основные логические функции языка формул приведены в таблице 5.1.

Таблица 5.1 – Логические функции

Функция	**Описание**
@IsAgentEnabled	Возвращает True, если выполняется программа-агент.
@IsAvailable	Проверяет, существует ли поле в документе.
@IsCategory	Возвращает True, если какой-либо документ в представлении является категорией.
@IsDocBeingEdited	Возвращает True, если документ находится в режиме редактирования.
@IsDocBeingLoaded	Возвращает True, если документ открывается.
@IsDocBeingRecalculated	Возвращает True, если документ пересчитывается.
@IsDocBeingSaved	Возвращает True, если документ сохраняется.
@IsMember	Возвращает True, если элемент текста или текстовый список является элементом другого текстового списка.
@IsNewDoc	Возвращает True, если документ создан, а не открыт.
@IsNotMember	Возвращает True, если элемент текста или текстовый список не является элементом другого текстового списка.
@IsNumber	Возвращает True, если проверяемое значение является числом.
@IsResponseDoc	Возвращает True, если документ является ответным документом.
@IsText	Возвращает True, если проверяемое значение является текстом.
@IsTime	Возвращает True, если проверяемое значение является значением даты/времени или списком дат/времен.
@IsUnAvailable	Возвращает True, если поле в данном документе отсутствует.
@IsValid	Возвращает True, если все формулы проверок не обнаружили ошибок.
@IsFalse	Возвращает False.
@IsTrue	Возвращает True.

Функции **@IsDocBeingEdited, @IsNewDoc** могут использоваться для сокрытия действий или чтобы определить, следует ли заполнять поле данными. Например**,** если поместить @**IsNewDoc** в окно формулы закладки *Action Hide When* для кнопки действия и включить флажок *Hide action if formula is true*, то при создании документа эта кнопка будет скрыта.

В таблице 5.2 перечислены основные функции, работающие со значениями даты и времени.

Таблица 5.2 – Функции даты/времени

Функция	**Описание**
@Accessed	Возвращает дату и время последнего обращения к документу.
@Adjust()	Изменяет дату как в сторону уменьшения, так и увеличения.
@Created	Возвращает дату и время создания документа.
@Date()	Возвращает дату, являющуюся составной частью значения даты/времени.
@Day()	Возвращает день месяца, входящий в значение даты/времени.
@Hour()	Возвращает часовую компоненту значения даты/времени.
@Minute()	Возвращает минуты, являющиеся компонентой значения даты/времени.
@Modified	Возвращает список дат и времен модификаций документа.
@Month()	Возвращает номер месяца, входящий в значение даты/времени.
@Now()	Возвращает текущую дату и время с клиентского компьютера.
@Second()	Возвращает секунды, являющиеся компонентой значения даты/времени.
@Texttotime()	Преобразует текстовое представление даты/времени в значение даты/времени.
@Time()	Возвращает временную компоненту значения даты/времени.
@Today	Возвращает текущую дату.
@Tomorrow	Возвращает завтрашнюю дату.
@Weekday()	Возвращает номер дня недели, входящий в значение даты/времени.
@Year()	Возвращает год, входящий в значение даты/времени.
@Yesterday()	Возвращает вчерашнюю дату.
@Zone	Возвращает часовой пояс, являющийся компонентом значения даты/времени.

Функции **@Accessed, @Created, @Modified** содержат дату и время последнего обращения к документу, дату и время создания документа, и дату и время модификации документа соответственно. Данные функции часто используются в полях, в которых подсчитывается значения даты/времени, отражающие информацию о работе с документом.

В таблице 5.3 перечислены основные строковые функции (функции для работы с текстовой информацией) языка формул. Все строковые функции можно разделить на 4 группы:

1) функции нахождения подстрок внутри строк;
2) функции выделения подстрок из строк;
3) функции сравнения строк;
4) функции манипуляции со строками.

Пример. Преобразуем дату модификации документа из стандартного формата месяц/дата/год (например, 05/15/08) в формат текстовой строки (например, 15–May–2008). В вычисляемом поле *DateModified* в свойстве *Value* запишем следующие формулы:

1. В переменную *date* занесем сегодняшнюю дату в стандартном формате: **date := @Today;**

2. Получим день месяца, который будет отображаться двумя символами (независимо от дня, т.е. 2→02): **day := @Right("0" + @Text(@Day(date)); 2);**

3. Получим месяц в трехбуквенном виде: **month := @Replace(@Text (@Month(date)); "1" : "2" : "3" : "4" : "5" : "6" : "7" : "8" : "9" : "10" : "11" : "12"; "Jan" : "Feb" : "Mar" : "Apr" : "May" : "Jun" : "Jul" : "Aug" : "Sep" : "Oct" : "Nov" : "Dec");**

4. Получим год в полном формате: **year := @Right(@Text(@Year (date)); 4);**

5. Составляем дату в требуемом порядке: **@Text(day) + "–" + @Text(month) + "–" + @Text(year)**.

Используя приведенные в примере функции и функции из таблицы 5.3 можно осуществлять различные преобразования значений даты/времени (например, преобразование, обратное приведенному в примере).

Таблица 5.3 – Строковые функции

Функция	**Номер группы**	**Описание**
@Begins()	1	Определяет, не начинается ли строка другой строкой.
@Contains()	1	Определяет, не содержит ли строка другую строку.
@Ends()	1	Определяет, не заканчивается ли строка другой строкой.
@Left()	2	Возвращает крайние левые символы строки, осуществляя поиск слева направо.
@LeftBack()	2	Возвращает крайние левые символы строки, осуществляя поиск справа налево.
@Length()	4	Возвращает длину строки.
@Like()	3	Сравнивает 2 строки, совместима с ANSI SQL.
@LowerCase()	4	Устанаваливает нижний регистр для отображения символов строки .
@Matches()	3	Сравнивает 2 строки.
@Middle()	2	Возвращает символы из середины строки, осуществляя поиск слева направо.
@MiddleBack()	2	Возвращает символы из середины строки, осуществляя поиск справа налево.
@ProperCase()	4	Устанаваливает соответствующий регистр для отображения символов строки, при этом первые буквы слов будут большими.
@Repeat()	4	Повторяет строку.
@Replace	4	Выполняет операции поиска и замены над текстовым списком
@ReplaceSubString()	4	Замещает элементы строки.
@Right()	2	Возвращает крайние правые символы строки, осуществляя поиск слева направо.
@RightBack()	2	Возвращает крайние левые символы строки, осуществляя поиск справа налево.
@Text()	4	Преобразует данные других типов в текстовые строки.
@Trim()	4	Удаляет пробелы в начале и конце строки.
@UpperCase()	4	Устанаваливает верхний регистр для отображения символов строки.

5.6 Получение информации о сеансе работы и пользователе

Основные функции работы с информацией о пользователе и сеансе работы приведены в таблице 5.4.

Таблица 5.4 – Функции применяемые для получения информации о сеансе работы и пользователе

Функция	**Результат**
@BrowserInfo()	Возвращает информацию о браузере Web.
@ClientType	Возвращает тип программы клиента (Web или Notes), используемой пользователем.
@UserName	Возвращает имя пользователя.
@Name()	Возвращает компоненты имени пользователя, когда используется совместно с функцией @UserName.
@MailDBName	Возвращает имя текущего почтового сервера пользователя и путь к его почтовому файлу.
@UserAccess()	Возвращает уровень доступа пользователя к базе данных.
@UserRoles	Возвращает список ролей пользователя в базе данных.

Функция **@Username** возвращает имя текущего пользователя в каноническом формате (если оно является иерархическим именем пользователя). Иерархическое имя состоит из следующих компонент: полное имя, название организации, название подразделения и иногда названия страны. Идентификатор пользователя *notes* – иерархическое имя. Если имя не иерархическое, то функция **@Username** отображает только ту его часть, которое является соответственно именем.

Не иерархическое имя может быть, например, у пользователя Web. Канонический формат содержит все компоненты имени, включая: общее имя (CN), подразделение (OU), организацию (O), страну (C). Имя в полном каноническом формате выглядит следующим образом:

CN=Fox Mulder/OU=VMIP/O=GGU/C=Belarus

Но часто не хочется отображать название компании, страну и т.д. Функция **@Name([аргумент];имя Notes)** извлекает различные части канонического имени. Аргументы функции *@Name()* приведены таблице 5.5. Наиболее часто используется аргумент [**CN]**, возвращающий общее имя, и аргумент [**Abbreviate**], который возвращает общее имя, название подразделения и название организации, разделенные наклонными чертами.

Таблица 5.5 – Аргументы функции @Name()

Аргумент	**Что возвращает**
[A]	Возвращает доменное имя административного управления.
[Abbreviate]	Отображает имя в сокращенном формате.
[C]	Возвращает название страны.
[Canonicalize]	Возвращает сокращенное имя в полном каноническом формате.
[CN]	Возвращает общее имя.
[G]	Возвращает имя.
[I]	Возвращает инициалы.
[O]	Возвращает название организации.
[OU*N*]	Возвращает название n-ого подразделения, например [OV1], [OV2].
[P]	Возвращает доменное имя частного управления.
[Q]	Возвращает обобщенный классификатор.
[S]	Возвращает фамилию.
[ToKeyword]	Отображает в обратном порядке все части имени (за исключением общего имени), которые разделяются обратными наклонными чертами (C\O\OU).

Аргументы [A], [G], [I], [P], [Q], [S] были спроектированы для использования в шлюзовых и других почтовых системах. Они **не используются с именами Notes.**

Пример. Пользователь *Fox Mulder* работает в подразделении *VMiP* организации *GGU*. Результатом выполнения формулы **@Name([CN];@UserName)** будет: Fox Mulder; а результатом выполнения формулы **@Name([CN];@UserName)** – Fox Mulder/VMIP/GGU.

Функции @Name и @UserName можно использовать для создания хронологии редактирования документа (имя пользователя и дата редактирования, см. [7, стр. 441]).

5.7 Работа с документами в языке формул

С помощью функций *@Functions* можно получать информацию о размере документа, об именах всех присоединенных файлов, о длине этих файлов и т.д. Основные функции для получения информации о документах приведены в [7, табл.19,8 с 442].

Например, если имеется БД, которая используется в качестве архива проектов, то для отображения информации о присоединенных файлах можно использовать следующие функции: **@Attachments, @AttachmentLengths, @AttachmentNames.**

Функции **@AllChildren** (возвращает ответные документы)**, @AllDescendants** (возвращает ответные документы и документы «ответ на ответ»)**, @IsResponseDoc** (возвращает значение True, если документ является ответным) используются в формулах отбора представления, чтобы отображать в представлении как главные документы, так и любые ответные документы.

В формулах отбора для представлений для отображения главных документов вместе с ответными лучше использовать функции **@AllChildren**, **@AllDescendants**, но не **@IsResponseDoc**. Функция **@IsResponseDoc** возвращает все ответные документы, а не только те из них, которые связаны с главными документами в представлении.

Логические функции «@Is» (@IsDocBeingEdited, @IsDocBeingLoaded, @IsDocBeingMailed, @IsDocBeingRecalculated, @IsDocBeingSaved) применяются тогда, когда необходимо предпринимать некоторые действия, зависящие от состояния документа. В поле **$Ref** храниться уникальный идентификатор UNID родительского документа, и оно обеспечивает ссылку на документ для обновления значения поля. Это можно использовать в функции **@SetDocField()** для обновления родительского документа из ответного.

5.8 Выборка данных с помощью функций @DBColumn и @DbLookup

Функции **@DBColumn()** и **@DbLookup()** используются для выборки данных. Также для извлечения данных из источников *ODBC* (Open Database Connectivity – стандарт доступа к базам данных, в соответствии с которым осуществляется объединение источников данных разных типов) может использоваться функция **@DBCommand()**.

Функции выборки данных **@DBColumn()**, **@DbLookup()**, **@DBCommand()** могут возвращать данные, размер которых не превышает 64 Кб. Чтобы обойти это ограничение можно использовать функцию **@PickList**.

Функция **@DBColumn** возвращает список значений из заданного столбца представления. Представление может находиться как в текущей базе данных, так и какой-либо другой. Функция **@DbLookup** отличается от **@DBColumn** тем, что в ней можно задавать значение ключа. Это значение сравнивается с первым столбцом сортировки представления, и функция выбирает значения только для документов, соответствующих ключу.

Функции **@DBColumn** и **@DbLookup** имеют похожий синтаксис:

@DBColumn (Class : NoCashe; server : database; view name; column number);

@DBLookup (Class : NoCashe; server : database; view name; key value; column number or field name);

Параметр *Class* определяет тип базы данных. Базы данных Notes обозначаются словом “Notes” или парой двойных кавычек. Параметр *NoCashe* указывает, что результат не будет запоминаться в оперативной памяти (используется по умолчанию). Параметр *server* – имя текущего сервера, вместо него можно использовать пару двойных кавычек. Параметр *database* может определять текущую базу данных (представленную парой двойных кавычек) или быть идентификатором ID реплики базы данных. Параметр *view name* – имя представления или его псевдоним. В функции **@DBColumn()** необходимо обязательно задавать номер столбца (*column name*), а в функции **@DBLookup()** можно задавать или номер столбца или имя поля документа.

Функции **@DBColumn() и @DbLookup()** могут использоваться для просмотра источников данных ODBC. Например, можно использовать драйвер ODBC для SQL server или Sybase. После завершения настройки драйвера ODBC можно будет просматривать информацию в другой базе данных, задавая имя таблицы, имя ключевого столбца и значение ключа (для функции **@DbLookup()**), или задавая имя таблицы и имя столбца (для функции **@DBColumn()**).

Пример. Просмотрим информацию об отделах в таблице из базы данных SQL server.

@DbColumn (“ODBC” : “NoCashe”; “HR Lookup”; skern : nreks ; Department; cDeptName; “Distinct” : “Ascending”)

Первые два параметра такие же, как в синтаксисе функции **@DbColumn()** для данных Domino. Третий параметр (*HR Lookup*) – имя источника данных. Четвертый параметр (*skern*) – список пользователей, у которых есть права на подключение к внешней базе данных, в седьмой версии Lotus после этого параметра идет список паролей соответствующих пользователей. Параметр *Department* – имя таблицы базы данных, параметр *cDeptName* – имя столбца в таблице базы данных. Предпоследний параметр (*Distinct*) удаляет повторяющиеся записи, последний параметр *Ascending* – порядок сортировки.

С помощью функции **@DbCommand()** можно передавать операторы SQL в источники данных ODBC.

Вопросы для самоконтроля

1. Какие типы констант существуют в языке формул?
2. Перечислите 6 основных типов операторов языка формул?
3. Чувствителен ли к регистру язык формул?
4. В чем отличие команд от функций?
5. Перечислите ключевые слова языка формул и области их применения?
6. Какие управляющие операторы существуют в языке формул?
7. Синтаксис оператора *@If*.
8. Перечислите основные логические функции языка формул?
9. Какие основные функции работы со значениями даты/ времени существуют в языке формул?
10. Перечислите основные функции работы с текстовой информацией в языке формул?
11. Какие функции применяются для получения информации о сеансе работы и пользователе?
12. Перечислите основные функции для работы с документами в языке формул?
13. Какие существуют особенности использования функции *@Column, @DBLookUp*?
14. Что такое *ODBC*?

Задание

Напишите формулы для ранее созданных в учебной базе данных кнопок-действий на форме и в представлениях. Сделайте вывод названия базы данных в наборе фреймов динамическим на основе заголовка заданного в свойствах БД. На форму в область заголовка добавьте вычисляемые поля, отображающие кто и когда создал документ, а также кто и когда его изменил. Напишите формулы для вычисляемых полей, отображающих информацию о работе с документом (размер документа, дата создания и т.д.) на второй вкладке таблицы. Даты создания и модификации документа в области заголовка формы необходимо выводить в формате текстовой строки (см. пример на стр. 40).

ЛИТЕРАТУРА

1 Березина, Н. С. Использование почты Lotus Notes R5 / Н. С. Березина, Е. Н. Трубникова. – М. : Интертраст, 2002. – 195 с.

2 Березина, Н. С. Начальный курс Lotus Notes R5 / Н. С. Березина, Е. Н. Трубникова. – М. : Интертраст, 2002. – 285 с.

3 Ионцев, Н. Н. Почтовая система сервера Lotus Domino R5 и ее конфигурирование / Н. Н. Ионцев. – М. : Интертраст, 2001. – 135 с.

4 Ионцев, Н. Н. Программирование в Lotus Domino R.5.: формулы и функции, язык LotusScript, встроенные классы LotusScript и Java / Н. Н. Ионцев, Е. В. Поляков, О. Г. Таранченко. – М. : Интертраст, 1999. – 456 с.

5 Керн, С. Lotus Notes и Domino 6. Руководство разработчика / С. Керн [и др.]; пер. с англ. – К. : ООО «ТИД «ДС», 2005. – 880 с.

6 Кирклэнд, Р. Domino 5 & 6. Администрирование сервера / Р. Кирклэнд; пер. с англ. – М. : ДМК Пресс, 2003. – 832 с.

7 Линд, Д. Lotes Notes и Domino 5/6. Энциклопедия программиста / Д. Линд, С. Керн; пер. с. англ. – 2-е изд., перераб. и доп. – К. : ООО «ТИД ДС», 2003. – 1024 с.

8 Некрасов, В. В. Администрирование Lotus Domino R5 в вопросах и ответах / В. В. Некрасов. – М. : Интертраст, 2002. – 518 с.

9 Некрасов, В. В. Почтовая система Сервера Lotus Domino 7.0 / В. В. Некрасов. – М. : ООО «Светотон», 2004. – 273 с.

10 Поляков, Е. В. Изучение новых возможностей IBM Lotus Domino Designer 6 / Е. В. Поляков. – М. : Интертраст, 2002. – 245 с.

11 Поляков, Е. В. Средства разработки приложений в Lotus Domino R5: Domino Designer/ Е. В. Поляков. – М. : Интертраст, 2003. – 467 с.

12 Поляков, Е. В. Язык @-формул в Lotus Domino R6. Справочник разработчика / Е. В. Поляков. – М. : Интертраст, 2004. – 347 с.

13 Поляков, Е. В. Domino Designer R6.5 – интегрированная среда разработки приложений в Lotus Domino: учебное пособие для вузов / Е. В. Поляков. – М. : Интертраст, 2005. – 640 с.

14 IBM Corp. Lotus Domino 6. Administering the Domino System, Volume 1. – IBM Corp., 2002 – 1354 p.

15 IBM Corp. Lotus Domino 6. Administering the Domino System, Volume 2. – IBM Corp., 2002 – 1150 p.

16 IBM Corp. Lotus Domino 6. Installing Domino Servers. – IBM Corp., 2002 – 187 p.

17 Speed, T. Lotus Notes Domino 8: Upgrader's Guide / T. Speed, D. McCarrick, B. Gibson. – Packt Publishing, 2008. – 276 p.

18 Watt D. Tunning IBM @server xSeries Servers for Performance. D Watt [and others]. – IBM Corp. 2002. – 866 p.

Учебное издание

КУЗЬМЕНКОВ Дмитрий Сергеевич

LOTUS DOMINO/NOTES

ПРАКТИЧЕСКОЕ РУКОВОДСТВО
по выполнению лабораторных работ

для студентов математического факультета специальностей
1-31 03 03-02 «Прикладная математика (научно-педагогическая деятельность)»
1-31 03 06 01 «Экономическая кибернетика (математические методы
и компьютерное моделирование в экономике)»
1-40 01 01 «Программное обеспечение информационных технологий»

В 2-х частях
Часть 1

В авторской редакции

Подписано в печать 11.10.2011 г. (9). Формат издания 60*80 $\frac{1}{16}$.
Бумага офсетная. Гарнитура «Таймс».
Усл. печ. л. 1,16. Уч.-изд. л. 1,25. Тираж 20 экз.

Отпечатано в учреждении образования
«Гомельский государственный университет
имени Франциска Скорины»,
246019, г. Гомель, ул. Советская, 104.