Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Тульский государственный университет»
Технический колледж им. С.И. Мосина

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

по дисциплине «Базы данных»

специальности
09.02.01 Компьютерные системы и комплексы

Тула 2022

УТВЕРЖДЕНЫ
Цикловой комиссией информационных технологий
Протокол от 13 января 2022г. № 6
Председатель цикловой комиссии
_____ И.В Миляева

Лабораторная работа № 1

СОЗДАНИЕ БАЗЫ ДАННЫХ СРЕДСТВАМИ СУБД. СОЗДАНИЕ ТАБЛИЦ

1. ЦЕЛЬ РАБОТЫ

Изучение средств автоматизации конструирования реляционных таблиц в СУБД ОО Base.

2. ОСНОВЫ ТЕОРИИ

Создание базы данных

База данных (БД) представляет собой совокупность средств для ввода, хранения, просмотра, выборки и управления информацией. К этим средствам относятся таблицы, формы, отчеты, запросы.

Поддерживаются два способа создания базы данных. Вы можете создать пустую базу данных, а затем добавить в нее таблицы, формы, отчеты и другие объекты. Такой способ является наиболее гибким, но требует отдельного определения каждого элемента базы данных. Кроме этого имеется возможность создать базу данных с помощью мастера со всеми необходимыми формами, таблицами и отчетами.

Создание новой базы данных осуществляется командой **Файл | Создать** . После ввода имени создаваемой базы и нажатия кнопки **Создать** откроется окно базы данных. Оно состоит из шести вкладок, которые пока пусты. Необходимо далее создать все компоненты, входящие в базу данных. Их перечень соответствует ярлыкам вкладок окна базы данных.

Основным компонентом базы данных являются таблицы. Они хранят всю информацию, помещаемую в БД. В таблицы будет водится информация, которая может дополняться изменяться и удаляться.

Создание таблицы осуществляется в окне БД. Создать таблицу можно несколькими способами: использовать мастер таблиц, создать таблицу в режиме конструктора таблиц, импортировать таблицу из внешнего файла. В любом случае каждой таблице присваивается определенное имя.

Создание таблицы в окне конструктора

Создание таблиц в окне конструктора предоставляет большие возможности и осуществляется выбором из списка вариантов значения **Конструктор.** В результате выполнения этих действий откроется окно конструктора. В верхней части окна диалога находится таблица, которая содержит следующие атрибуты: наименование поля, тип данных и описание. Наименование каждого из полей таблицы выбирается произвольно в соответствии с помещаемой в него типом информации.

Наименование поля должно быть уникально может содержать до 64

символов, исключая точку (.), восклицательный знак (!), прямых скобок ([]) и управляющих символов с кодами ASCII – 0-31.

Тип поля определяется типом данных, хранящихся в этом поле. В допустимы следующие типы: текстовый (до 255 символов), числовой, денежный (8 байт, до 4 знаков после запятой), счетчик (данные не редактируются), дата/время, логический, поле MEMO (до 64000 символов), поле объекта OLE (размер определяется объемом жесткого диска), мастер подстановок (создает поле, в котором предполагается выбор значений из раскрывающегося списка значений других таблиц).

Каждый из типов поля наделен собственными свойствами, которые отображаются в разделе «Свойства поля» окна конструктора.

Создать структуру таблицы можно следующим образом:

1. В окне конструктора в столбце **Имя поля** вводится имя поля данных

2. В столбец **Тип данных** из раскрывающегося списка вводится значение типа данных.

3. Столбец Описание представляет из себя пояснение, которое вы даете своим полям. Это пояснение появляется в строке состояния во время работы с БД.

4. Аналогичным образом вводится описание всех полей таблицы.

5. Завершив ввод структуры, ее надо сохранить, выполнив команду **Файл|Сохранить** .

Создание таблицы в режиме таблицы

Рассмотрим способ создания таблиц, который отличается своей простотой и наглядностью. Приведем последовательность действий, которую предстоит выполнить:

1) Перейдите на вкладку «Таблицы» окна базы данных и нажмите кнопку Создать. 2) В окне диалога «Новая таблица» выберите из списка вариантов значение **Режим таблицы** и нажмите кнопку **ОК.**

В результате выполнения этих действий откроется окно диалога «Таблица», содержащее созданную по умолчанию таблицу. Эта таблица содержит 20 столбцов и 30 строк, и этого вполне достаточно для начала. После сохранения этой таблицы, конечно, можно добавить столько строк и столбцов, сколько вам понадобится. 3. Наименования полей таблицы определены по умолчанию, но очень просто присвоить полям новые имена. Для этого нажмите дважды кнопкой мыши на область выбора первого поля (заголовок которого содержит Поле1). Имя поля выделяется и появляется Мигающий курсор. Введите имя первого поля и нажмите клавишу TAB. Аналогично введите остальные имена полей вашей таблицы в следующих столбцах.

4. Теперь заполните несколько строк вашей таблицы, вводя информацию в том виде, в каком она будет вводиться и в будущем. Старайтесь записывать все в одном стиле (например, если первую дату вы записали 10/14/96, то не пишите следующую в виде Ноябрь 3, 1996). Конечно, если установит

неправильный тип данных, вы можете позже его изменить, но и вам желательно стараться вводить все правильно с самого начала.

5. Сохраните таблицу, выполнив команду **Файл | Сохранить макет** или нажав кнопку **Сохранить** на панели инструментов, В открывшемся окне диалога «Сохранение» присвойте таблице имя и нажмите кнопку **ОК**.

6. На запрос о необходимости создания для таблицы первичного ключа нажмите кнопку **Да**, и создаст таблицу, удалив лишние строки и столбцы.

7. Теперь убедитесь, что выбрал для каждого поля правильные типы данных. Для этого перейдите в окно конструктора таблицы, выполнив команду Вид | Конструктор таблиц. Если вас что-то не устраивает в структуре таблицы, внесите необходимые изменения.

3. ОБЪЕКТЫ И СРЕДСТВА ИССЛЕДОВАНИЯ

Предметная область в соответствии с вариантом задания. ПК типа IBM PC, дисплейный класс, операционная система Windows XP и выше, СУБД OO Base.

4. ПОДГОТОВКА К РАБОТЕ

Ознакомиться с теоретическими положениями лабораторной работы.

5. ПРОГРАММА РАБОТЫ

По предложенному преподавателем варианту разработать необходимые таблицы БД и заполнить их информацией. Оформить отчет.

ВАРИАНТЫ ЗАДАНИЙ . Выбирается преподавателем из списка заданий (См. Приложение) и остается единым для всех лабораторных работ.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

6.1. Назовите основные компоненты базы данных ?

6.2. Основные способы создания таблиц?

6.3. Создание таблиц в режиме мастера?

6.4. Создание таблиц в режиме таблиц?

Лабораторная работа № 2

РАБОТА С ТАБЛИЦАМИ: ДОБАВЛЕНИЕ, РЕДАКТИРОВАНИЕ, УДАЛЕНИЕ, НАВИГАЦИЯ ПО ЗАПИСЯМ

1. ЦЕЛЬ РАБОТЫ

Изучение средств модификации реляционных таблиц в СУБД ОО Base.

2. ОСНОВЫ ТЕОРИИ

Модификация структуры таблицы

Описанную структуру таблицы, не сделав при этом ни одной ошибки, вам необходимо сохранить.

Для исправления возможных допущенных ошибок предоставляет необходимые средства. К их числу относятся:

1) Изменение наименования поля и/или его типа . 2) Вставка пропущенного поля 3)Удаление ошибочно введенного поля. 4) Изменение порядка следования полей в таблице

Для модификации структуры таблицы, входящей в базу данных, установите в окне базы данных указатель на модифицируемую таблицу и нажмите кнопку **Конструктор.**

Изменение наименования поля или его типа осуществляется после установки указателя на данное поле или тип, который надо ввести. Неправильные символы удаляются клавишей Delete. После этого вводятся правильные символы.

Изменение порядка следования полей осуществляется нажатием на область выбора поля и после выделения строки и повторного нажатия левой кнопки мыши перетаскиванием строки (столбца) в нужное место.

Удаление полей таблицы осуществляется клавишей Delete после их выделения.

Добавление нового поля осуществляется командой **Вставка | Поле.**

Установка первичного ключа

Когда напоминает вам об отсутствии первичного ключа и предлагает его создать, нажмите Да. добавит поле код в первой строке описания структуры таблицы.

Если вы хотите установить первичный ключ самостоятельно, нажмите на поле, которое будет в дальнейшем использоваться в качестве первичного ключа. Затем нажмите правую кнопку мыши и выберите пункт всплывающего **Первичный** ключ. В области выбора поля, которое будет использоваться как ключ, появится маленькая пиктограмма с изображением первичного ключа. Для установки первичного ключа также можно использовать кнопку Ключевое поле на панели инструментов.

3. ОБЪЕКТЫ И СРЕДСТВА ИССЛЕДОВАНИЯ

Предметная область в соответствии с вариантом задания. ПК типа IBM PC, дисплейный класс, операционная система Windows XP и выше, СУБД OO Base.

4. ПОДГОТОВКА К РАБОТЕ

Ознакомиться с теоретическими положениями лабораторной работы.

5. ПРОГРАММА РАБОТЫ

По предложенному преподавателем варианту заполнить таблицы информацией, добавить, отредактировать записи. При построении таблиц используйте индексацию записей. Оформить отчет.

ВАРИАНТЫ ЗАДАНИЙ . Выбирается преподавателем из списка заданий (См. Приложение) и остается единым для всех лабораторных работ.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

6.1. Расскажите о назначении и формах индексации записей БД.

6.2. Как осуществляется навигация в таблицах БД?

Лабораторная работа № 3

СОРТИРОВКА, ПОИСК И ФИЛЬТРАЦИЯ ДАННЫХ

1. ЦЕЛЬ РАБОТЫ

Изучение средств автоматизации сортировки, поиска данных в реляционных таблицах.

2. ОСНОВЫ ТЕОРИИ

Одним из основных требований, предъявляемых к СУБД, является возможность быстрого поиска требуемых записей среди большого объема информации. Индексы представляют собой наиболее эффективное средство, которое позволяет значительно ускорить поиск данных в таблицах по сравнению с таблицами, не содержащими индексов. В зависимости от количества полей, используемых в индексе, различают *простые* и *составные индексы.*

В допускается создание произвольного количества индексов. Индексы создаются при сохранении макета таблицы и автоматически обновляются при вводе и изменении записей. Можно в любое время добавить новые или удалить ненужные индексы в окне конструктора таблиц.

Требование уникальности индекса в не является обязательным. Для ускорения поиска требуемой информации могут быть использованы индексы, не являющиеся уникальными

Важной особенностью индексов является то, что можно использовать индексы для создания *первичных ключей.* Первичный ключ содержит информацию, которая однозначно идентифицирует запись. В этом случае индексы должны быть уникальными. Это означает, что для таблицы, содержащей только одно индексное поле, уникальными должны быть значения этого поля. Для составных индексов величины в каждом из индексных полей могут иметь повторяющиеся значения. Однако индексное выражение должно быть уникальным.

Прежде чем определить первичный ключ в таблице, просмотрите все поля создаваемой вами таблицы. Есть ли хоть одно поле, информация которого была бы уникальна для каждой записи? Даже если использовать полное имя, то есть фамилию, имя и отчество одновременно, оно также не будет неповторимым.

Часто наилучшее решение этой проблемы заключается в том, чтобы каждой записи в таблице поставить в соответствие идентификационный номер. Это и делает , когда вы предлагаете ему создать первичный ключ. Он создает поле Код с типом данных **Счетчик.** Это означает, что каждый раз при создании новой записи значение счетчика увеличивается на 1. Этот номер и является первичным ключом для каждой новой записи.

Создание индекса для одного поля

Для создания простого индекса используется свойство поля **Индексированное** поле, позволяющее ускорить выполнение поиска и сортировки записей по одному полю таблицы. Индексированное поле может содержать как уникальные, так и повторяющиеся значения.

Данное свойство поля **Индексированное поле** может принимать следующие значения:

Нет Значение по умолчанию. Индекс не создается.

Да (Допускаются совпадения) В индексе допускаются повторяющиеся значения

Да (Совпадения не допускаются) Повторяющиеся значения в индексе не допускаются

Для создания простого индекса необходимо выполнить следующие действия: 1) В окне конструктора таблицы выберите в верхней половине окна поле, для которого создается индекс. 2) В нижней половине окна для свойства **Индексированное поле** выберите одно из следующих значений: **Да (Допускаются совпадения)** или **Да (Совпадения не допускаются).**

Значение **Да (Совпадения не допускаются)** обеспечивает уникальность каждого значения данного поля.

Не допускается создание индексов для полей MEMO и полей объектов OLE.

Создание составного индекса

Индексы, содержащие несколько полей, следует определять в окне индексов.

1. В окне конструктора откройте таблицу, для которой вы создаете составной индекс. Для этого в окне базы данных установите указатель на данную таблицу и нажмите кнопку Конструктор.
2. Нажмите кнопку Индексы на панели инструментов. На экране откроется окно диалога «Индексы»
3. В открывшемся окне диалога введите имя индекса в поле столбца Индекс в первой пустой строке. В качестве имени индекса можно использовать имя одного из полей, включенных в индекс, или любое допустимое имя.
4. В столбце Имя поля той же строки нажмите кнопку раскрытия списка и выберите первое поле индекса.
5. В столбце Имя поля следующей строки выберите имя следующего поля индекса. (В этой строке поле столбца Индекс следует оставить пустым). Определите таким же образом остальные поля индекса. Индекс может включать до 10 полей.
6. Закончив выбор полей для индекса, нажмите кнопку закрытия окна, расположенную в строке заголовка окна диалога.

По умолчанию задается порядок сортировки По возрастанию. Для сортировки конкретного поля по убыванию выберите в столбце **Порядок сортировки** строки с выбранным полем значение **По убыванию**.

Установка первичного ключа

Когда напоминает вам об отсутствии первичного ключа и предлагает его создать, нажмите Да. добавит поле код в первой строке описания структуры таблицы.

Если вы хотите установить первичный ключ самостоятельно, нажмите на поле, которое будет в дальнейшем использоваться в качестве первичного ключа. Затем нажмите правую кнопку мыши и выберите пункт всплывающего **Первичный** ключ. В области выбора поля, которое будет использоваться как ключ, появится маленькая пиктограмма с изображением первичного ключа. Для установки первичного ключа также можно использовать кнопку Ключевое поле на панели инструментов.

3. ОБЪЕКТЫ И СРЕДСТВА ИССЛЕДОВАНИЯ

Предметная область в соответствии с вариантом задания. ПК типа IBM PC, дисплейный класс, операционная система Windows XP и выше, СУБД OO Base.

4. ПОДГОТОВКА К РАБОТЕ

Ознакомиться с теоретическими положениями лабораторной работы.

5. ПРОГРАММА РАБОТЫ

По произвольно выбранным полям осуществить сортировку и поиск записей. Оформить отчет.

ВАРИАНТЫ ЗАДАНИЙ . Выбирается преподавателем из списка заданий (См. Приложение) и остается единым для всех лабораторных работ.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

6.1. Назовите основные элементы таблиц БД?

6.2. Что такое первичный ключ и индекс?

6.3. Как осуществляется поиск информации в БД?

Лабораторная работа № 4

СПОСОБЫ ОБЪЕДИНЕНИЯ ТАБЛИЦ

1. ЦЕЛЬ РАБОТЫ

Изучение средств автоматизации проектирования схем БД.

2. ОСНОВЫ ТЕОРИИ

Определение отношений между таблицами

Вы можете устанавливать постоянные отношения между таблицами, которые будут поддерживаться при создании форм, отчетов и запросов.

Устанавливая связи между двумя таблицами, вы выбираете поле, которое содержит одну и ту же информацию. Чаще всего вы будете связывать первичный ключ одной таблицы с совпадающими полями другой таблицы.

Поля, с помощью которых устанавливается связь между двумя таблицами, могут иметь различные имена, но удобнее использовать совпадающие имена.

Из существующих типов связей тип «один-ко-многим» наиболее важен, поэтому самое пристальное внимание следует уделить ему. В отношении «один-ко-многим» *главной таблицей* является таблица, которая содержит первичный ключ и составляет часть «один» в отношении «один-ко-многим». *Внешний ключ* — это поле (или поля), содержащее такой же тип информации в таблице со стороны «много» в отношении «один-ко-многим», которую называют *подчиненной таблицей.*

Окно диалога «Схема данных»

Создание связей между таблицами в осуществляется в окне диалога «Схема данных». Для определения связей между таблицами необходимо выполнить следующие действия:

1. Откройте окно диалога «Схема данных», выполнив команду **Сервис | Схема данных** или нажав кнопку **Схема данных** на панели инструментов. На экране откроется окно диалога «Схема данных».

Перед определением связей между таблицами необходимо предварительно закрыть все открытые таблицы. Не допускается создание или удаление связей между открытыми таблицами.

2. Добавьте в это окно диалога последовательно две связываемые таблицы. Для этого выполните команду **Связи | Показать таблицу** или нажмите кнопку **Добавить таблицу** на панели инструментов. На экране откроется окно диалога «Добавление таблицы»

3. В списке таблиц выделите первую добавляемую таблицу и нажмите кнопку **Добавить.** Затем выберите вторую добавляемую таблицу и также нажмите кнопку **Добавить.** Затем нажмите кнопку **Закрыть** для закрытия

окна диалога «Добавление таблицы». В окне диалога «Схема данных» появились две связываемые таблицы.

4. Для связывания таблиц выберите поле в первой связываемой таблице и переместите его с помощью мыши на соответствующее поле второй таблицы. Для связывания сразу нескольких полей выберите эти поля при нажатой клавише *Ctrl* и переместите во вторую таблицу группу выделенных полей.

В большинстве случаев связывают ключевое поле (представленное в списке полей полужирным шрифтом) одной таблицы с соответствующим ему полем внешнего ключа (часто имеющим то же имя) во второй таблице. Связанные поля не обязательно должны иметь одинаковые имена, однако, они должны иметь одинаковые типы данных (из этого правила существует два исключения) и иметь содержимое одного типа. Кроме того, связываемые поля типа **Числовой** должны иметь одинаковые значения свойства Размер поля. Исключениями из этого правила являются поля счетчика с последовательной нумерацией, которые могут связываться с числовыми полями размера **Длинное целое**, а также поля счетчика с размером **Код репликации**, связываемые с полями типа **Числовой**, для которых также задан размер **Код репликации**.

5. На экране откроется окно диалога «Связи» . В данном окне диалога проверьте правильность имен связываемых полей, находящихся в столбцах. При необходимости выберите другие имена полей. Затем нажмите кнопку **Создать.** Вы вернетесь в окно диалога «Схема данных».

Тип создаваемой связи зависит от полей, которые были указаны при определении связи;

- Отношение «один-ко-многим» — создается в том случае, когда только одно из полей является ключевым или имеет уникальный индекс.

- Отношение «один- к-одному» — создается в том случае, когда оба связываемых поля являются ключевыми или имеют уникальные индексы.

- Связь с отношением «многие-ко-многим» — фактически представляет две связи с отношением «один-ко-многим» через третью таблицу, ключ которой состоит, по крайней мере, из двух полей, которые являются полями внешнего ключа в двух других таблицах.

В окне диалога «Схема данных» при переносе поля, не являющегося ключевым или не имеющего уникального индекса, на другое поле, которое также не является ключевым или не имеет уникального индекса, создается неопределенное отношение. В запросах, содержащих таблицы с неопределенным отношением, по умолчанию создает линию объединения между таблицами, но условия целостности данных при этом не поддерживаются и нет гарантии уникальности записей в любой из таблиц.

В окне диалога «Схема данных» можно также выполнять следующие действия:

1) Изменить структуру таблицы. 2) Изменить существующую связь. 3) Удалить связь. 4) Удалить таблицу из окна диалога «Схема данных». 5) Вывести на экран все существующие связи или связи только для конкретной

таблицы. 6) Определить связи для запросов, не задавая условия целостности данных.

Связывание двух полей одной таблицы

Иногда возникает необходимость в определении поля с подстановкой значений из той же таблицы. Для связывания поля таблицы с другим полем той же таблицы дважды добавьте эту таблицу в окно диалога «Схема данных» и создайте требуемую связь, соединив поля линией связи.

Создание между таблицами отношения «многие-ко-многим»

Рассмотрим создание между таблицами отношения «многие-ко-многим». В отношение «многие-ко-многим» представляет две связи с отношением «один-ко-многим» через третью таблицу, ключ которой состоит, по крайней мере, из двух полей, являющихся полями внешнего ключа в двух других таблицах.

Рассмотрим создание такой связи:

1) Создайте таблицы, между которыми требуется определить связь с отношением «многие-ко-многим». 2) Создайте третью (связующую) таблицу с полями, описание которых совпадает с описанием ключевых полей в каждой из двух связываемых таблиц. В этой таблице ключевые поля выполняют роль внешнего ключа. Другие поля в связующую таблицу можно добавлять без ограничений. 3) Определите в связующей таблице ключ, содержащий все ключевые поля двух связываемых таблиц. 4) Определите связи с отношением «один-ко-многим» между каждой из двух таблиц и связующей таблицей.

Изменение структуры таблицы в окне диалога «Схема данных»

При создании связи в окне диалога «Схема данных» может возникнуть необходимость в изменении структуры таблицы. При этом вы можете не покидать окна диалога, а внести нужные изменения в структуру таблицы непосредственно в окне диалога:

1. Находясь в окне диалога «Схема данных», установите указатель мыши на модифицируемую таблицу.
2. Нажмите правую кнопку мыши и выберите из контекстного меню команду **Конструктор таблиц**.
3. Внесите в структуру таблицы необходимые изменения.
4. Закончив внесение изменений, нажмите кнопку закрытия окна в строке заголовка окна диалога. В ответ на запрос о сохранении изменений выберите **Да** для сохранения изменений и возвращения в окно диалога «Схема данных».

Изменение существующей связи

Прежде чем приступить к изменению связей между таблицами, закройте все открытые таблицы. не допускает изменение связей между открытыми таблицами. Затем выполните следующую последовательность действий:

1. Находясь в окне базы данных, нажмите кнопку **Схема данных** на панели инструментов,

2. Если таблицы, связи между которыми требуется изменить, не отображаются в окне диалога «Схема данных», нажмите кнопку **Добавить таблицу** на панели инструментов, установите указатель на имя нужной таблицы и дважды нажмите кнопку мыши. После этого нажмите кнопку **Закрыть,**

3. Установите указатель на линию связи, которую требуется изменить, и дважды нажмите кнопку мыши.

4. В открывшемся окне диалога «Связи» внесите нужные изменения и нажмите кнопку **ОК.**

Удаление связи

Нажмите кнопку **Схема данных** на панели инструментов, установите указатель на линию связи, которую требуется удалить, и выделите ее, нажав кнопку мыши. Нажмите клавишу *Delete.* Когда предложит вам подтвердить удаление связи, нажмите кнопку **Да.**

Удаление таблицы из макета схемы данных

1. Откройте окно диалога «Схема данных». 2. Выберите таблицу, которую требуется удалить из данного окна, и нажмите клавишу *Delete.* Таблица будет удалена из макета схемы данных вместе с определенными для нее связями. Данная операция изменяет только макет в окне диалога «Схема данных». И таблица, и ее связи будут по-прежнему сохраняться в базе данных.

Определение условий целостности данных

Целостность данных является одним из самых важных требований, предъявляемых к базам данных. Для задания условий целостности данных служат установленные между таблицами отношения. Условиями целостности данных называют набор правил, используемых в MS для поддержания связей между записями в связанных таблицах. Эти правила делают невозможным случайное удаление или изменение связанных данных Условия целостности данных выполняются при следующих условиях:

1) Связанное поле главной таблицы является ключевым полем или имеет уникальный индекс. 2) Связанные поля имеют один тип данных. 3) Обе таблицы принадлежат одной базе данных .

Если таблицы являются присоединенными таблицами, то они должны быть таблицами MS . Невозможно определить условия целостности данных для присоединенных таблиц из баз данных других форматов.

При определении условия целостности данных действуют следующие ограничения:

- Невозможно ввести в поле внешнего ключа связанной таблицы значение, не содержащееся в ключевом поле главной таблицы. Однако возможен ввод в поле внешнего ключа пустых значений, показывающих, что записи не являются связанными.

- Не допускается удаление записи из главной таблицы, если существуют связанные с ней записи в подчиненной таблице.

- Невозможно изменить значение ключевого поля в главной таблице, если имеются записи, связанные с этой записью. Например, невозможно удалить код сотрудника в таблице Сотрудники, если в таблице Заказы имеются заказы, относящиеся к данному сотруднику.

Определение целостности данных предполагает выполнение следующих действий:

1) В окне диалога «Схема данных» два раза мышью на линии связи между двумя таблицами. Откроется окно диалога «Связь». 2) Установите флажок **Обеспечение целостности данных** и нажмите **ОК.**

Для того чтобы преодолеть ограничения на удаление или изменение связанных записей, сохраняя при этом целостность данных, следует включить режимы каскадного обновления и каскадного удаления. При установленном флажке **Каскадное обновление связанных полей** изменение значения в ключевом поле главной таблицы приводит к автоматическому обновлению соответствующих значений во всех связанных записях. При установленном флажке **Каскадное удаление связанных записей** удаление записи в главной таблице приводит к автоматическому удалению связанных записей в подчиненной таблице.

Использование каскадных операций

Обратимся к окну диалога «Связи». При установке опции **Обеспечение целостности данных** вам стали доступны опции **Каскадное обновление связанных полей** и **Каскадное удаление связанных нолей**. При выборе этих опций, выполняет изменения в связанных таблицах таким образом, чтобы сохранить целостность данных, даже если вы изменяете значения ключевых полей или удаляете запись в главной таблице.

Каскадные изменения

Может возникнуть необходимость внести изменения в ключевое поле записи, связанной по типу «один-ко-многим». Если вы не установили опцию **Каскадное обновление связанных полей, то** не позволит этого сделать. Вместо изменения выдаст вам предупреждение о том, что вы нарушите целостность данных, если попытаетесь осуществить это изменение. Это является нарушением потому что это поле является первичным ключом в связи «один-ко-многим», и изменяемое значение появляется, по крайней мере, в одной записи в поле связанной с ним таблицы. Если же вы установили **Каскадное обновление связанных полей,** проблем не возникнет. Вы можете ввести изменения. выполнит все необходимые изменения в связанных таблицах автоматически.

Каскадные удаления

При попытке удалить запись в ключевом поле записи, связанной по типу «один-ко-многим» при не установленной опции **Каскадное удаление связанных полей**, выдаст сообщение об ошибке.

Однако, если выбрана опция **Каскадное удаление связанных полей,** вы можете удалить запись из главной таблицы. Связанные с нею записи в подчиненных таблицах будут также автоматически удалены, соблюдая ,таким образом, правила целостности данных. Другими словами, если выбрана опция **Каскадное удаление связанных полей** то все связанные поля из подчиненной таблицы удаляются, как только удаляется запись из главной таблицы.

3. ОБЪЕКТЫ И СРЕДСТВА ИССЛЕДОВАНИЯ

Предметная область в соответствии с вариантом задания. ПК типа IBM PC, дисплейный класс, операционная система Windows XP и выше, СУБД OO Base.

4. ПОДГОТОВКА К РАБОТЕ

Ознакомиться с теоретическими положениями лабораторной работы.

5. ПРОГРАММА РАБОТЫ

Установите отношения между таблицами. Определите условия целостности данных. Оформить отчет.

ВАРИАНТЫ ЗАДАНИЙ . Выбирается преподавателем из списка заданий (См. Приложение) и остается единым для всех лабораторных работ.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

6.1 Как формируются отношения между реляционными таблицами?

6.2. Опишите технологию создания схемы данных?

6.5. Что такое каскадное изменение и удаление ?

Лабораторная работа № 5

СОЗДАНИЕ ФОРМ

1. ЦЕЛЬ РАБОТЫ

Изучение средств автоматизации конструирования форм просмотра и редактирования данных в СУБД.

2. ОСНОВЫ ТЕОРИИ

Форма позволяет объединить поля в группы по определенным признакам. Это облегчает восприятие информации.

Простейший способ создания формы

Для создания формы любым способом на начальном этапе необходимо выполнить следующие действия:

1. Открыть окно БД. 2. Перейти на вкладку Формы. 3. Нажать кнопку **Создать** . Выбрать нужный вариант. Ок.

Можно воспользоваться мастером форм и проделать следующие операции:

1. Открыть окно БД. 2. В окне перейти на вкладку «Таблицы». 3. Указать на таблицу, для которой создается экранная форма. 4. Выполнить команду **Вставка|Автоформа** или нажать кнопку **Новый объект** и из списка выбрать опцию **Автоформа.** На экране появится готовая форма.

Создание формы в конструкторе форм

В данном разделе обсудим создание формы с помощью конструктора форм. Процесс по созданию формы может включать в себя все или часть из приведенных ниже процедур:

- Размещение текста.
- Размещение полей.
- Создание управляющих кнопок.
- Размещение линий, прямоугольников и рисунков.
- Установка цвета объектов формы.
- Перемещение объектов формы.

Наиболее частым источником ошибок в базе данных является ввод пользователем неправильных данных. Тщательно разработав форму, которую пользователи будут использовать для ввода, редактирования или просмотра данных, можно предотвратить возникновение большого количества ошибок. При создании форм учитывайте следующее

- Если пользователи привыкли к использованию стандартных бланков, формы должны выглядеть так же, как эти бланки. Необходимость каждый

раз искать местонахождение информации никогда не способствовала безошибочному вводу данных.

- Для группировки элементов управления используйте линии и прямоугольники. При этом пользователь будет вводить близкие по смыслу данные (такие, как вся идентификационная информация по товару или полный домашний адрес) вместе.

- Не концентрируйте элементы управления в какой-либо части формы. Это затрудняет чтение информации. Пользователь должен наглядно видеть, с каким элементом управления он работает в данное время.

- Пояснительный текст формы должен быть максимально информативным и иметь минимальную длину.

- Используйте условия правильности ввода данных что поможет предотвратить ввод неверных данных.

- Используйте маски ввода для ввода стандартизированной информации. Например, если вам известно точное количество символов, необходимых для ввода номера телефона, используйте маску ввода.

- Для облегчения восприятия чисел используйте форматы ввода.

Настройка формы

Для создания формы войдите в окно конструктора форм. Для этого, находясь на вкладке «Формы» БД нажмите кнопку **Создать** и в открывшемся окне диалога «Новая форма» нажмите кнопку **ОК.** На экране откроется окно конструктора форм.

Первое, что нужно сделать — определить свойства самой формы как объекта. Каждая форма имеет свойства, определяющие расположение ее в основном окне , размер, заголовок, стиль и некоторые другие параметры.

1. Для определения или изменения стиля формы, находясь в конструкторе форм, выполните команду **Формат|Автоформат** или нажмите кнопку **Автоформат** на панели инструментов На экране откроется окно диалога «Автоформат» . Выберите из списка стили и посмотрите на их внешний вид в окне просмотра После того как вы нашли нужный стиль, нажмите кнопку ОК.

2. Для задания размеров формы используйте мышь. Установите указатель мыши в нижний правый угол формы. При этом курсор примет вид двунаправленной стрелки. Нажмите кнопку мыши и установите требуемый размер формы.

3. Для настройки остальных параметров формы откройте окно ее свойств, выполнив команду **Вид | Свойства**. На экране появится окно свойств со стандартными значениями свойств формы .

4. Перейдите на вкладку «Макет» окна свойств. На этой вкладке, редактируя свойства формы, вы можете задать различные параметры окна формы, например:

• Наличие кнопок оконного меню в верхнем правом углу с помощью свойства **Кнопка оконного меню**

• Наличие кнопок свертывания и развертывания окна с помощью свойства **Кнопки размеров** окна

• Отображение кнопки закрытия формы с помощью свойства **Кнопка закрытия**

• Расположение формы в центре окна приложения установкой значения свойства **Выравнивание по** центру равным **Да**

• Заголовок окна вводом текста заголовка в поле свойства Подпись

5. Для связывания формы с таблицей или запросом используется свойство **Источник записей** вкладки «Данные». Это свойство определяет в качестве источника данных для формы таблицу или запрос. Нажмите кнопку раскрытия списка и из списка таблиц и запросов выберите тот источник записей, для которого вы хотите создать форму.

6. Для использования формы только для просмотра, установите для свойства **Разрешить изменение** значение **Нет**. Данная установка не позволит пользователю редактировать содержимое элементов управления, связанных с таблицей или запросом.

7. Если установить для свойства **Разрешить добавление** и **Разрешить добавление** значения **Нет,** то становятся недоступными команды **Вставка|Запись, Записи|Ввод данных и Правка|Удалить запись.**

Размещение текстовой информации

Размещение текста в экранной форме осуществляется с помощью инструмента **Надпись,** который находится на панели элементов. Под текстом понимается любая текстовая информация: заголовки, поясняющая информация. Для размещения текста в форме выполните следующие действия:

1. Выберите инструмент **Надпись** на панели элементов. Если данная панель отсутствует на экране, для ее отображения выполните команду **Вид |Панели инструментов** и в списке панелей инструментов установите опцию **Панель элементов** или нажмите кнопку **Панель элементов** на панели инструментов.

2. Установите указатель мыши на место предполагаемого расположения текстового объекта и введите текст.

3. Закончив ввод текста, нажмите клавишу *Enter*.

4. Выделите созданный объект.

5. Используя панель инструментов «Форматирование» или окно свойств созданного объекта, задайте для него тип шрифта, размер, цвет шрифта, цвет рамки, тип и цвет фона и другие параметры оформления. Все свойства вы можете определить в окне свойств. Некоторые, наиболее часто используемые, можно задавать и с помощью панели «Форматирование»

Размещение полей ввода

Следующим шагом в создании формы является добавление в нее полей различных типов. Наиболее простым типом поля является поле ввода. Для размещения поля ввода в форме выполните следующие действия:

1. Выберите инструмент **Поле** на панели элементов.

2. Нажмите мышью место, в котором вы предполагаете разместить поле. В форме появится связанный объект, состоящий из поля ввода и его надписи. Выделите поле ввода и откройте для него окно свойств.

3. Чтобы связать созданное поле с полем таблицы или запроса выберите свойство **Данные** вкладки «Данные». В поле ввода свойства воспользуйтесь кнопкой раскрытия списка и выберите из списка всех полей открытой таблицы поле, которое хотите добавить в форму. Если вы хотите связать поле с выражением, нажмите кнопку **Построить.** Создайте необходимую формулу с помощью построителя выражений.

4. Используя панель инструментов форматирования или окно свойств поля ввода, задайте для него тип шрифта, размер, цвет шрифта, цвет рамки, тип, цвет фона и другие параметры.

5. Если вы создаете поле, информация из которого должна быть доступна только для чтения, необходимо установить значение свойства **Доступ** равным **Нет.**

6. Свойство **Всплывающая подсказка** вкладки «Другие» позволяет создать краткое пояснение к полю, которое будет появляться на экране, когда указатель установлен на поле и удерживается на нем некоторое время.

7. Для определения значения поля по умолчанию задайте свойство **Значение по умолчанию.**

8. Выделите надпись к полю ввода и откройте для него окно свойств.

9. Чтобы задать текст надписи, выберите свойство **Подпись** вкладки «Макет» и в поле ввода свойства введите текст надписи к полю.

10. Используя панель инструментов «Форматирование» или окно свойств, задайте для надписи тип шрифта, размер, цвет шрифта, цвет рамки, тип, цвет фона и другие параметры.

Скрытие поля

Используя окно свойств поля, вы можете сделать поле невидимым (и поле ввода и надпись к нему), так что его не будет видно в режиме просмотра формы. Аналогичная возможность существует в режиме просмотра таблиц, в котором для скрытия столбца таблицы используется команда **Формат | Скрыть столбцы.** В форме для скрытия поля и надписи к нему необходимо для свойства **Вывод на экран** вкладки «Макет» поля задать значение **Нет.**

Отображение сообщений в строке состояния

При создании таблиц упоминалось, что информация, помещенная в столбец «Описание», в режиме просмотра формы выводится в строке состояний, когда курсор находится в данном поле. Если этот столбец пуст, то сообщение не выводится.

Вы можете изменить сообщение в форме с помощью свойства **Текст строки состояния** вкладки «Другие». Для этого откройте окно свойств поля

и введите в поле ввода данного свойства текст, который будет появляться в строке состояния.

Изменение формата отображения дат и чисел

Для задания формата отображения дат и/или чисел в форме используется свойство **Формат поля** вкладки «Макет».

Размещение списка и раскрывающегося списка

При вводе информации в таблицу во многих случаях удобнее выбирать повторяющееся значение из списка, чем каждый раз вводить одно из двух-трех значений. Кроме того, выбор из списка позволяет быть уверенным, что введенное значение является допустимым.

Элементы панели элементов **Список** и **Поле со списком** предназначены для отображения на экране элементов списка. Выбор одного из этих двух типов объектов определяется требованиями пользователя и наличием свободного места в форме.

При небольшом числе элементов списка удобнее использовать **Список**, который всегда раскрыт на экране. Когда список большой, используйте элемент **Поле со списком**. Он занимает меньше места на экране, поскольку список раскрывается на экране только при нажатии кнопки раскрытия списка.

Список и **Поле со списком** состоят из строк данных. Строки содержат один или несколько столбцов, определяемых с помощью следующих средств:

- Список фиксированных значений.
- Список полей.
- Значения поля таблицы или запроса.

Тип источника данных определяется свойством **Тип источника** строк вкладки «Данные»,

Объекты типа списка и поля со списком имеют дополнительные свойства, которые отсутствовали у ранее рассмотренных объектов Надпись и Поле.

Для размещения списка, поля со списком, группы параметров и ряда других элементов управления вы можете использовать мастера. Для этого вам необходимо установить режим использования мастера и выбрать соответствующий инструмент на панели элементов. После указания места расположения элемента запускается соответствующий мастер.

Размещение флажка

Для индикации состояния, которое может иметь только одно из двух допустимых значений, используются *флажки*. Они могут использоваться по одному или группами. Установленный флажок будет соответствовать значению «постоянный», а снятый — значению «временный».

Рассмотрим последовательность ваших действий при создании флажка для редактирования поля , которое имеет тип **Логический.**

1. Откройте форму в режиме конструктора и выберите инструмент **Флажок** на панели элементов.
2. Нажмите мышью место предполагаемого размещения элемента в форме.
3. Откройте окно свойств размещенного в форме флажка.
4. Чтобы связать созданное поле с полем таблицы, выберите свойство **Данные** вкладки «Данные». В поле ввода значения свойства воспользуйтесь кнопкой раскрытия списка и выберите поле из списка полей таблицы.
5. Выделите надпись созданного флажка и в его окне свойств скорректируйте свойство **Подпись** вкладки «Макет». Просмотрите форму в режиме формы.

Для создания объектов логического типа используются элементы **Переключатель** и **Выключатель** панели элементов, соответственно.

Создание кнопок управления

Кнопки используются в формах для выполнения определенного действия или ряда действий. Например, можно создать в форме кнопку, открывающую другую форму, или создать набор кнопок для перемещения по записям таблицы, если вас не устраивают стандартные средства перемещения, предусмотренные в форме. Для того чтобы кнопка выполняла какое-либо действие, необходимо создать макрос или процедуру обработки события и связать их со свойством кнопки **Нажатие кнопки.**

В предусмотрено создание более 30 разных кнопок, что избавляет пользователя от необходимости самостоятельно разрабатывать макросы. Достаточно лишь воспользоваться мастером по созданию кнопки. Так для создания кнопки, открывающей другую форму нужно выполнить следующие действия:

1. Установите режим использования мастера на панели элементов и выберите инструмент **Кнопка** на панели элементов.
2. Установите указатель мыши на место в форме, в котором вы предполагаете расположить кнопку и нажмите кнопку мыши. Запускается мастер создания **кнопки конструктора форм.**
3. В первом окне диалога расположены два списка: Категории и **Действия**. При перемещении по списку Категории список Действия обновляется. Рассмотрите внимательно содержимое этих списков. Список **Категории** содержит наборов действий, а список **Действия** — действия, которые будут выполняться при нажатии на данную кнопку. Выберите нужное значение из списка **Действия** и нажмите кнопку **Далее**.
4. Во втором окне диалога мастера из списка форм базы данных выберите форму, которая будет открываться при нажатии на кнопку.
5. На следующем шаге определяется тип отображаемой информации на кнопке: текстовая или графическая.

Если вы выбрали текстовую информацию, то в поле ввода рядом с опцией **Текст** введите текст, отображаемый на кнопке. При размещении графической информации установите опцию **Рисунок** и нажмите кнопку **Обзор** для открытия окна диалога «Выбор рисунка», в котором выберите графическое изображение. После чего переходите к следующему шагу.

6. На заключительном шаге работы мастера задается имя создаваемого объекта Введите имя, затем нажмите кнопку **Готово**. Кнопка, при нажатии на которую открывается другая экранная форма, создана.

Добавление рисунка или другого объекта в форму

Способ вставки рисунка или объекта зависит от того, какой объект предполагается создать: присоединенный или свободный. Присоединенный объект хранится в таблице. При переходе к новой записи в форме или отчете отображается другой объект. Например, этот способ удобен для хранения фотографий сотрудников фирмы. Свободный объект является частью структуры формы. При переходе к новой записи объект не изменяется. Например, таким способом в форме или отчете сохраняют эмблему фирмы, созданную в приложении MS Paint.

Размещение графического изображения

В экранные формы можно вставлять различные графические изображения, позволяющие облегчать восприятие информации. Для этого используется инструмент **Рисунок** панели элементов. При размещении данного элемента в форме открывается окно диалога «Выбор рисунка» . Используя список **Тип файла,** укажите тип используемого изображения. Выбрав имя вставляемого рисунка, нажмите кнопку **ОК.**

Размещение объекта типа OLE

Примером использования данного типа объектов является расположение в форме поля Фотография таблицы Сотрудники. В данном поле таблицы хранятся фотографии всех сотрудников фирмы.

Для присоединенного объекта в форме выполните следующие действия:

1. Для добавления графического поля типа OLE в форму выберите инструмент **Присоединенная рамка объекта** на панели элементов.

2. Нажмите мышью место, где вы хотите добавить поле. Удерживая кнопку мыши в нажатом состоянии, переместите указатель по диагонали так, чтобы получилась рамка требуемого размера.

3. Откройте окно свойств созданного объекта.

4. Чтобы связать созданное поле с полем таблицы, выберите свойство **Данные. В** поле ввода свойства воспользуйтесь кнопкой раскрытия списка и из списка полей открытой таблицы Сотрудники выберите поле типа OLE Фотография.

5. Просмотрите форму в режиме формы. Если рисунок не помещается в рамке целиком, вернитесь в режим конструктора и увеличьте размер поля.

Использование линий и прямоугольников

Линии и прямоугольники в экранной форме применяются дня улучшения внешнего вида формы и восприятия информации, а также для объединения объектов в логические группы.

Для добавления в экранную форму линий используется инструмент **Линия** панели элементов. Чтобы нарисовать вертикальную или горизонтальную линию на экране выберите данный инструмент, нажмите мышью то место, где должна начинаться линия, и переместите указатель до получения линии нужной длины. Настройка параметров линии осуществляется с помощью ее свойств.

Для добавления в экранную форму прямоугольников используется инструмент **Прямоугольник** панели элементов. Чтобы нарисовать прямоугольник выберите данный инструмент, нажмите мышью место расположения одного из углов прямоугольника, и переместите указатель до получения прямоугольника нужного размера.

3. ОБЪЕКТЫ И СРЕДСТВА ИССЛЕДОВАНИЯ

Предметная область в соответствии с вариантом задания. ПК типа IBM PC, дисплейный класс, операционная система Windows XP и выше, СУБД OO Base.

4. ПОДГОТОВКА К РАБОТЕ

Ознакомиться с теоретическими положениями лабораторной работы.

5. ПРОГРАММА РАБОТЫ

Разработайте формы для часто встречающихся групп данных с использованием конструктора форм. Добавьте для наглядности в форму подходящее графическое изображение. Если ваши данные отображаются в виде графика, то постройте его в форме. Оформить отчет.

ВАРИАНТЫ ЗАДАНИЙ. Выбирается преподавателем из списка заданий (См. Приложение) и остается единым для всех лабораторных работ.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

6.1. Каково назначение форм в базах данных ?

6.2. Опишите технологию создания простейших форм в СУБД ?

6.3. Назовите элементы управления формами и способы их реализации?

Лабораторная работа № 6

ФОРМИРОВАНИЕ ОТЧЕТОВ

1. ЦЕЛЬ РАБОТЫ

Изучение средств автоматизации конструирования отчетов в СУБД.

2. ОСНОВЫ ТЕОРИИ

Отчет (report) — это объект базы данных, который используется для вывода на экран, в печать или файл структурированной информации. Reports позволяют извлечь из таблиц или запросов базы данных необходимую информацию и представить ее в виде удобном для восприятия. Report содержит заголовок, область данных, верхний и нижний колонтитулы, примечание и разбит на страницы.

Для создания **отчетов** можно использовать различные средства (рис. 1):

- Мастер отчетов
- Конструктор отчетов
- Инструмент Report
- Пустой report

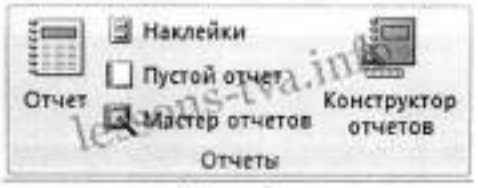


Рис. 1.

Отчеты целесообразно выполнять с помощью Мастера или других указанных инструментов, а дорабатывать их, т.е. вносить необходимые изменения можно в режиме макета или конструктора. В Microsoft 2007 предусмотрено два режима внесения изменений и дополнений в reports: режим макета и режим конструктора.

Режим макета — это более наглядный режим редактирования и форматирования (изменения) отчетов, чем режим конструктора. В тех случаях, когда в режиме макета невозможно выполнить изменения в отчете, то целесообразно применять режим конструктора.

Мастер отчетов. Для создания отчета при помощи Мастера отчетов необходимо выполнить следующие действия:

- В окне базы данных щелкнуть на вкладке Создание и затем щелкнуть на кнопке Мастер отчетов в группе Отчеты. Появится диалоговое окно Создание отчетов.
- В поле Таблицы и отчеты щелкнуть на стрелке и выбрать в качестве источника данных таблицу Студенты.
- Щелкнуть на кнопке ОК (в результате получим вид окна "Создание отчетов", представленный на рис. 2).

- Все "Доступные поля" переведем в "Выбранные поля", выделив их и щелкнув на кнопку >>.
- На следующем шаге (Добавить уровни группировки?) щелкаем далее.
- На шаге "Выберите порядок сортировки записей". В раскрывающемся списке выберем "Фамилия" для сортировки по возрастанию.
- На шаге "Выберите вид макета для отчета". Выбираем: Макет - блок, ориентация - книжная. Щелкнуть на кнопке Далее.
- На шаге " Выберите требуемый стиль". Выбираем - Изящная.
- Следующий шаг - "Задайте имя отчета". Вводим имя - Студенты мастер_отчетов. Дальнейшие действия: Просмотреть report; Изменить макет отчета. Выбираем Просмотреть, щелкаем на кнопке Готово. Report открывается в режиме Предварительного просмотра, который позволяет увидеть, как будет выглядеть report в распечатанном виде (Рис. 3).
- Перейдите в режим Конструктора и выполните редактирование и форматирование отчета. Для перехода из режима предварительного просмотра в режим конструктора необходимо в области переходов щелкнуть правой кнопкой мыши на имени отчета и в контекстном меню выбрать режим конструктора. На экране появится report в режиме Конструктора (Рис. 4).

Рис. 2.

Инструмент Отчет. Для быстрого создания отчета, т.е. создания одним щелчком мыши можно воспользоваться инструментом Report. В этом случае report формируется на базе существующей таблицы или запроса. В созданном отчете будут отображаться все записи таблицы или запроса, на базе которых создается report. Но созданный report можно будет изменить в режиме макета или конструктора.

Для создания отчета необходимо выполнить следующее. В области переходов надо выделить таблицу (например, Студенты), на основе которой нужно создать report. Затем перейти на вкладку Создание и щелкнуть на пикто-

грамме Report. На экране будет отображен простой Отчет на основе текущей таблицы Студенты.

Средство Пустой отчет. Инструмент "Пустой report" позволяет создавать reports с нуля в режиме макета. Для этого надо щелкнуть Пустой report в группе Отчеты на вкладке Создание. В окне редактирования 2007 появится Отчет1 с пустой областью данных, а в правой части окна будет отображаться область "Список полей" существующих таблиц. Щелкнув на знак "+" таблицы (например, Студенты), откроется список необходимых полей.

Перетащите требуемые поля из этого списка в report, нажав и удерживая левую клавишу мыши. С помощью инструментов из группы "Элементы управления" на вкладке Формат, можно доработать report, добавив заголовок, номера страниц, дату и время. При необходимости его можно доработать в режиме конструктора. Сохраните report

3. ОБЪЕКТЫ И СРЕДСТВА ИССЛЕДОВАНИЯ

Предметная область в соответствии с вариантом задания. ПК типа IBM PC, дисплейный класс, операционная система Windows XP и выше, СУБД OO Base.

4. ПОДГОТОВКА К РАБОТЕ

Ознакомиться с теоретическими положениями лабораторной работы.

5. ПРОГРАММА РАБОТЫ

Разработайте отчеты для представления информации. Оформить отчет.

ВАРИАНТЫ ЗАДАНИЙ . Выбирается преподавателем из списка заданий (См. Приложение) и остается единым для всех лабораторных работ.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

6.1. Каково назначение отчетов в базах данных ?

6.2. Опишите технологию создания простейших отчетов в СУБД?

6.3. Назовите структуру отчетов и способы их реализации?

Лабораторная работа № 7

СОЗДАНИЕ МЕНЮ. ГЕНЕРАЦИЯ, ЗАПУСК

1. ЦЕЛЬ РАБОТЫ

Изучение средств автоматизации конструирования меню в СУБД.

2. ОСНОВЫ ТЕОРИИ

Главная кнопочная форма создается с целью навигации по базе данных. Эта форма может использоваться в качестве главного меню БД. Элементами главной кнопочной формы являются объекты форм и отчётов.

Запросы и таблицы не являются элементами главной кнопочной формы. Поэтому для создания кнопок Запросы или Таблицы на кнопочной форме можно использовать макросы. Сначала в окне базы данных создают макросы «Открыть Запрос» или «Открыть Таблицу» с уникальными именами, а затем в кнопочной форме создают кнопки для вызова этих макросов.

Для одной базы данных можно создать несколько кнопочных форм. Кнопки следует группировать на страницах кнопочной формы таким образом, чтобы пользователю было понятно, в каких кнопочных формах можно выполнять определенные команды (запросы, отчеты, ввода и редактирования данных). Необходимо отметить, что на подчиненных кнопочных формах должны быть помещены кнопки возврата в главную кнопочную форму.

Технология создания кнопочных форм следующая:

- создать страницу главной кнопочной формы (ГКФ);
- создать необходимое количество страниц подчиненных кнопочных форм (например, формы для ввода данных, для отчетов, для запросов и т.д.);
- создать элементы главной кнопочной формы;
- создать элементы для кнопочных форм отчетов и форм ввода или изменения данных;
- создать макросы для запросов или для таблиц с уникальными именами;
- создать элементы для кнопочных форм запросов или таблиц.

Структура кнопочных форм может быть представлена в следующем виде.

Рис. 1.

Для создания главной кнопочной формы и ее элементов необходимо открыть базу данных, (например, «Успеваемость_ студентов») и выполнить команду Сервис / Служебные программы / Диспетчер кнопочных форм. Если кнопоч-

ная форма ранее не создавалась, то откроется окно диалога «Диспетчер кнопочных форм».

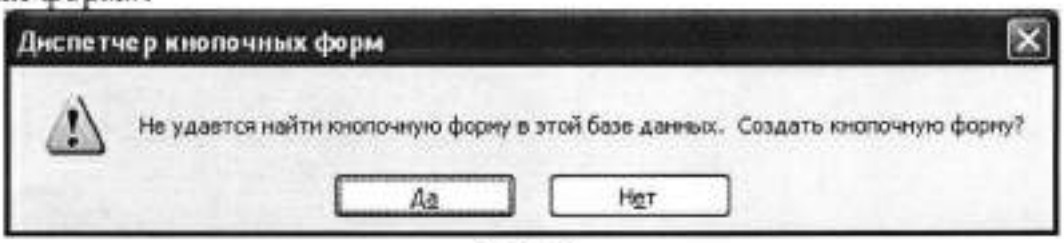


Рис. 2.

В окне диалога надо нажать кнопку «Да», тем самым подтвердить создание кнопочной формы. В результате будет создана страница Главной кнопочной формы.

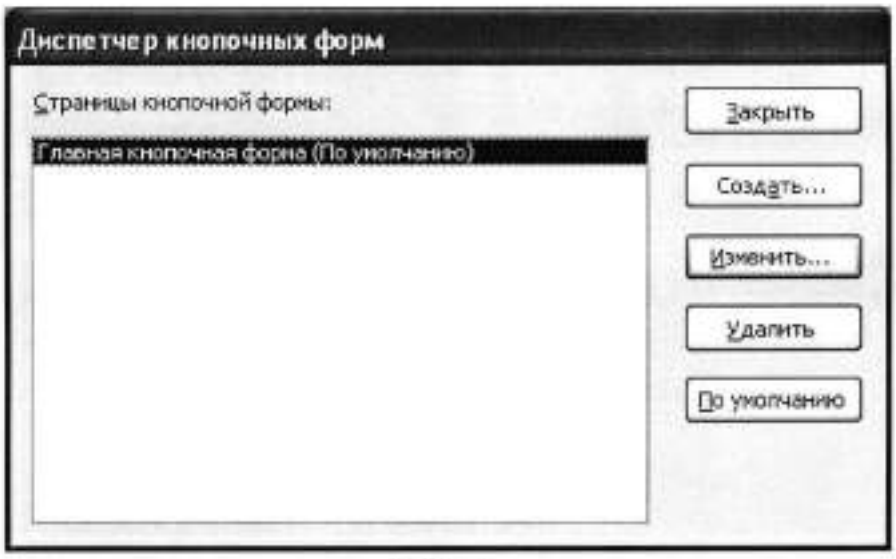


Рис. 3.

Далее можно создать еще три страницы кнопочной формы: Формы ввода данных, Отчеты и Запросы. Для этого следует щелкнуть на кнопке «Создать» и в появившемся окне ввести имя новой страницы «Формы ввода данных» и щелкнуть на кнопке «ОК».

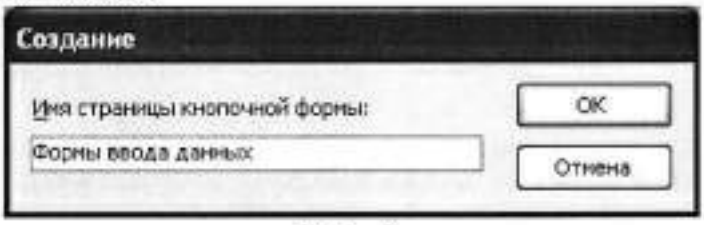


Рис. 4.

Будет создана страница кнопочной формы «Формы ввода данных». Аналогичным образом надо создать еще две страницы, в итоге получим четыре страницы кнопочных форм, которые отображаются в окне «Диспетчер кнопочных форм».

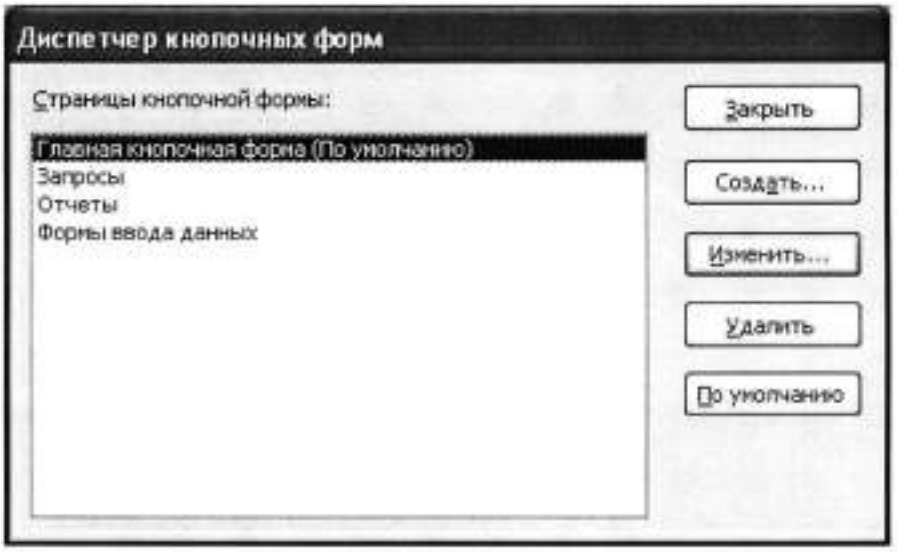


Рис. 5.

После этого создаем элементы ГКФ, для этого в «Окне диспетчер кнопочных форм» выделяем страницу «Главная кнопочная форма» и щелкаем «Изменить», откроется новое окно «Изменение страниц кнопочной формы».

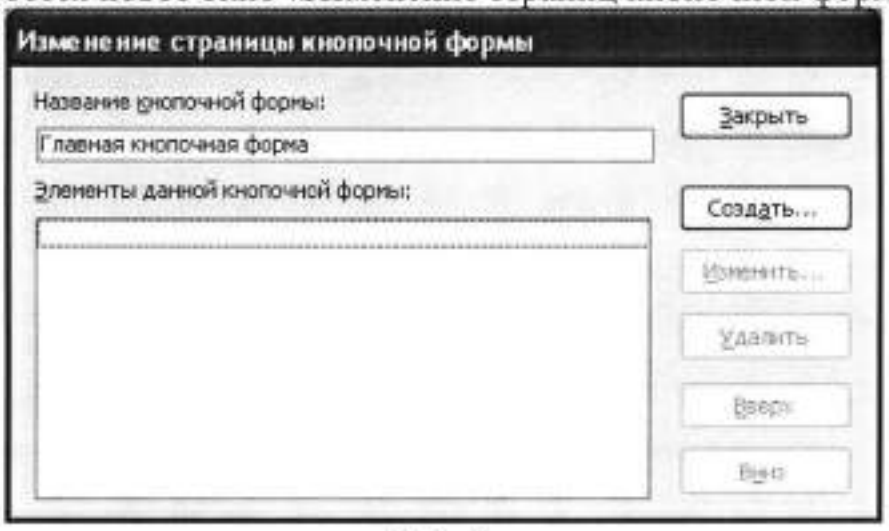


Рис. 6.

В этом окне щелкаем на кнопке «Создать», откроется новое окно «Изменение элемента кнопочной формы».

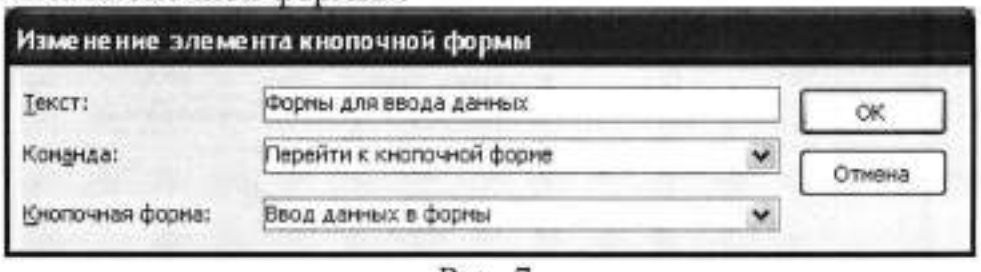


Рис. 7.

В окне выполняем следующее:

- вводим текст: Формы для ввода данных;
- выбираем из раскрывающегося списка команду: Перейти к кнопочной форме;
- выбираем из списка кнопочную форму: Ввод данных в формы, щелкаем на кнопке «ОК».

В окне «Изменение страницы кнопочной формы» отобразится элемент кнопочной формы «Формы для ввода данных».

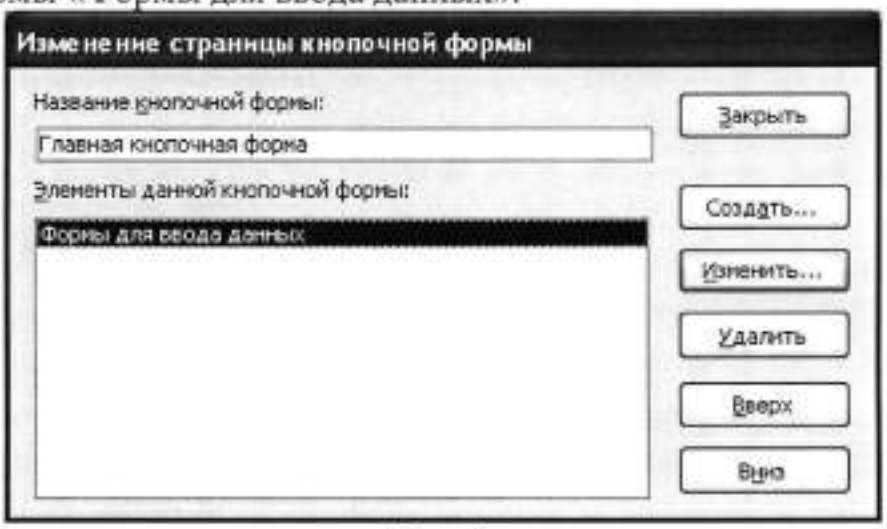


Рис. 8.

Аналогичным методом надо создать элементы: «Запросы» и «Отчеты», а затем элемент (кнопку) "Выход из БД".

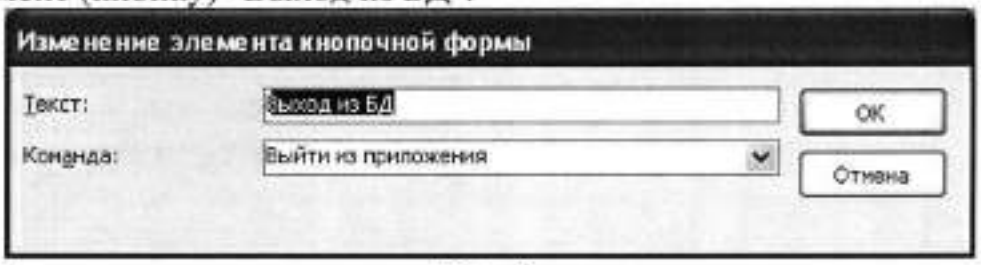


Рис. 9.

В результате в окне «Изменение страницы кнопочной формы» будут отображаться все элементы главной кнопочной формы.

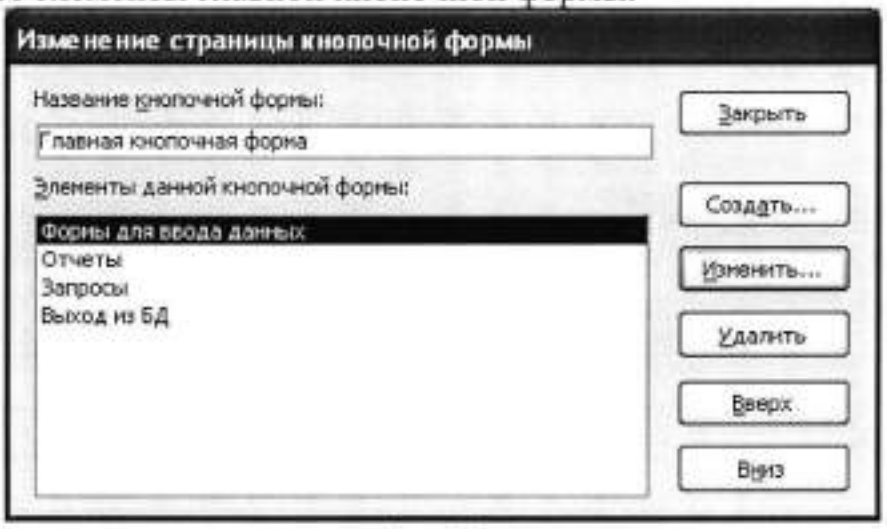


Рис. 10.

Кнопочная форма появится в списке в области окна базы данных на вкладке Формы на панели Объекты, а на вкладе Таблицы в списках появится таблица Switchboard Items. Дважды щелкнув на надписи "Кнопочная форма", откроется Главная кнопочная форма.

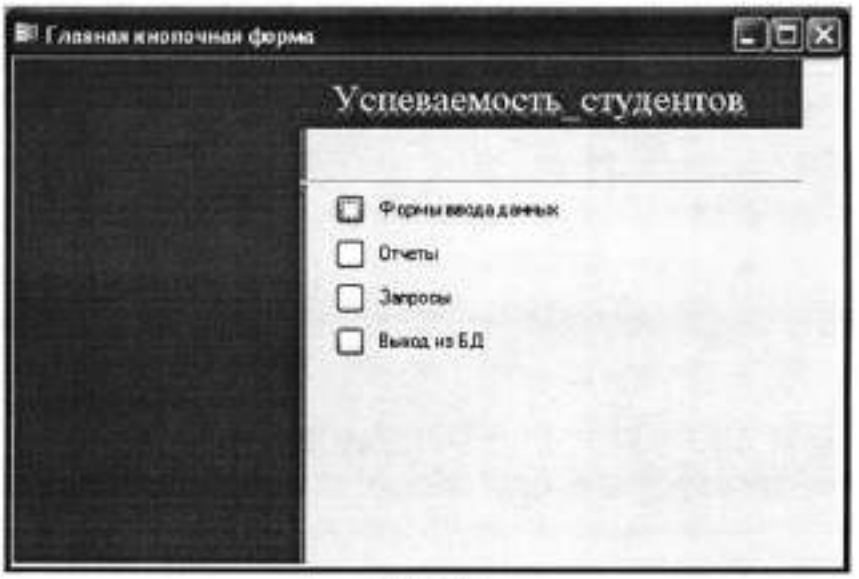


Рис. 11.

Для того чтобы эта форма отображалась при запуске базы данных, необходимо выполнить команду Сервис/Программы запуска, и в открывшемся окне выбрать "Кнопочная форма" из раскрывающегося списка, кроме того, надо снять флажки Окно базы данных и Строка состояния. Можно также ввести заголовок и значок приложения.

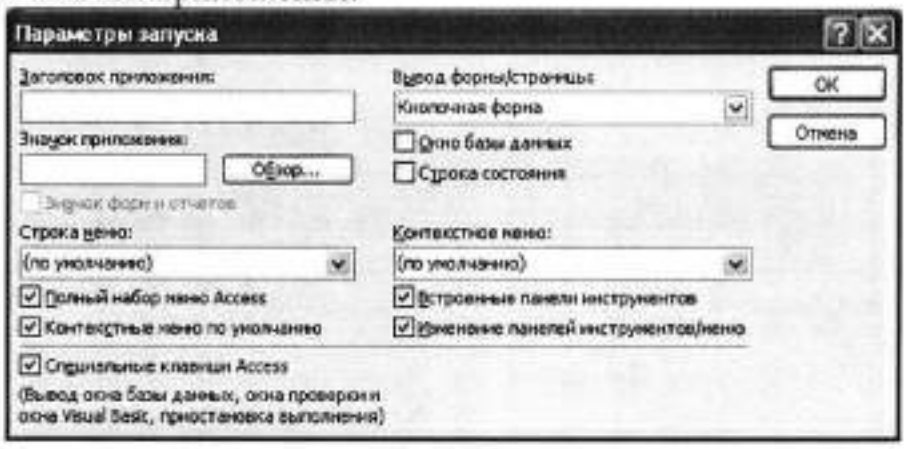


Рис. 12.

Но на этом создание кнопочных форм еще не законченно, так как на подчиненных кнопочных формах (Формы ввода данных, Отчеты, Запросы) нет элементов. Каким образом поместить элементы на подчиненные формы рассмотрим в следующем разделе.

3. ОБЪЕКТЫ И СРЕДСТВА ИССЛЕДОВАНИЯ

Предметная область в соответствии с вариантом задания. ПК типа IBM PC, дисплейный класс, операционная система Windows XP и выше, СУБД OO Base.

4. ПОДГОТОВКА К РАБОТЕ

Ознакомиться с теоретическими положениями лабораторной работы.

5. ПРОГРАММА РАБОТЫ

Разработайте кнопочную форму навигации по функциям БД. Оформить отчет.

ВАРИАНТЫ ЗАДАНИЙ . Выбирается преподавателем из списка заданий (См. Приложение) и остается единым для всех лабораторных работ.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

6.1. Каково назначение кнопочной формы в базах данных ?

6.2. Опишите технологию создания меню в СУБД?

6.3. Определите структуру меню и способы его реализации?

Лабораторная работа №8

СОЗДАНИЕ БАЗЫ ДАННЫХ С ПОМОЩЬЮ КОМАНД SQL. СТРУКТУРЫ И ТИПЫ ДАННЫХ. СТАНДАРТЫ ЯЗЫКА SQL

1. ЦЕЛЬ РАБОТЫ

Получить навыки использования SQL-запросов для решения различных задач.

2. ОСНОВЫ ТЕОРИИ

Запрос SQL — это запрос, создаваемый при помощи инструкций SQL (Инструкция (строка) SQL. Выражение, определяющее команду SQL, например **SELECT**, **UPDATE** или **DELETE**, и включающее предложения, например WHERE или ORDER BY. Инструкции/строки SQL обычно используются в запросах и в статистических функциях.). Язык SQL (Structured Query Language) используется при создании запросов, а также для обновления и управления реляционными базами данных, такими как базы данных Microsoft .

Когда пользователь создает запрос в режиме конструктора запроса, Microsoft автоматически создает эквивалентную инструкцию SQL. Фактически, для большинства свойств запроса, доступных в окне свойств в режиме конструктора, имеются эквивалентные предложения или параметры языка SQL, доступные в режиме SQL (Режим SQL. Окно, в котором выводится инструкция SQL текущего запроса или которое используется для создания запроса SQL (запроса на объединение, запроса к серверу или управляющего запроса). При необходимости, пользователь имеет возможность просматривать и редактировать инструкции SQL в режиме SQL. После внесения изменений в запрос в режиме SQL его вид в режиме конструктора может измениться.

Некоторые запросы SQL невозможно создать в бланке запроса. Для запросов к серверу (запрос SQL, используемый для передачи команд прямо на сервер базы данных ODBC. Запрос к серверу позволяет непосредственно работать с таблицами на сервере вместо обработки их данных с помощью ядра Microsoft Jet), управляющих запросов (запрос SQL, содержащий инструкции, которые позволяют создавать или изменять объекты в базе данных) и запросов на объединение (запрос, в котором оператор **UNION** используется для объединения результатов двух или нескольких запросов на выборку) необходимо создавать инструкции SQL непосредственно в окно запроса в режиме SQL. Для подчиненного запроса (инструкция SQL **SELECT**, расположенная внутри другого запроса на выборку или запроса на изменение) пользователь должен ввести инструкцию SQL в строку **Поле** или **Условие отбора** в бланке запроса.

1. Инструкция SELECT

По этой инструкции ядро базы данных Microsoft Jet возвращает данные из базы данных в виде набора записей.

SELECT [*предикат*] { * | *таблица*.* | [*таблица*.]*поле_1*
[AS *псевдоним_1*] [, [*таблица*.]*поле_2* [AS *псевдоним_2*] [, ...]]}
FROM *выражение* [, ...] [IN *внешняяБазаДанных*]
[WHERE...]
[GROUP BY...]
[HAVING...]
[ORDER BY...]
[WITH OWNER OPTION]

Ниже перечислены аргументы инструкции SELECT:

Элемент	Описание
предикат	Один из следующих предикатов отбора: ALL, DISTINCT, DISTINCTROW или TOP. Предикаты используются для ограничения числа возвращаемых записей. Если они отсутствуют, по умолчанию используется предикат ALL.
*	Указывает, что выбраны все поля заданной таблицы или таблиц.
таблица	Имя таблицы, из которой должны быть отобраны записи.
поле_1, *поле_2*	Имена полей, из которых должны быть отобраны данные. Если включить несколько полей, они будут извлекаться в указанном порядке.
псевдоним_1, *псевдоним_2*	Имена, которые станут заголовками столбцов вместо исходных названий столбцов в *таблице*.
выражение	Имена одной или нескольких таблиц, которые содержат отбираемые данные.
внешняяБазаДанных	Имя базы данных, которая содержит таблицы, указанные с помощью аргумента *выражение*, если они не находятся в текущей базе данных.

Элемент	Описание
ALL	Ядро базы данных Microsoft Jet отбирает все записи, соответствующие условиям, заданным в инструкции SQL.
DISTINCT	Исключает записи, которые содержат повторяющиеся значения в выбранных полях. Чтобы запись была включена в результат выполнения запроса, значения в каждом поле, включенном в инструкцию SELECT, должны быть уникальными. Например, в таблице «Сотрудники» есть однофамильцы. Если две записи содержат значение «Иванов» в поле «Фамилия», то следующая

	инструкция SQL возвратит только одну из них: SELECT DISTINCT Фамилия FROM Сотрудники; Если опустить предикат DISTINCT, этот запрос возвратит обе записи для фамилии Иванов. Если предложение SELECT содержит более одного поля, то для включения записи в результат выполнения запроса необходимо, чтобы совокупность значений во всех этих полях была уникальной. Результат выполнения инструкции SQL, содержащей предикат DISTINCT, является необновляемым и не отражает последующие изменения, внесенные другими пользователями.
DISTINCTROW	Опускает данные, основанные на целиком повторяющихся записях, а не отдельных повторяющихся полях. Следующая инструкция SQL показывает, как можно использовать предикат DISTINCTROW для получения списка клиентов, разместивших хотя бы один заказ, без включения сведений о самих заказах: SELECT DISTINCTROW Название FROM Клиенты INNER JOIN Заказы ON Клиенты.КодКлиента= Заказы.КодКлиента ORDER BY Название; Если опустить предикат DISTINCTROW, в результат выполнения запроса будет включено несколько строк о каждом клиенте, сделавшем несколько заказов. Предикат DISTINCTROW влияет на результат только в том случае, если в запрос включены не все поля из анализируемых таблиц. Предикат DISTINCTROW игнорируется, если запрос содержит только одну таблицу или все поля всех таблиц.
TOP *n* [PERCENT]	Возвращает определенное число записей, находящихся в начале или в конце диапазона, описанного с помощью предложения ORDER BY. Следующая инструкция SQL позволяет получить список 25 лучших студентов выпуска 1994 года: SELECT TOP 25 Имя, Фамилия FROM Студенты WHERE ГодВыпуска = 1994 ORDER BY СреднийБалл DESC; Если предложение ORDER BY будет опущено, запрос возвратит произвольный набор 25 записей из таблицы «Студенты», удовлетворяющих предложению WHERE. Предикат TOP не осуществляет выбор между равными значениями. Если в предыдущем примере средние баллы двадцать пятого и двадцать шестого студента будут равны, то запрос возвратит 26 записей.

Кроме того, можно использовать зарезервированное слово PERCENT для возврата определенного процента записей, находящихся в начале или в конце диапазона, описанного с помощью предложения ORDER BY. Предположим, что вместо 25 лучших студентов следует отобрать студентов, попавших в последние 10 процентов:

```
SELECT TOP 10 PERCENT
Имя, Фамилия
FROM Студенты
WHERE ГодВыпуска = 1994
ORDER BY СреднийБалл ASC;
```

Предикат ASC обеспечивает возврат последних значений. Значение, следующее после предиката TOP должно быть числовым значением типа **Integer** без знака. Предикат TOP не влияет на возможность обновления запроса.

Инструкции SELECT не изменяют данные в базе данных. Обычно слово SELECT является первым словом инструкции SQL. Большая часть инструкций SQL является инструкциями SELECT или SELECT...INTO.
Минимальный синтаксис инструкции SELECT *поля* FROM *таблица*
Для отбора всех полей таблицы можно использовать символ звездочки (*). Если несколько таблиц, включенных в предложение FROM, содержат одноименные поля, перед именем такого поля следует ввести имя таблицы и оператор . (точка).

Предложение FROM

... FROM *выражение* [IN *внешняяБазаДанных*]

Ниже перечислены аргументы инструкции SELECT, содержащей предложение FROM:

Элемент	**Описание**
выражение	Выражение, определяющее одну или несколько таблиц, откуда извлекаются данные. Это выражение может быть именем отдельной таблицы, именем сохраненного запроса или результатом операции INNER JOIN, LEFT JOIN или RIGHT JOIN.

Операция INNER JOIN объединяет записи из двух таблиц, если связующие поля этих таблиц содержат одинаковые значения.

...FROM *таблица_1* INNER JOIN *таблица_2* ON *таблица_1.поле_1* ***оператор*** *таблица_2.поле_2*

Оператор - Любой оператор сравнения: "=", "<", ">", "<=", ">=" или "<>".

Операция LEFT JOIN используется для создания левого внешнего объединения. Левое внешнее объединение включает все записи из первой (левой) таблицы, даже если нет совпадающих значений для записей из второй (правой) таблицы.
Операция RIGHT JOIN используется для создания правого внешнего объединения. Правое внешнее объединение включает все записи из второй (правой) таблицы, даже если нет совпадающих значений с записями из первой (левой) таблицы.

Предложение WHERE

Определяет, какие записи из таблиц, перечисленных в предложении FROM, следует включить в результат выполнения инструкции SELECT, UPDATE или DELETE. Ядро базы данных Microsoft Jet отбирает записи, соответствующие условиям, перечисленным в предложении WHERE. Предложение WHERE не является обязательным, однако, если оно присутствует, то должно следовать после предложения FROM. Например, можно отобрать всех сотрудников отдела продаж (WHERE Отдел = 'Продажи') или всех клиентов в возрасте от 18 до 30 лет (WHERE Возраст Between 18 And 30). Предложение WHERE может содержать до 40 выражений, связанных логическими операторами, такими как **And** и **Or**. При указании аргумента *условиеОтбора* литералы даты (символы дат) должны вводиться в американском формате, даже если используется неамериканская версия ядра базы данных Jet. Например, дата 10 мая 1996 года записывается в России как 10.05.96, а в США как 5/10/96.

Предложение GROUP BY

Объединяет записи с одинаковыми значениями в указанном списке полей в одну запись. Если инструкция SELECT содержит статистическую функцию SQL, например **Sum** или **Count**, то для каждой записи будет вычислено итоговое значение.

SELECT *списокПолей*
  FROM *таблица*
  WHERE *условиеОтбора*
  [GROUP BY *группируемыеПоля*]

Используйте предложение WHERE для исключения записей из группировки, а предложение HAVING для применения фильтра к записям после группировки. Если поле, включенное в предложение GROUP BY, не является полем типа Memo или объект OLE, оно может содержать ссылку на любое поле, перечисленное в предложении FROM, даже если это поле не включено в инструкцию SELECT, при условии, что инструкция SELECT содержит по крайней мере одну статистическую функцию SQL. Ядро базы данных Microsoft®

Jet не поддерживает группировку полей MEMO или объектов OLE. При использовании предложения GROUP BY все поля в списке полей инструкции SELECT должны быть либо включены в предложение GROUP BY, либо использоваться в качестве аргументов статистической функции SQL.

Предложение HAVING

Определяет, какие сгруппированные записи отображаются при использовании инструкции SELECT с предложением GROUP BY. После того как записи будут сгруппированы с помощью предложения GROUP BY, предложение HAVING отберет те из полученных записей, которые удовлетворяют условиям отбора, указанным в предложении HAVING.

SELECT *списокПолей*
  FROM *таблица*
  WHERE *условиеОтбора*
  [GROUP BY *группируемыеПоля*]
  [HAVING *условиеГруппировки*]

Предложение HAVING похоже на предложение WHERE, которое определяет, какие записи должны быть отобраны. После того как записи будут сгруппированы с помощью предложения GROUP BY, предложение HAVING указывает, какие из полученных записей должны быть отобраны. Предложение HAVING может содержать до 40 выражений, связанных логическими операторами, такими как **And** и **Or**

2. Инструкция SELECT...INTO

Создает запрос на создание таблицы.

SELECT *поле_1*[, *поле_2*[, ...]] INTO *новаяТаблица* [IN *внешняяБазаДанных*]
  FROM *источник*

Ниже перечислены аргументы инструкции SELECT...INTO:

Элемент	Описание
поле_1, *поле_2*	Имена полей, которые следует скопировать в новую таблицу.
новаяТаблица	Имя создаваемой таблицы. Это имя должно удовлетворять стандартным правилам именования. Если *новаяТаблица* совпадает с именем существующей таблицы, возникает перехватываемая ошибка.
внешняяБазаДанных	Путь к внешней базе данных. Более подробные сведения об этом аргументе можно найти в описании предложения IN.
источник	Имя существующей таблицы, из которой отбираются записи. Это может быть одна таблица, несколько таблиц или запрос.

Запрос на создание таблицы можно использовать для архивирования записей, создания резервных копий таблицы, копий для экспорта в другую базу данных, а также в качестве основы отчета, отображающего данные за конкретный период времени. Например, можно создать отчет «Ежемесячные продажи по областям», выполняя каждый месяц один и тот же запрос на создание таблицы.
Чтобы узнать, какие записи будут отобраны при выполнении запроса на создание таблицы, сначала просмотрите результаты инструкции SELECT, использующей те же условия отбора.

3. Инструкция INSERT INTO

Добавляет запись или записи в таблицу. Эта инструкция образует запрос на добавление записей.
Запрос на добавление нескольких записей:

INSERT INTO *назначение* [(*поле_1*[, *поле_2*[, ...]])] [IN *внешняяБазаДанных*]
  SELECT [*источник.*]*поле_1*[, *поле_2*[, ...]
  FROM *выражение*

Запрос на добавление одной записи:

INSERT INTO *назначение* [(*поле_1*[, *поле_2*[, ...]])]
  VALUES (*значение_1*[, *значение_2*[, ...])

Ниже перечислены аргументы инструкции INSERT INTO:

Элемент	Описание
назначение	Имя таблицы или запроса, в который добавляются записи.
поле_1, *поле_2*	Имена полей для добавления данных, если они следуют за аргументом *назначение*; имена полей, из которых берутся данные, если они следуют за аргументом *источник*.
внешняяБазаДанных	Путь к внешней базе данных.
источник	Имя таблицы или запроса, откуда копируются записи.
выражение	Имена таблицы или таблиц, откуда вставляются данные. Это выражение может быть именем отдельной таблицы или результатом операции INNER JOIN, LEFT JOIN или RIGHT JOIN, а также сохраненным запросом.
значение_1, *значение_2*	Значения, добавляемые в указанные поля новой записи. Каждое значение будет вставлено в поле, занимающее то же положение в списке: *значение_1* вставляется в *поле_1* в новой записи, *значение_2* - в *поле_2* и т. д. Каждое значение текстового поля следует заключать в кавычки (' '); для разделения значений используйте запятые.

Инструкцию INSERT INTO можно использовать для добавления одной записи в таблицу с помощью запроса на добавление одной записи, описанного выше. В этом случае инструкция содержит имя и значение каждого поля за-

писи. Нужно определить все поля записи, в которые будет помещено значение, и значения для этих полей. Если поля не определены, в недостающие столбцы будет вставлено значение по умолчанию или значение **Null.** Записи добавляются в конец таблицы.
Инструкцию INSERT INTO можно также использовать для добавления набора записей из другой таблицы или запроса с помощью предложения SELECT ... FROM, как показано выше в запросе на добавление нескольких записей. В этом случае предложение SELECT определяет поля, добавляемые в указанную таблицу *назначение*.
Вместо добавления существующих записей из другой таблицы, можно указать значения полей одной новой записи с помощью предложения VALUES. Если список полей опущен, предложение VALUES должно содержать значение для каждого поля таблицы; в противном случае инструкция INSERT не будет выполнена. Используйте дополнительную инструкцию INSERT INTO с предложением VALUES для каждой добавляемой новой записи.

4. Инструкция DELETE

Создает запрос на удаление записей, предназначенный для удаления записей из одной или нескольких таблиц, перечисленных в предложении FROM, которые удовлетворяют предложению WHERE.

DELETE [*таблица*.*]
  FROM *таблица*
  WHERE *условиеОтбора*

Ниже перечислены аргументы инструкции DELETE:

Элемент	**Описание**
таблица	Необязательное имя таблицы, из которой удаляются записи.
таблица	Имя таблицы, из которой удаляются записи.
условиеОтбора	Выражение, определяющее удаляемые записи.

Инструкция DELETE особенно удобна для удаления большого количества записей. При этом сохраняются структура таблицы и все остальные ее свойства, такие как атрибуты полей и индексы. Инструкцию DELETE можно использовать для удаления записей из таблиц, связанных отношением «один-ко-многим» с другими таблицами. Операции каскадного удаления приводят к удалению записей из таблиц, находящихся на стороне отношения «многие», когда в запросе удаляется соответствующая им запись на стороне «один». Запрос на удаление удаляет записи целиком, а не только содержимое указанных полей. Чтобы удалить данные в конкретном поле, следует создать запрос на обновление, который заменяет имеющиеся значения на значения **Null.**

5. Инструкция DROP

Удаляет существующую таблицу, процедуру или представление из базы данных или удаляет существующий индекс из таблицы.

DROP {TABLE *таблица* | INDEX *индекс* ON *таблица* | PROCEDURE *процедура* | VIEW *представление*}

Ниже перечислены аргументы инструкции DROP:

Элемент	Описание
таблица	Имя таблицы, которую следует удалить или из которой следует удалить индекс.
процедура	Имя удаляемой процедуры.
представление	Имя удаляемого представления.
индекс	Имя индекса, удаляемого из *таблицы*.

Прежде чем удалить таблицу или удалить из нее индекс, необходимо ее закрыть. Кроме того, для удаления индекса из таблицы можно использовать инструкцию ALTER TABLE.
Для создания таблицы можно использовать инструкцию CREATE TABLE, а для создания индекса - инструкцию CREATE INDEX или ALTER TABLE. Чтобы изменить таблицу, примените инструкцию ALTER TABLE.

6. Инструкция CREATE TABLE

Создает новую таблицу.

CREATE [TEMPORARY] TABLE *таблица* (*поле_1 тип* [(*размер*)] [NOT NULL] [WITH COMPRESSION | WITH COMP] [*индекс_1*] [, *поле_2 тип* [(*размер*)] [NOT NULL] [*индекс_2*] [, ...]] [, CONSTRAINT *составнойИндекс* [, ...]])

Ниже перечислены аргументы инструкции CREATE TABLE:

Элемент	Описание
таблица	Имя создаваемой таблицы.
поле_1, *поле_2*	Имена одного или нескольких полей, создаваемых в новой таблице. Таблица должна содержать хотя бы одно поле.
тип	Тип данных поля в новой таблице.
размер	Размер поля в знаках (только для текстовых и двоичных полей).
индекс_1, *индекс_2*	Предложение CONSTRAINT, предназначенное для создания простого индекса. Для получения более подробных сведений смотрите описание предложения CONSTRAINT.
составнойИндекс	Предложение CONSTRAINT, предназначенное для создания составного индекса. Для получения более подробных сведений смотрите описание предложения CONSTRAINT.

Инструкция CREATE TABLE используется для описания новой таблицы, ее полей и индексов. Если для поля добавлено ограничение NOT NULL, то при добавлении новых записей это поле должно содержать допустимые данные. Предложение CONSTRAINT устанавливает различные ограничения на поле и может быть использовано для определения ключа. Кроме того, для создания ключа или дополнительного индекса для существующей таблицы можно использовать инструкцию CREATE INDEX.
Допускается использование ограничения NOT NULL для одиночного поля, а также внутри именованного предложения CONSTRAINT, которое применяется к одиночному полю или к именованному предложению CONSTRAINT, предназначенному для создания составного индекса. Однако ограничение NOT NULL можно наложить на поле только один раз. При попытке применить это ограничение несколько раз возникает ошибка выполнения.
Создаваемая временная (TEMPORARY) таблица будет доступна только в том сеансе, где эта таблица была создана. По завершении данного сеанса она автоматически удаляется. Временные таблицы могут быть доступны для нескольких пользователей.
Использование атрибута WITH COMPRESSION допускается только для типов данных CHARACTER и MEMO (он же TEXT) и их синонимов. Атрибут WITH COMPRESSION был добавлен к столбцам CHARACTER вследствие перехода к формату представления знаков Юникод. Если столбец CHARACTER был определен с этим атрибутом, то при сохранении в нем данных осуществляется их автоматическое сжатие, а при извлечении данных - обратная операция.
Предусмотрены следующие типы данных: **Big Integer**, **Binary**, **Byte**, **Boolean**, **Char**, **Currency**, **Date**, **Decimal**, **Double**, **Float**, **GUID**, **Integer**, **Long**, **Long Binary** (**OLE Object**), **Memo**, **Numeric**, **Single**, **String**, **Text**, **Time**, **TimeStamp**, **VarBinary**, **Variant** (используемый по умолчанию), типы, определенные пользователем (создаваемые с помощью инструкции **Type**), и объектные типы данных, которые включают объектные типы данных приложения и типы объектов доступа к данным.

7. Инструкция UPDATE

Создает запрос на обновление, который изменяет значения полей указанной таблицы на основе заданного условия отбора.

UPDATE *таблица*
  SET *новоеЗначение*
  WHERE *условиеОтбора*;

Ниже перечислены аргументы инструкции UPDATE:

Элемент	Описание
таблица	Имя таблицы, данные в которой следует изменить.
новоеЗначение	Выражение, определяющее значение, которое должно быть вставлено в указанное поле обновленных записей.
условиеОтбора	Выражение, отбирающее записи, которые должны быть изменены. При выполнении этой инструкции будут изменены только записи, удовлетворяющие указанному условию.

Инструкцию UPDATE особенно удобно использовать для изменения сразу многих записей или в том случае, если записи, подлежащие изменению, находятся в разных таблицах.
Одновременно можно изменить значения нескольких полей. Следующая инструкция SQL увеличивает стоимость заказа на 10 процентов, а стоимость доставки на 3 процента:

```
UPDATE Заказы
SET СуммаЗаказа = СуммаЗаказа * 1.1,
СтоимостьДоставки = СтоимостьДоставки * 1.03
WHERE СтранаПолучателя = 'Литва';
```

8. Операция UNION

Создает запрос на объединение, который объединяет результаты нескольких независимых запросов или таблиц.

[TABLE] *запрос_1* UNION [ALL] [TABLE] *запрос_2* [UNION [ALL] [TABLE] *запрос_n* [...]]

Ниже перечислены аргументы операции UNION:

Элемент	Описание
запрос_1-n	Инструкция SELECT, имя сохраненного запроса или имя сохраненной таблицы, перед которым стоит зарезервированное слово TABLE.

В одной операции UNION можно объединить в любом наборе результаты нескольких запросов, таблиц и инструкций SELECT. В следующем примере объединяется существующая таблица «Новые счета» и инструкции SELECT:

```
TABLE [Новые счета] UNION ALL
SELECT *
FROM Клиенты
WHERE СуммаЗаказа > 1000;
```

По умолчанию повторяющиеся записи не возвращаются при использовании операции UNION, однако в нее можно добавить предикат ALL, чтобы гарантировать возврат всех записей. Кроме того, такие запросы выполняются быстрее. Все запросы, включенные в операцию UNION, должны отбирать одинаковое число полей; при этом типы данных и размеры полей не обязаны

совпадать. В каждом аргументе *запрос* допускается использование предложения GROUP BY или HAVING для группировки возвращаемых данных. В конец последнего аргумента *запрос* можно включить предложение ORDER BY, чтобы отсортировать возвращенные данные.

9. Статистические функции SQL

Статистические функции SQL используются для определения статистических данных на основе наборов числовых значений.

Функция **Avg**
Функция **Count**
Функции **First, Last**
Функции **Min, Max**
Функции **StDev, StDevP**
Функция **Sum**
Функции **Var, VarP**

5. ПРОГРАММА РАБОТЫ

1. Ознакомьтесь с теоретическими положениями.
2. Откройте БД.

№	ФИО студента	№ зачетки	Дисциплина	Вид ИА	Дата сдачи	Оценка
				Экзамен|зачет	Текущая дата	

3. Войдите в SQL-режим конструктора запросов и создайте таблицы в соответствии с указанной структурой, перенесите в них данные из ранее сформированной таблицы;

№	ФИО студента	№ зачетки
Ключ		

4. В SQL-режиме конструктора запросов создайте запросы на добавление/удаление новых записей в таблицы;
5. Из исходной таблицы удалите поле ФИО студента, сделайте поле «№зачетки» ключевым;
6. Выведите информацию о студентах, получивших за указанный период оценку «отлично» на экзамене.
7. Выведите список дисциплин (2-3), по которым студенты получают наихудшие оценки (в среднем);
8. Выведите ФИО студента, имеющего наилучшую успеваемость.

1. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое SQL-запросы?
2. Как создать SQL-запрос в ?
3. Какие SQL-инструкции реализованы в ?

Лабораторная работа № 9

ПРИМЕНЕНИЕ КОНСТРУКТОРА И МАСТЕРА ЗАПРОСОВ

1. ЦЕЛЬ РАБОТЫ

Изучение средств автоматизации формирования запросов в СУБД.

Отработка методов конструирования запросов, форм представления запросов и их реализация.

2. ОСНОВЫ ТЕОРИИ

Практическое использование базы данных основано на выборке из исходных таблиц определенного сорта информации, удовлетворяющей определенным критериям. Критерии могут определяться сочетанием ряда условий. Для решения задач, связанных с выборкой данных, предназначены мастера и конструкторы запросов.

Под запросом обычно понимается вопрос, сформулированный к БД. реализует метод формирования запроса по образцу. Запрос по образцу – это интерактивное средство для выбора данных из одной или нескольких таблиц. Формирование запроса осуществляется путем заполнения бланка запроса, который располагается в окне конструктора запросов. Такой метод формирования запроса прост для изучения и способствует эффективному использованию возможностей СУБД.

Для создания простейших запросов можно использовать мастер запросов.

Создание запросов

Неоспоримым преимуществом мастеров является возможность быстрого получения результатов при минимуме знаний о механизме его получения. Но, к сожалению, мастера запросов не позволяют указать условие выборки, критерий упорядочивания и более сложные, но и более интересные параметры выборки. Применение конструктора запросов в таких случаях позволяет создать сложные запросы в интерактивном режиме.

Окно конструктора запросов

Для вызова конструктора запросов перейдите в окне базы данных на вкладку «Запросы» и нажмите кнопку **Создать. В** окне диалога «Новый запрос» выберите опцию **Конструктор** и нажмите кнопку **ОК.** предложит вам выбрать таблицу или запрос, на основе которых будет осуществляться выборка. На этом этапе выбор таблицы или запроса не обязателен (вы можете добавить их и позже), но очевидно, что запрос должен выполняться, по крайней мере, на основе какой-либо таблицы или ранее созданного запроса. Выберите таблицу, нажмите кнопку **Добавить** и закройте окно диалога. На

экране появится окно конструктора запросов , а в основном меню — команда **Запрос.**

Для формирования запроса в окне конструктора запросов необходимо выбрать таблицы, из которых осуществляется выборка, и поля результата запроса, указать критерии для выборки, группировки и упорядочивания данных.

Меню **Запрос** содержит команды добавления таблиц в окно конструктора запросов и удаления их из нее, команды выбора типа запросов (Выборка, Создание таблицы, Перекрестный, Обновление, Добавление, Удаление), команду запуска запроса и управления запросом (Переключатель режимов, Групповые операции, Добавление Имени таблицы, Свойства).

В верхней части окна конструктора запросов находится схема данных запроса. Эта схема очень сильно напоминает схему данных базы данных. В отличие от нее, данная схема содержит список таблиц, включенных в запрос, и отображает связи между ними. В нижней части окна располагается бланк запроса. Каждая строка этого бланка выполняет определенную функцию:

- Поле. В этой строке помещаются те поля, которые вы используете для создания запроса, каждое в своей ячейке таблицы.

- Имя таблицы. Эта строка показывает вам, из какой таблицы (или запроса) выбрано данное поле.

- Сортировка. В этой строке вы указываете тип сортировки информации, возвращаемой в запросе, по возрастанию (от А до Я, от большего к меньшему, от более раннего к более позднему и т.д.) или по убыванию (от Я к А и т.д.).

- Вывод на экран. Если вы хотите, чтобы показывал информацию, найденную в поле, пометьте эту ячейку, чтобы установить флажок просмотра поля. Если же поле используется только для задания условия выбора данных, которые возвращает ваш запрос, оставьте его пустым.

- Условие отбора. В этой строке (и в строке, расположенной ниже ее) вы вводите ограничения поиска, задавая определенные условия, которые принято называть *критерием поиска.*

Создание простого запроса

Для создания простейших запросов достаточно данных одной таблицы. Рассмотрим создание такого запроса.

Вся необходимая информация находится в выбранной вами таблице. Поэтому для создания запроса выполните следующие действия:

1) В окне базы данных перейдите на вкладку «Запросы» и нажмите кнопку **Создать**. 2) Откроется окно диалога «Добавление таблицы», в котором выберите нужную таблицу и нажмите кнопку **Добавить**. Закройте окно диалога. 3) На экране открывается окно конструктора запросов, схема данных которого содержит всего одну таблицу, а бланк запроса пуст.

Добавление полей в бланк запроса

Для выбора полей, которые должны присутствовать в результирующей таблице, вам необходимо отобразить их в бланке запроса.

В существует два варианта выбора полей результирующей таблицы. Вы можете воспользоваться наиболее приемлемым с вашей точки зрения:

1. Для добавления в таблицу отдельных полей вы можете выбрать поле таблицы на схеме данных и дважды нажать кнопку мыши. Выбранное поле будет вставлено в следующий доступный столбец в строке Поле бланка запроса. В строке **Имя таблицы** сразу же появится имя таблицы, а позиция **Вывод на экран** будет помечена.

2. В широко используется механизм *перенести-и-оставить* (drag-and-drop). Для использования этого механизма при выборе полей перейдите в таблицу в схеме данных, из которой вам надо выбрать поля. Выделите поля, которые вы собираетесь отобразить в запросе, нажмите кнопку мыши и, не отпуская ее, перенесите выбранные поля в бланк запроса.

Для выделения группы полей используется мышь совместно с клавишами *Shift* и *Ctrl*. Для выбора отдельно расположенных полей вместо клавиши *Shift* используйте клавишу *Ctrl*.

В некоторых случаях необходимо выбрать все поля исходной таблицы. Для этого в существуют два способа, первый из которых состоит в выборе всех полей таблицы двойным нажатием мыши на строке заголовка и переносе выделенных полей в бланк запроса.

Вы можете использовать звездочку в списке полей таблицы для пометки всех полей в таблице. Для этого нажмите на звездочку в первой строке списка полей и, удерживая кнопку мыши в нажатом состоянии, перенесите ее в бланк запроса. Имя поля в бланке запроса будет содержать имя таблицы, за которым следует точка, а затем символ звездочки (например, клиенты.*). Это означает, что были выбраны все поля исходной таблицы. В отличие от первого способа, в режиме конструктора вы не увидите каждое поле в отдельном столбце, но после запуска запроса все они будут выбраны.

У метода переноса звездочки есть одно большое преимущество. Если вы добавите в таблицу новое поле после того, как вы создали и запустили запрос, включающий «*» ваш запрос автоматически включит в себя новое поле. При использовании первого способа запрос выбирает только те поля, которые были перенесены в бланк запроса.

Удаление полей из бланка запроса

Команда **Правка | Очистить бланк** удаляет все поля из бланка запроса.

Для удаления лишнего поля из запроса нажмите на область выбора столбца, а затем клавишу *Delete*. Поле исчезнет.

Прежде чем удалить поле, удостоверьтесь, что оно не используется в вашем запросе для определения условия выборки. В этом случае для запрета отображения поля в результирующей таблице снимите флажок **Вывод на экран.**

Изменение порядка полей

Порядок полей в бланке запроса определяет порядок появления их в результирующей таблице. Для того чтобы изменить расположение поля в этом списке, выполните следующие действия:

1) Установите указатель мыши на область выбора столбца, который располагается прямо над названием поля. Когда указатель изменит вид на стрелку, нажмите кнопку, чтобы выделить столбец. 2) Нажмите и удерживайте кнопку мыши в этом положении. На конце указателя появится прямоугольник. 3) Перемещайте столбец в требуемом направлении. Толстая вертикальная линия покажет его текущее положение. 4) Отпустите кнопку, когда толстая вертикальная линия окажется в требуемом месте. Поле будет перемещено в новое место.

Сортировка результатов выборки

Вы можете сортировать результаты выборки по одному или нескольким полям так же, как вы это делали при сортировке таблицы.

Порядок сортировки записей результирующей таблицы определяется порядком следования полей в бланке запроса и критерием упорядочивания отдельных полей. Чтобы отсортировать данные в отдельном поле, перейдите на строку **Сортировка** требуемого поля и из раскрывающегося списка выберите значение **По возрастанию** или **По убыванию.** Если вы решили отказаться от сортировки по полю, выберите значение **Отсутствует**. Для сортировки по нескольким полям последовательно переходите на сортируемые поля и установите для них критерий сортировки. В отличие от сортировки в режиме таблицы, вы увидите результат только после выполнения запроса.

Запуск запроса

Мы закончили создание простого запроса, и наступило время запустить его

на выполнение. Нажмите кнопку Запуск на панели инструментов или выполните команду **Запрос| Запуск**.

Сохранение запроса

Созданный запрос можно использовать в дальнейшем. Для этого вы Должны присвоить ему имя и сохранить его. Сохранение запроса осуществляется командой **Файл| Сохранить как/Экспорт,** которое откроет окно диалога ввода имени запроса. по умолчанию предложит имя запроса в поле ввода **Новое имя**, но лучше подобрать что-нибудь более значимое.

Использование критерия выборки записей для ограничения поиска

Предположим, что вы захотели при формировании запроса ограничиться только вашими знакомыми из Ленинграда. Для этого вы должны задать критерий выборки записей, который определяется следующим образом:

1. В списке полей таблицы найдите нужное поле и дважды нажмите на нем мышью, чтобы добавить в бланк запроса.

2. В строке **Условие отбора** для только что созданного поля введите его значение и снимите флажок вывода на экран для поля , так как это поле используется только для задания условия выборки записей.

3. Теперь запустите запрос, и результаты появятся в режиме таблицы,

4. Вы можете сохранить этот запрос под новым именем и использовать его в дальнейшем.

Изменение внешнего вида результирующей таблицы

Вы можете достаточно легко изменить внешний вид результирующей таблицы, используя те же средства, что и для обычных таблиц. Например, вы можете делать поля невидимыми, зафиксировать их, изменить шрифт, размеры столбцов и строк. Для выполнения этих функций перейдите в режим таблицы и выберите соответствующую команду из меню **Формат.**

3. ОБЪЕКТЫ И СРЕДСТВА ИССЛЕДОВАНИЯ

Предметная область в соответствии с вариантом задания. ПК типа IBM PC, дисплейный класс, операционная система Windows XP и выше, СУБД OO Base.

4. ПОДГОТОВКА К РАБОТЕ

Ознакомиться с теоретическими положениями лабораторной работы.

5. ПРОГРАММА РАБОТЫ

По предложенному преподавателем варианту разработать несколько вариантов часто используемых запросов по выборке данных из таблиц. Средствами конструктора запросов сформировать необходимые поля в бланке запросов. Предусмотреть сортировку выборок, а также различных критериев выборки записей из полей в соответствии с характером задания на лабораторные работы. Оформить отчет.

ВАРИАНТЫ ЗАДАНИЙ . Выбирается преподавателем из списка заданий (См. Приложение) и остается единым для всех лабораторных работ.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

6.1. Поясните метод формирования запросов к БД по образцу?

6.2. Опишите общую концепцию построения запросов в СУБД ?

6.3. Расскажите о технологии использования критериев выборки данных в вашей БД?

6.4. Опишите используемый вами способ формирования многотабличных запросов?

Лабораторная работа № 10

СОЗДАНИЕ В ЗАПРОСАХ ВЫЧИСЛЯЕМЫХ ПОЛЕЙ. ИСПОЛЬЗОВАНИЕ УСЛОВИЙ

1. ЦЕЛЬ РАБОТЫ

Изучение средств автоматизации формирования запросов в СУБД.

Отработка методов конструирования запросов, форм представления запросов и их реализация.

2. ОСНОВЫ ТЕОРИИ

Использование в запросах вычислимых полей

В результате выполнения запроса позволяет вам не только выбирать из таблицы содержащуюся в ней информацию, но также производить вычисления и отображать результат вычисления в результирующей таблице. Таким образом, вы можете получить данные, отсутствующие в исходной таблице.

При выполнении запроса вы можете вычислять значения по одному или нескольким полям исходной таблицы. Кроме того, вы можете использовать вычисляемые поля для объединения нескольких полей исходной таблицы в одно выходное поле. Например, вы можете в выходной таблице объединить фамилию и имя сотрудника.

Результаты вычислений, выводящиеся в поле, не запоминаются в исходной таблице. Вместо этого, вычисления выполняются каждый раз при запуске запроса, поэтому результаты всегда представляют текущее содержимое базы данных. Вы не можете редактировать результаты вычисления.

Для расчетов с использованием формул, определяемых пользователем, требуется создать новое вычисляемое поле прямо в бланке запроса. Вычисляемое поле создается с помощью выражения, которое вводится в пустую ячейку поля в бланке запроса или создается с помощью построителя выражений.

Выражение содержит формулы, которые связываются с помощью операторов. В качестве элементов формулы могут использоваться поля, константы и функции. Для изменения порядка вычислений и группировки данных в выражениях используются круглые скобки.

Обратите внимание, что наименование поля в выражении записывается в виде [Имя Таблицы] ! [Поле]. Имена таблицы и поля заключены в квадратные скобки, а между ними находится восклицательный знак. На этом примере видно неоспоримое преимущество использования построителя: вам не обязательно знать все принятые в соглашения. По выбранному вами значению построитель сам преобразует его в требуемый формат.

После завершения формирования выражения нажмите кнопку **ОК,** и выражение будет перенесено в строку **Поле** бланка запроса. Так как каждое поле результирующей таблицы должно иметь имя, автоматически задаст имя вычисляемого поля (например. Выражение 1), которое отделяется от выражения двоеточием.

Построение условий для выбора записей

На практике в большинстве случаев требуется получить не все записи исходной таблицы, а лишь ту часть, которая удовлетворяет определенным условиям. Простейший критерий для выбора записей предполагает точное совпадение значений поля. Определение такого критерия было рассмотрено ранее, поэтому перейдем к построению более сложных условий выборки записей.

Точное несовпадение значений одного из полей

Возможно, потребуется найти в таблице записи, значения которых не удовлетворяют определенному условию. Для установки таких условий используется оператор Not, который печатается перед сравниваемым значением. Выполните следующие действия:

1. В строке **Условие отбора** поля вставьте перед сравниваемым значением оператор Not или <>.

2. Проверьте установку флажка вывода на экран для контролируемого поля.

3. Для выполнения запроса нажмите кнопку Запуск. На экране появится результирующая таблица, которая содержит записи вне значения контролируемого поля.

Условие неточного совпадения

Требование точного задания чисел или последовательности символов в качестве критерия поиска является жестким ограничением. Время от времени возникает неприятная ситуация, при которой вы не помните точного написания условия поиска. Кроме того, вы можете просто не знать, какой регистр (верхний, нижний или их сочетание) был использован при вводе каждой записи. При работе с большими таблицами такая ситуация превращается в серьезную проблему.

В подобных случаях вы можете осуществить выбор записей по условию неточного совпадения значений. Данное условие позволяет найти требуемые записи, зная лишь приблизительное написание значения. В для этих целей используется оператор Like, который осуществляет сравнение значения поля с заданным образцом. Образец задается сразу же за оператором и может содержать точное значение (например, Like "Иванов") или использовать символы шаблонов для поиска диапазона значений (например, Like "Ив*"). Приведенная ниже таблица содержит символы шаблонов, которые используются с оператором Like, и количество цифр или символов, которым они соответствуют.

Символы в образце	**Соответствие в выражении**
?	Любой один знак
*	Нуль или более знаков
#	Любая одна цифра
[список_зиаков]	Любой один знак в **списке_знаков**
[!список_знаков]	Любой один знак, не входящий в **список_знаков**

Группа из одного или более символов, заключенная в квадратные скобки, используется для установления совпадения с одним символом выражения и может содержать любые символы, за исключением перечисленных ниже особых символов.

Особыми символами являются открывающая квадратная скобка ([), вопросительный знак (?), знак числа (#) и знак звездочки (*). Для сравнения с ними необходимо заключить их в квадратные скобки. Закрывающую квадратную скобку нельзя применять в группе для установки соответствия с ней самой, но она может стоять вне группы как отдельный символ.

Помимо простого списка символов, заключенного в квадратные скобки, **список_знаков** позволяет задать диапазон символов. Вы можете указать знак (-) для разделения верхней границы диапазона от нижней. Например, использование шаблона [Г-Л] в образце позволяет выбрать все записи, которые в указанной позиции поля содержат одну из заглавных букв в диапазоне от Г до Л. Вы можете использовать даже несколько диапазонов внутри одной пары квадратных скобок без специальных разделителей. Например, [г-лГ-Л] позволяет выбрать как заглавные, так и прописные буквы.

Восклицательный знак в начале **списка_знаков** означает, что совпадение наступит, если в выражении будет найден любой символ, отсутствующий в **списке знаков.** Восклицательный знак, использованный вне квадратных скобок, соответствует самому себе.

Знак (-) для установления соответствия с самим собой можно использовать в начале (после восклицательного знака, если он есть) или в конце **списка_зваков.**

Рассмотрим простой пример использования оператора **Like.** Предположим, что в таблице Клиенты требуется найти запись о представителе фирмы, название которой начинается на Ric. Однако вы не уверены в правильности написания фирмы. Попытаемся найти требуемую запись. Для задания условия выберите поле с названием и в строке **Условие** отбора поля напечатайте Like 'Ric*' .

Рассмотрим более сложный пример использования оператора **Like.** Допустим, вам надо выбрать всех клиентов, наименования которых начинаются на А или С. Для этого в строке **Условие отбора** поля Название введите следующее условие: Like ' [АС] * '.

Выбор записей по диапазону значений

Часто необходимо выбрать из таблицы записи , данные которых попадают в диапазон значений. При этом границы диапазона заданы точно.

Для задания диапазона значений в окне конструктора запросов используются операторы > (больше), >= (не менее), < (меньше), <= (не более) **и Between,** которые вы можете использовать с текстовыми и цифровыми полями, а также полями дат. Для задания диапазона значений этого выполните следующие действия:

1) Откройте на экране окно конструктора запросов для нужной таблицы. 2) Выделите поля, которые хотите отобразить в запросе и перенесите их в бланк запроса. 3) Для задания условия отбора выберите поле и в строке **Условие отбора** поля напечатайте условие(например, >100000). 4) Нажмите кнопку **Запуск,** и на экране появится результирующая таблица, содержащая записи о значениях полей свыше 100 000.

В рассмотренном примере мы задавали только одну нижнюю границу диапазона. Очевидно, что можно использовать более сложные конструкции операторов

В заключение рассмотрим пример задания диапазона строковых данных. В этом случае кроме операторов сравнения вы можете использовать и оператор Like. Например, для получения списка клиентов, наименования которых начинаются с С по G, в строку Условие **отбора** поля Название введите условие отбора : Like '[C-G]* '

Объединение критериев нескольких полей

В предыдущих примерах мы вводили условие запроса только для одного из полей таблицы. Однако довольно часто возникают ситуации, когда вам необходимо использовать более сложный критерий выборки, в котором задаются условия для нескольких полей таблицы или же несколько условий для одного поля. Если запись выбирается только в случае выполнения всех условий, то условие такого выбора называется логическим И, а запрос — И-запросом. Если же запись выбирается при выполнении хотя бы одного из всех условий, то условие такого поиска называется логическим ИЛИ, а запрос — ИЛИ- запросом.

Для задания И- выражения вы должны просто задать условие в строке **Условие отбора** для каждого из полей, образующих критерий. В качестве примера предположим что из всех записей о заказах за определенный интервал времени требуется выбрать те, код доставки которых равен 1. Для решения этой задачи откройте запрос, в котором выбирались заказы в диапазоне дат, и выполните следующие действия:

1) Добавьте в бланк запроса поле Доставка. 2) Перейдите на строку **Условие отбора** поля и напечатайте 1 . 3) Выполните запрос, и вы увидите записи, содержащие сведения о заказах за указанный период дат, в которых использовалась доставка с кодом 1.

При задании ИЛИ- выражения каждое из условий выбора, образующих критерий должно располагаться на отдельной строке бланка запроса. Например, для выбора списка клиентов из США и Швеции нужно просто расположить первое условие в строке Условие **отбора,** а второе - в строке **Или** .

При формировании ИЛИ- выражения вы можете расположить условия выбора для различных полей в разных строках бланка запроса.

При вводе условия вы можете использовать операторы **Or** и **And,** которые позволяют вам формировать в одной строке сложное условие выборки. Например, при поиске записей о клиентах из США и Швеции, вы можете поместить ИЛИ- выражение в одной строке. Результирующая таблица будет содержать те же записи, что и запрос. При вводе условия вы можете формировать любое допустимое в логическое условие, которое может содержать функции, операторы сравнения, Or, And, Not и скобки для изменения порядка выполнения выражения.

Многотабличные запросы

При выборе данных из таблиц наиболее часто используются многотабличные запросы, поскольку информация в реляционных базах данных содержится не в одной отдельной таблице, а в совокупности связанных таблиц. Связь между таблицами осуществляется на основании совпадающих полей.

Для формирования многотабличного запроса нужно добавить в окно конструктора запросов все таблицы, участвующие в выборке, и определить условия их объединения. Для добавления таблицы выберите команду **Запрос | Добавить таблицу** или нажмите кнопку **Добавить таблицу** на панели инструментов. В открывшемся окне диалога «Добавление таблицы» выберите нужную таблицу. Образ таблицы появится в схеме данных запроса. Если в базе данных установлены отношения между таблицами, участвующими в запросе, то эта связь будет отображаться в виде линии, соединяющей таблицы. В этом случае вам не придется устанавливать связь между таблицами в конструкторе запросов. Если же между таблицами не существует связи, то вы можете установить требуемую связь, используя механизм перенести -и-оставить. Для этого выберите поле в одной из таблиц, нажмите кнопку мыши и перенесите выбранное поле на связываемое поле в другой таблице.

В отличие от определения постоянных отношений между таблицами, при задании условия объединения таблиц вы можете использовать любые поля таблиц.

Объединение двух таблиц

Вначале рассмотрим простое объединение двух таблиц на следующем примере. Отпускаемый клиентам товар может доставляться разными средствами. Для определения типа доставки товаров необходимо использовать информацию из двух связанных таблиц Т1 и ТО. Пусть при создании базы данных между этими таблицами уже определены постоянные отношения. Для решения этой задачи выполните следующие действия:

1) Добавьте таблицу Т1 в новое окно конструктора запросов. 2) Нажмите кнопку **Добавить таблицу** на панели инструментов и добавьте в запрос еще одну таблицу — ТО. Поскольку между этими таблицами установлено постоянное отношение, то после добавления таблицы в окне конструктора запросов автоматически отобразится связь между ними. 3) Перенесите в бланк запроса из таблицы ТО код товара, а из таблицы Т1 — код заказа и тип доставки . 4) Выполните запрос, и на экране появится результирующая таблица, содержащая требуемую информацию .

Условие отбора в многотабличных запросах

Теперь обратимся к более сложному запросу, в котором из всех записей в таблицах необходимо выбрать только те записи, которые отвечают определенному условию отбора.

Для задания условия отбора перейдите на строку **Условие отбора** поля, по которому производится выборка, снимите флажок **Вывод на экран** для этого поля, так как не имеет смысла выводить столбец, каждая строка которого будет содержать одно и то же значение. Перейдите на **Условие отбора** поля другой записи, а затем снимите для нее флажок **Вывод на экран.** Далее добавьте в бланк запроса дополнительные необходимые поля и выполните запрос. Вы увидите на экране результирующую таблицу, содержащую информацию **обо** всех интересующих вас значениях.

Итоговые запросы

При необходимости можно объединить в запросе любое количество требуемых таблиц. Однако довольно часто имеющихся в таблицах данных оказывается недостаточно. Например, вам нужно найти сумму, максимальную величину в поле или количество записей, содержащих определенную величину. В этом случае вам необходимо выполнить итоговые вычисления. Для этих целей используются математические возможности , который отлично приспособлен для выполнения таких вычислений.

Для определения суммы значений полей или нахождения среднего следует создать итоговый запрос. Для его создания, как и для обычного запроса, откройте новое окно конструктора запросов. Далее выберите используемые в запросе таблицы, а затем перенесите в бланк запроса нужные поля.

Запросы, выполняющие вычисления в группах записей, называются *итоговыми запросами.* В итоговом запросе выполняется не только Суммирование, но и другие виды вычислений. Например, вы можете найти среднее, минимальное и максимальное значения поля.

Для создания итогового запроса выберите **Вид| Групповые операции** или нажмите кнопку **Групповые операции** на панели инструментов. В бланке запроса появится новая строка с наименованием **Групповая операция**. В этой строке вы должны указать тип выполняемого вычисления (например, суммирование – **Sum)**.

Табл. 4.1 содержит перечень всех допустимых видов итоговых операций, которые можно выбрать из раскрывающегося списка в строке Групповая операция. Две из них — StDev и Var -являются известными операциями статистки. Операция Count требует некоторых дополнительных разъяснений. Она действует следующим образом: просматриваются все записи для указанного поля, и вычисляется количество записей, которые содержат в этом поле какое-либо значение. Другими словами, данная операция пропускает пустые значения.

Таблица. 4.1

Типы операций, доступные в строке **Групповая операция** бланка запроса

Значение	Выполняемая операция	Значение	**Выполняемая операция**
Sum	Сложение	**Min**	Минимальное значение
Avg	Среднее значение	**Count**	Количество записей, содержащих значения
Var	Дисперсия	**First, Last**	Значение в первой и последней записи
Max	Максимальное значение	**StDev**	Стандартное отклонение

Для удаления строки **Групповая операция** достаточно нажать еще раз на кнопку **Групповая операция**.

Группировка полей запроса

Группировка позволяет получить вычисляемую информацию о подгруппах таблицы. Например, сгруппировав данные в таблице Заказано по полю КодТовара, можно получить сведения об итоговом количестве заказов каждого вида товара.

Как обычно, создание запроса начните с добавления таблицы в окно конструктора запросов. Далее выполните следующие действия:

1) После размещения полей в бланке запроса выберите команду **Вид | Групповые операции** для добавления в бланк запроса **строки Групповая операция.** 2) В строке **Групповая операция** группируемого столбца установите значение **Группировка,** а в другом столбце можно подсчитать количество записей, установив значение **Count.**

При формировании групповых итоговых запросов довольно часто данные упорядочиваются по итоговым полям, Например, в запросе для поля, в котором определяется количество записей, установлен признак сортировки по убыванию. Это означает, что в результирующей таблице первыми будут расположены наиболее часто встречающиеся записи.

Поле, используемое для группировки, не обязательно должно находиться в гой же таблице, что и итоговое поле. Оно может находиться в другой таблице.

Группировка по нескольким полям

Вы можете группировать данные не только по одному полю, но и по нескольким полям одновременно, а также создавать группы внутри групп. Для этого вы можете сгруппировать записи сначала по одному полю, затем — по другому. Порядок полей группировки, размещенных в бланке запроса, определяет порядок группировки выбираемых данных.

Включение в запрос выражений

предоставляет вам возможность выполнять итоговые операции над вычисляемыми полями выборки. Например, таблица Заказы содержит данные о клиентах, а таблица Заказано — о количестве товаров, проданных в каждой партии и цене товара. На основании этой информации вы можете вычислить итоговую стоимость заказа каждым покупателем по каждому товару.

Чтобы включить в итоговый запрос выражение, добавьте в бланк запроса вычисляемое поле и укажите тип итоговых имений, осуществляемых над этим полем (т.е. выражение для вычисления).

Выражения для вычисления могут содержать формулы, связанные арифметическими операторами. В качестве элементов формулы могут использоваться поля, константы и функции. Для изменения порядка вычислений в выражениях используются круглые скобки.

Изменение наименований итоговых полей

По умолчанию поля результирующей таблицы имеют те же наименования, что и поля исходной таблицы, а итоговым полям присваиваются наименования в соответствии с принятым в соглашением. Начальная часть имени обычно содержит имя итоговой операции, за которым следует имя поля, над которым выполняется итоговая операция. Во многих случаях предпочтительнее давать таким полям более осмысленные наименования. позволяет вам по своему усмотрению изменить наименования полей результирующей таблицы.

Для изменения наименования поля в строке **Поле** перед именем поля или выражения напечатайте новое имя поля и отделите его двоеточием.

Изменение наименований полей в бланке запроса действует только на поля в результирующей таблице. Имена полей в исходной таблице остаются без изменения.

Использование более сложных условий отбора записей в итоговых запросах

В итоговых запросах существует два типа критериев отбора записей. Первый тип аналогичен критерию, используемому в обычных запросах. Он исключает записи, не удовлетворяющие заданному критерию, перед выполнением итоговых вычислений. Второй тип является специфичным для итоговых запросов. Данный критерий применяется к результату итоговых вычис-

лений. Критерии вводятся в строке **Условие отбора.**

3. ОБЪЕКТЫ И СРЕДСТВА ИССЛЕДОВАНИЯ

Предметная область в соответствии с вариантом задания. ПК типа IBM PC, дисплейный класс, операционная система Windows XP и выше, СУБД OO Base.

4. ПОДГОТОВКА К РАБОТЕ

Ознакомиться с теоретическими положениями лабораторной работы.

5. ПРОГРАММА РАБОТЫ

По предложенному преподавателем варианту разработать несколько вариантов часто используемых запросов по выборке данных из таблиц. Использовать в запросе одно или несколько вычислимых полей. Разработать один или несколько вариантов многотабличных запросов с использованием группировки полей. Разработать необходимые формы для просмотра и редактирования данных с использованием конструктора форм. Оформить отчет.

ВАРИАНТЫ ЗАДАНИЙ . Выбирается преподавателем из списка заданий (См. Приложение) и остается единым для всех лабораторных работ.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

6.1. Поясните метод формирования запросов к БД по образцу?

6.2. Опишите общую концепцию построения запросов в СУБД ?

6.3. Расскажите о технологии использования критериев выборки данных в вашей БД?

6.4. Опишите используемый вами способ формирования многотабличных запросов?

Лабораторная работа № 11

УПРАВЛЕНИЕ ДОСТУПОМ К ОБЪЕКТАМ БАЗЫ ДАННЫХ

1. ЦЕЛЬ РАБОТЫ

Получить навыки интеграции различных баз данных с приложениями.

ОСНОВЫ ТЕОРИИ

Компоненты ADO

Компоненты для работы с Microsoft® ActiveX® Data Objects (далее ADO) - это технология стандартного обращения к реляционным данным от Microsoft. Эта технология аналогична BDE по назначению и довольно близка по возможностям.

Обзор компонент

Для работы с ADO на вкладке компонентов ADO есть шесть компонентов: **TADOConnection, TADOCommand, TADODataSet, TADOTable, TADOQuery, TADOStoredProc.**

Рис. 6.1. Палитра компонент ADO

TADOConnection аналогичен компоненту BDE **TDatabase** и используется для указания базы данных и работы транзакциями.

TADOTable – таблица доступная через ADO.

TADOQuery – запрос к базе данных. Это может быть как запрос, в результате которого возвращаются данные и базы (например, SELECT), так и запрос не возвращающий данных (например, INSERT).

TADOStoredProc – вызов хранимой процедуры. В отличие от BDE и InterBase хранимые процедуры в ADO могут возвращать набор данных, поэтому компонент данного типа является потомком от **TDataSet** и может выступать источником данных в компонентах типа **TDataSource**.

TADOCommand и **TADODataSet** являются наиболее общими компонентами для работы с ADO, но и наиболее сложными в работе. Оба компонента позволяют выполнять команды на языке провайдера данных (так в ADO называется драйвер базы данных).

Разница между ними в том, что команда, исполняемая через **TADODataSet,** должна возвращать набор данных и этот компонент позволяет работать с ними средствами Delphi (например, привязать компонент типа

TDataSource). А компонент **TADOCommand** позволяет исполнять команды не возвращающие набор данных, но не имеет штатных средств Delphi для последующего использования возвращенного набора данных.

Очевидно, что все компоненты должны связываться с базой данных. Делается это двумя способами либо через компонент **TADOConnection** либо прямым указанием базы данных в остальных компонентах. К **TADOConnection** остальные компоненты привязываются с помощью свойства **Connection**, к базе данных напрямую через свойство **ConnectionString**.

База данных может быть указана двумя способами через файл линка к данным (файл в формате Microsoft Data Link, расширение UDL), либо прямым заданием параметров соединения.

Значение свойства всех **ConnectionString** этих компонент могут быть введены напрямую в текстовой форме, но куда проще вызвать редактор свойства нажав на кнопку «...» в конце поля ввода. Окно этого свойства выглядит так:

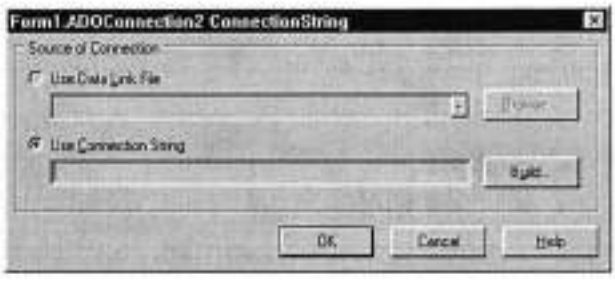


Рис. 6.2.

При выборе «Use data link file» и нажатии на кнопку «Browse...» появляется стандартный диалог выбора файла. Этот файл можно создать в любом окне explorer-а (в этом окне открытия файла, в самом explorer, на desktop и т.д.) вызвав контекстное меню и выбрав пункт «New/Microsoft Data Link». Потом вызовите локальное меню для созданного файла и выберите в нем пункт «Open». После этого появится property sheet описанный чуть ниже. Эти же вкладки содержит и property sheet, вызываемый через пункт «Property» локального меню UDL файла, но в нем еще есть вкладки относящиеся к самому файлу.

Использование файлов Microsoft Data Link упрощает поддержку приложений, так как возможно использовать средства Windows для настройки приложения.

При выборе в редакторе свойства «Use connection string» и нажатии на кнопку «Build...» появляется такой же property sheet, как и при выборе «Open» для Microsoft Data Link файла.

В этом окне выбирается тип базы данных, местоположение базы и параметры соединения.

На первой странице выбирается тип базы данных или Provider, в терминах ADO.

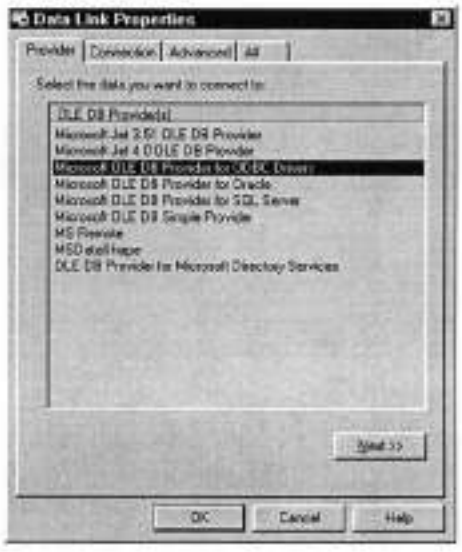


Рис. 6.3.

Базы доступны как через «Microsoft Jet OLE DB Provider», так и через «Microsoft OLE DB Provider for ODBC».

Следующая страница зависит от выбранного типа базы, однако для всех типов есть кнопка «Test connection» позволяющая проверить правильность и полноту параметров.

Для «Microsoft Jet OLE DB Provider» она выглядит так:

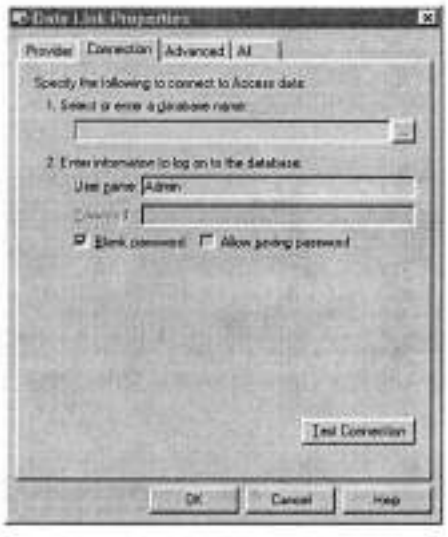


Рис. 6.4.

Checkbox «Blank password» подавляет диалог ввода идентификатора и пароля пользователя при установлении соединения, если поле пароля пустое.

Checkbox «Allow saving password» сохраняет пароль в строке параметров соединения. Если он не отмечен, то введенный пароль будет использоваться только при выполнении тестового соединения.

Для «Microsoft OLE DB Provider for ODBC» эта страница выглядит так:

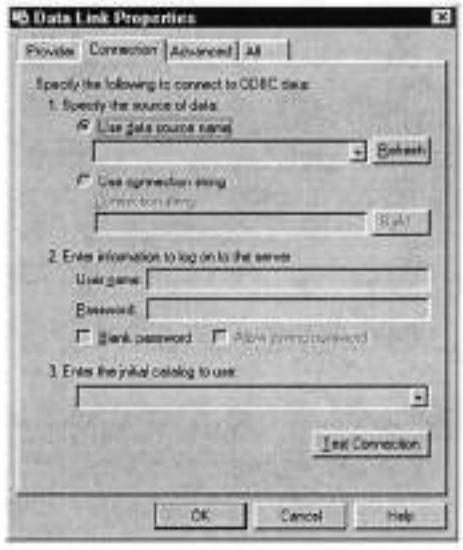


Рис. 6.5.

Радиокнопка «Use data source name» позволяет ввести предустановленный алиас ODBC, а через «Use connection string» вводится как алиасы так и тип ODBC драйвера и параметры соединения.

Параметры идентификации пользователя аналогичны выше описанным.

На странице «Advanced» расположены дополнительные параметры, с помощью которых устанавливается уровень доступа к файлу базы данных, таймаут сетевого соединения (то есть время через которое связь будет считаться потерянной, если сервер не отвечает) и уровень глубины проверки секретности соединения.

Рис. 6.6

На странице «All» можно отредактировать все параметры с предыдущих страниц и параметры зависящие от провайдера, но не вошедшие на страницу «Connection». Редактирование осуществляется в виде параметр –

значение, причем в текстовой форме, никаких диалогов нет. Помощи то же нет, эти параметры описаны только в документации на провайдер. Ее можно найти в MSDN Data Services/Microsoft Data Components (MDAC) SDK/Microsoft ActiveX Data Objects (ADO)/Microsoft ADO Programmer's Reference/Using Providers with ADO.

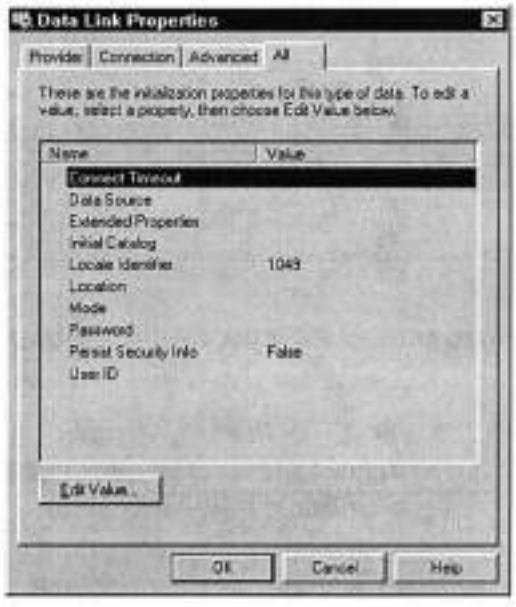


Рис. 6.7.

В компоненте **TADOConnection** есть свойства **Provider**, **DefaultDatabase** и **Mode** которые являются альтернативным методом задания частей строки параметров соединения – провайдера, базы данных (например, пути до базы) и режима совместного использования файлов базы данных. Эти значение этих свойств автоматически включаются в строку соединения, если были заданы до активизации компонента и автоматически выставляются после соединения.

Простейшее приложение

Создадим простейшее приложение, состоящее из одной таблицы. Создаем форму состоящую из трех компонент

TADOTable с палитры компонент ADO,

TDataSource с палитры компонент Data ,

TDBGrid с палитры компонент Data Controls.

Рис 6.8

Связываем компоненты, устанавливая свойство **TDBGrid DataSource** на компонент **TDataSource**, свойство **DataSet** компонента **TDataSource** на компонент **TADOTable**.

Теперь нам необходимо указать базу данных. Делается это в свойстве **ConnectionString** компонента **TADOTable**. При нажатии на кнопку «…» появится редактор параметров соединения. Отметим радокнопку «Use data link file», нажмите на кнопку «Browse…» и выберите в появившемся окне после файл линка к базе данных «\Program Files\Common Files\System\ole db\Data Links\DBDEMOS.UDL». Этот линк указывает на базу в формате , входящую в поставку Delphi.

После этого в свойстве **TableName** компонента **TADOTable** выберем таблицу customer.

Активизируем компонента **TADOTable**, установив свойство **Active** в **True**.

Приложение можно запускать. Этот пример можно найти в директории Simple.

Обзор ADO

ADO основано на технологии COM. Все объекты и интерфейсы ADO являются интерфейсами и объектами COM.

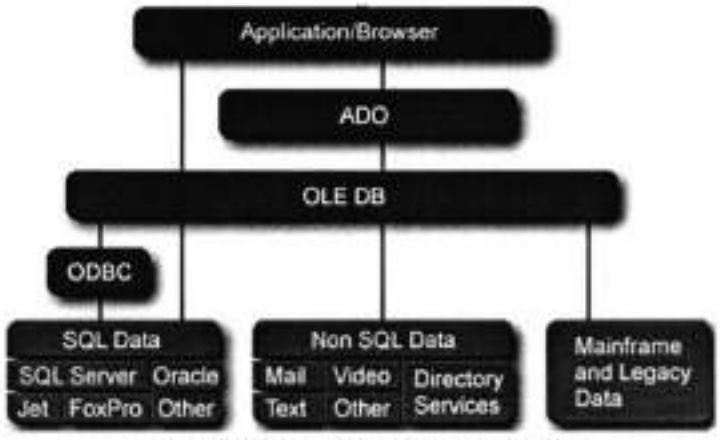


Рис 6.9. Архитектура ADO

Интерфейс **Connection**

Объекты этого типа выполняют следующие функции:

Связь с сервером.

Управление транзакциями.

Получение информации о произошедших ошибках (свойство **Errors**).

Получение информации о схеме данных (таблицы, поля и так далее).

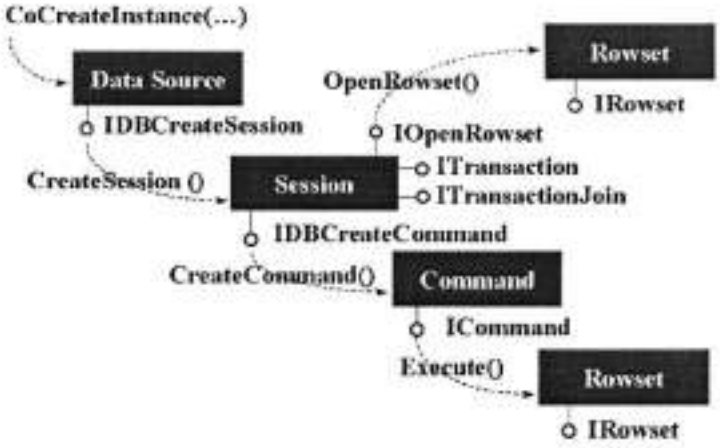


Рис 6.10. Схема взаимодействия в ADO основных COM интерфейсов

Интерфейсы **Recordset** и **Field**

Интерфейс **Recordset** (на нижнем уровне ADO это IRowset) является аналогом **TDataSet** в Delphi.

Поддерживает текущее положение и перемещение курсора, закладки (bookmarks), чтение, изменение и удаление записей и так далее. Значение полей и их типы доступны с помощью свойства **Fields**.

Интерфейс **Field** позволяет получать значение поля, его тип длину и так далее.

Интерфейсы **Command** и **Parameter**

Эти два типа позволяют работать с командами источника данных. Синтаксис команд для каждого из источников свой.

Интерфейс **Property**

Все объекты, кроме **Parameter**, имеют свойство **Properties**, которое позволяет получать и устанавливать параметры специфические для провайдера данных.

Библиотека довольно запутанная, многие функции дублированы в разных объектах. Например, **Recordset** можно создавать напрямую, методом **Open**, (причем предварительно создавать **Connection** не обязательно), можно получить как результат выполнения метода **Command.Execute**, либо после **Connection.Execute** задав команду без параметров.

Интерфейс **Command** инкапсулирован во все компоненты за исключением **TADOConnection**. Это сделано потому, что в ADO нет возможности получить данные не выполнив команду.

Интерфейс **Recordset** инкапсулирован в компоненты производные от **TCustomADODataSet.** Это компоненты **TADODataSet, TADOTable,**

TADOQuery, TADOStoredProc. Получать данные из них возможно штатными средствами Delphi.

Возможно получение данных и при выполнении компонента **TADOCommand**. Метод этого компонента **Execute** возвращает тип **_Recordset**. После чего его можно, например, связать с компонентом **TADODataSet** следующим образом

```
ADODataSet1.RecordSet := ADOCommand1.Execute;
```

Компоненты **TADOTable, TADOQuery** и **TADOStoredProc** являются частными случаями команды, соответственно для таблицы, SQL запроса и хранимой процедуры.

Тип **Connection** инкапсулируется в компонент **TADOConnection**.

Когда вы выполняете команду предварительно не открывая соединение, оно все равно создается. Получить к нему доступ возможно через свойство **Recordset**. Привязать компонент **TADOConnection** к уже открытому соединению возможно через свойство **ConnectionObject**.

Информацию о структуре базы данных можно получить с помощью метода **OpenSchema** компонента **TADOConnection**. Эта информация представлена как набор таблиц, как стандартизованных, так и специфических для провайдера. Таким способом можно узнать список таблиц, запросов, хранимых процедур и многое другое. Однако изменять структуру базы с помощью возвращаемых наборов данных невозможно.

Пример использования TADOConnection

В этом примере рассматривается работа с компонентом **TADOConnection**, SQL запросами с параметрами и трансакциями.

Создадим приложение из следующих компонентов

Connect типа **TADOConnection**

MasterSQL и **DetailSQL** типа **TADODataSet**

MasterDS и **DetailDS** типа **TDataSource**

MasterGrid и **DetailGrid** типа **TDBGrid**

Рис 6.11. Master-detail форма на этапе дизайна

Связываем **MasterGrid, MasterDS, MasterSQL** и **DetailGrid, DetailDS, DetailSQL** аналогично предыдущему примеру, за исключением того, что вместо типа **TADOTable** используется тип **TADODataSet.**

Привязываем **Connect** к базе данных. Для этого в редакторе свойства **ConnectionString** выбираем ту же базу данных, что и в предыдущем примере.

Для ввода SQL запросов необходимо отредактировать свойство **CommandText** компонентах **MasterSQL** и **DetailSQL**. После нажатия на кнопку «...» появится редактор компонент, который выглядит следующим образом

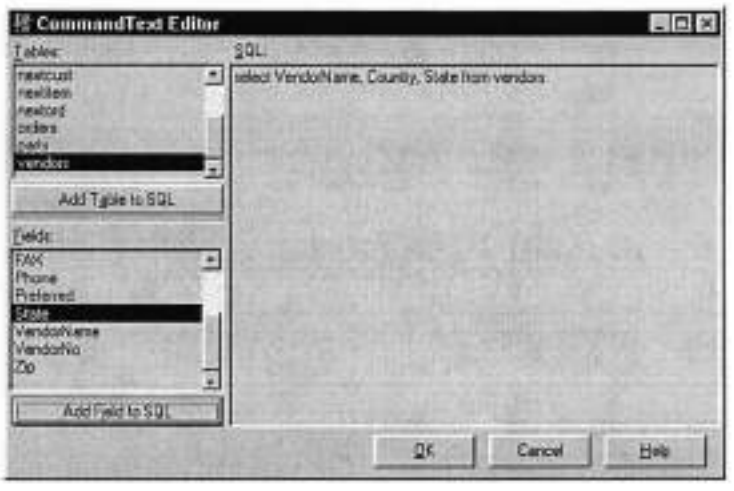


Рис. 6.12

Кнопка «Add Table to SQL» добавляет в текст SQL запроса таблицу, выбранную в списке «Tables», а «Add Field to SQL» поле таблицы, выбранное в списке «Fields».

Запрос для **MasterSQL**

```
select VendorNo, VendorName, Country, City, State, Preferred
from vendors
```

Запрос в **DetailSQL** должен выбирать только те детали, поставщик которых является текущим в **MasterSQL**. Для этого установим свойство **DataSource** компонента **DetailSQL** в значение **MasterDS.**

Запрос для **DetailSQL** следующий:

```
select PartNo, OnOrder, OnHand, ListPrice, Description, Cost
from parts
where VendorNo = :VendorNo
```

:VendorNo в части **where** – параметр запроса. Параметры при установленном **DataSource** берутся из него.

Активизируем **MasterSQL** и **DetailSQL** аналогично предыдущему примеру.

Приложение можно запускать. Этот пример можно найти в директории MasterDetail.

Пример использования параметров запроса

Теперь ограничим выборку поставщиков по значению поля **State.** Для этого добавим к форме следующие компоненты **StateEdit** типа **TEdit** с вкладки Standard, Query**Button** типа **TButton** с вкладки Standard

Изменим запрос в **MasterSQL** на

```
select VendorNo, VendorName, Country, City, State, Preferred
from vendors
where State = :StateID
```

:StateID – параметр, вместо которого при выполнении подставляется значение.

Добавим так же обработчик события **OnClick** в **QueryButton** следующего содержания

```
procedure TForm1.QueryButtonClick(Sender: TObject);
begin
  MasterSQL.Active := False;
  DetailSQL.Active := False;
  MasterSQL.Parameters.ParamByName('StateID').Value := StateEdit.Text;
  MasterSQL.Active := True;
  DetailSQL.Active := True;
end;
```

Программа готова. Этот пример можно найти в директории Param.

Синхронизация данных клиента и сервера

В ADO используются три метода синхронизации данных на клиенте и сервере.

Первый – с помощью метода **Resync**, который повторно считывает записи набора. Этот метод используется при выполнении метода **Refresh** Delphi.

Второй – повторный запрос методом **Requery**, который заново выполняет запрос на сервере. Выполнение этого метода то же самое, что и выполнение подряд закрытия и открытия набора данных.

Третий- уведомление сервером клиента в случае изменения данных.

Этих методы доступны во всех компонентах имеющих набор данных. Однако эти функции доступны не для всех баз данных.

Работа с транзакциями

В компонентах ADO работа с транзакциями осуществляется через компонент **TADOConnection**.

Тип транзакции устанавливается в свойстве **IsolationLevel** одной из следующих констант:

IlUnspecified	Сервер будет использовать лучший, по его мнению, тип изоляции.
IlChaos	Транзакции с более высоким уровнем изоляции не могут изменять данные измененные, но не подтвержденные в текущей транзакции.
IlReadUncommitted	Чтение данных измененных в не подтвержденных транзакций. То есть изменения видны сразу после того как другая транзакция передала их на сервер.
IlBrowse	То же самое что и **IlReadUncommitted**
IlReadCommitted	Чтение данных измененных подтвержденными транзакциями. То есть изменение данных будет видимо после выполнения Commit в другой транзакции.
IlCursorStability	То же самое что и **IlCursorStability**.
IlRepeatableRead	Изменения, сделанные другими транзакциями не видимы, но при выполнении перезапроса они транзакция может получать

	новый набор данных.
IIIsolated	Транзакция не видит изменений данных произведенных другими транзакциями.
IISerializable	То же самое что и **IIIsolated**.

Обратите внимание на то, что не все типы провайдеров данных поддерживают все типы изоляции или работу с транзакциями.

Свойство **Attributes** устанавливает открывать ли новую транзакцию автоматически

xaCommitRetaining – при подтверждении транзакции

xaAbortRetaining – при отмене транзакции

Так же у компонента **TADOConnection** есть три метода для работы с транзакциями:

BeginTrans Начинает транзакцию

CommitTrans Подтверждает сделанные изменения

RollbackTrans Откатывает транзакцию.

Пример работы с транзакциями

За базовый возьмем пример использования **TADOConnection**. Добавим к форме две кнопки (**StCmButton** и **RollbackButton**) типа **TButton,** обработчики событий **OnClick** этих кнопок, процедуру **fix_controls** без параметров к форме, обработчик события **OnActivate** формы

Программу можно запускать. Этот пример находится в директории Trans.

Обратите внимание на то, что когда транзакция не запущена, все равно можно изменять данные и они записываются немедленно.

Доступ к данным

В отличие от BDE, ADO поддерживает больше настроек работы данных.

В ADO есть понятие набора данных (recordset) и тесно связанное с ним понятие курсора (cursor). Что такое курсор в документации на ADO не описано. Однако почему то месторасположение набора данных называется положением курсора. Я думаю, что это терминологическая путаница в Microsoft и курсор то же самое что набор данных.

Во всех компонентах имеющих набор данных (то есть в **TADODataSet, TADOTable, TADOQuery, TADOStoredProc**) есть свойства **CursorLocation**, **CursorType**, **LockType** и **MarshalOptions**, устанавливающие параметры обмена с сервером. Все эти свойства должны быть установлены до того, как набор данных открывается. Если вы установите их позже, то эффекта не будет.

CursorLocation – определяет, где выполняется работа с набором на клиенте (**clUseClient**) или на сервере (**clUseServer**). Если набор данных расположен на клиенте, то с сервера данные запрашиваются однократно (или до

```
type
TForm1 = class(TForm)
...
private
procedure fix_controls;
...
procedure TForm1.fix_controls;
begin
if Connection.InTransaction then
begin
StCmButton.Caption := 'Commit';
RollbackButton.Enabled := True;
end
else
begin
StCmButton.Caption := 'Begin';
RollbackButton.Enabled := False;
end;
MasterSQL.Requery;
DetailSQL.Requery;
end;

procedure TForm1.StCmButtonClick(Sender: TObject);
begin
if not Connection.InTransaction
then Connection.BeginTrans
else Connection.CommitTrans;
fix_controls;
end;

procedure TForm1.RollbackButtonClick(Sender: TObject);
begin
Connection.RollbackTrans;
fix_controls;
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
fix_controls;
end;
```

выполнения повторного запроса), в дальнейшем вся выборка данных и позиционирование идет на клиенте. Однако модификация данных производится немедленно.

CursorType – устанавливает тип курсора. Значение одно из:

ctUnspecified – библиотека ADO сама определяет оптимальный тип блокировки.

ctStatic – статический курсор. Статическая копия набора записей, которую вы можете использовать, например, для генерации отчета. Добавления, изменения или удаление записей другими пользователями не видимы.

ctOpenForwardOnly – идентичен статическому курсору, за исключением того, что вы можете переходить только вперед. Это тип улучшает эффективность в ситуациях, когда вы делаете только один проход через набор данных.

ctDynamic – динамический курсор. Добавления, изменения и удаление другими пользователями видимы и возможны все типы передвижения по набору данных. Закладки (bookmarks) возможны только, если провайдер данных их поддерживает.

ctKeyset – курсор набора данных. Аналогичен динамическому курсору, за исключением того, что вы не увидите записи добавленные другими пользователями, а записи удаленные другими пользователями недоступны из вашего набора данных. Изменения данных другими пользователями видимы.

Надо заметить, что **TDBGrid** и другие компоненты, одновременно работающие с несколькими записями, могут работать только когда закладки поддерживаются. Поэтому для компонент с которыми вы будите связывать такие компоненты должны использоваться типы **ctKeyset** или **ctDynamic.**

LockType – определяет тип блокировки записей в наборе данных. Оно из:

ltUnspecified – библиотека ADO сама определяет какой тип будет использоваться.

ltReadOnly – только чтение, изменение данных невозможно.

ltPessimistic – пессимистическая блокировка. Запись блокируется сразу после начала редактирования и до сохранения записей.

ltOptimistic – оптимистическая блокировка. Запись блокируется только когда изменения сохраняются.

ltBatchOptimistic – тоже самое что и **ltOptimistic**, но используется отложенное сохранение изменений записей. Более подробно она рассматривается в следующем пункте.

MarshalOptions – это свойство определяет будут ли отправлены на сервер те поля, которые не были изменены. При значении **moMarshalAll** будут, а при **moMarshalModifiedOnly** не будут.

Работа с отложенными изменениями

Обратите внимание, что в компонентах ADO нет свойства **CachedUpdates**, но это не означает, что невозможно отложить передачу изменений данных на сервер. Эта возможность встроена с ADO и называется Batch Updates.

Для ее использования необходимо использовать клиентский курсор (то есть установить свойство **CursorLocation** в **clUseClient**) и **LockType** в **ltBatchOptimistic**

Так же есть метод сохраняющий изменения **UpdateBatch** и метод их отменяющий **CancelBatch.**

К каким записям из набора данных применяется действие зависит от единственного параметра этих функций

arCurrent – текущая запись

arFiltered – записи, которые попали в фильтрацию.

arAll – все записи набора.

Пример работы с отложенными изменениями

За основу возьмем пример работы с транзакциями.

Добавим компоненты

BatchCB типа **TCheckBox**

ApplyButton типа **TButton**

CancelButton типа **TButton**

Добавим обработчики событий **OnClick** во все эти три компонента.

Изменим обработчик события **OnActivate** формы.

События ADO

События ADO предназначены для той же цели, что и события VCL. Многие из них имеют аналогичные события VCL и компоненты вызывают из событий ADO события VCL. В компонентах доступны как события ADO, так и события BDE.

События соединения

OnConnectComplete – после установки соединения.

OnDisconnect – при разрыве соединения.

```
procedure TForm1.FormActivate(Sender: TObject);
begin
  fix_controls;
  ApplyButton.Visible  := BatchCB.State = cbChecked;
  CancelButton.Visible := BatchCB.State = cbChecked;
end;

procedure TForm1.BatchCBClick(Sender: TObject);
begin
  MasterSQL.Close;
  DetailSQL.Close;
  if BatchCB.State = cbChecked then
   begin
     MasterSQL.LockType := ltBatchOptimistic;
     DetailSQL.LockType := ltBatchOptimistic;
   end
  else
   begin
     MasterSQL.LockType := ltOptimistic;
     DetailSQL.LockType := ltOptimistic;
   end;
  MasterSQL.Open;
  DetailSQL.Open;
  ApplyButton.Visible  := BatchCB.State = cbChecked;
  CancelButton.Visible := BatchCB.State = cbChecked;
end;

procedure TForm1.ApplyButtonClick(Sender: TObject);
begin
  MasterSQL.UpdateBatch;
  DetailSQL.UpdateBatch;
end;

procedure TForm1.CancelButtonClick(Sender: TObject);
begin
  MasterSQL.CancelBatch;
  MasterSQL.CancelBatch;
end;
```

Эти события инкапсулированы в компоненте **TADOConnection**.

События транзакции.

OnBeginTransComplete – при выполнении **BeginTrans**.

OnCommitTransComplete – при выполнении **CommitTrans**.

OnRollbackTransComplete – при выполнении **RollbackTrans**.

Эти события инкапсулированы в компоненте **TADOConnection**.

События выполнения команд

OnWillExecute и **OnExecuteComplete** вызываются перед и после выполнением команды.

Эти события инкапсулированы в компоненте **TADOConnection**, а не в компоненте **TADOCommand**, как можно было бы предположить. Это связано с тем, что в ADO объекта команды как такого нет и по этой причине он не может получать сообщения.

В **TADOConnection** также инкапсулировано событие **OnInfoMessage,** которое вызывается при приходе с сервера дополнительной информации. Формат и назначение зависят от сервера, с которым вы работаете.

В ADO так же есть события связанные с набором данных, а не с соединением, как вышеописанные. Они инкапсулированы в компоненты имеющие набор данных – **TADODataSet**, **TADOTable**, **TADOQuery** и **TADOStoredProc**.

Эти события можно разбить на три группы.

События выборки данных

OnFetchProgress – многократно вызывается в процессе выборки набора данных.

OnFetchComplete – завершение выборки.

Уведомления об изменении положения текущей записи в наборе.

OnWillMove, **OnMoveComplete** – вызываются до и после изменения положения текущей записи. **OnWillMove** позволяет отменить действие.

OnEndOfRecordset – вызывается при достижении конца набора данных, позволяет добавить новую запись.

Уведомления об изменении набора данных

OnWillChangeField, **OnFieldChangeComplete** – до и после изменения текущей записи набора.

OnWillChangeRecord, **OnRecordChangeComplete** – вызываются до и после изменения, добавления, удаления строки набора и отмене этих действий.

OnWillChangeRecordset, **OnRecordsetChangeComplete** - вызываются до и после открытия, закрытия, повторного запроса и синхронизации набора данных.

Многие события допускают прерывание действия. Эта возможность бывает полезна при асинхронной работе с сервером. Например, для прерывания слишком длинного запроса.

Асинхронная работа с сервером

В ADO есть возможность не имеющая аналогов ни в BDE ни в InterBase. Это асинхронное выполнение операций с сервером. Могут асинхронно выполнятся установка соединения с сервером (Connection), выполнение команды (Execute) и выборка набора данных (Fetch)

Асинхронное соединение

Для включения этого режима необходимо установить свойство **ConnectOptions** компонента **TADOConnection** в **coAsyncConnect**.

При установлении соединения происходит

Вызывается обработчик события **OnWillConnect**

Управление передается в программу

После окончания соединения, как успешного, так и ошибочного, вызывается обработчик события **OnConnectComplete**

Асинхронное выполнение команды

Надо заметить, что все компоненты ADO, за исключением компонента **TADOConnection** при активизации или выполнении исполняют команду ADO. Эти компоненты **TADOCommand, TADODataSet, TADOTable, TADOQuery, TADOStoredProc**. Установите в свойстве **ExecuteOptions eoAsyncExecute**.

При исполнении происходит

Вызывается обработчик события **OnWillExecute**

Управление передается в программу

После окончания выполнения команды, как успешного, так и ошибочного, вызывается обработчик события **OnExecuteComplete**

Асинхронная выборка данных

Асинхронная выборка данных поддерживается в компонентах **TADODataSet, TADOTable, TADOQuery, TADOStoredProc**. Для ее включения установите в свойстве **ExecuteOptions eoAsyncFetch**.

После исполнения команды происходит передача данных. В процессе передачи многократно вызывается обработчик сообщения **OnFetchProgress**. В его параметрах есть как количество уже переданных записей, так и общее количество. Это очень удобно для создания индикаторов типа TProgressBar. После того, как выборка с сервера закончена, вызывается обработчик события **OnFetchComplete**.

2. ПРОГРАММА РАБОТЫ

1. Ознакомиться с теоретическими положениями.
2. Разработать на прикладном языке программирования произвольную таблицу с любой комбинацией атрибутов предыдущих лабораторных работ.
3. Значения атрибутов с помощью технологии ADO передать из реляционных баз данных, разработанных ранее, в построенную таблицу.
4. Предусмотреть в прикладной программе какую-либо математическую обработку данных в столбцах пользовательской таблице (например, среднее значение или сумма значений или т.п.)

Лабораторная работа № 12

ИСПОЛЬЗОВАНИЕ МЕХАНИЗМОВ ЗАЩИТЫ БАЗЫ ДАННЫХ

1. ЦЕЛЬ РАБОТЫ

Изучение методов и построение алгоритмов обеспечения защиты базы данных.

2. ОСНОВЫ ТЕОРИИ

Защита информации в базах данных, в отличие от защиты данных в файлах, имеет и свои особенности:

- необходимость учета функционирования системы управления базой данных при выборе механизмов защиты;
- разграничение доступа к информации реализуется не на уровне файлов, а на уровне частей баз данных.

При создании средств защиты информации в базах данных необходимо учитывать взаимодействие этих средств не только с ОС, но и с СУБД. При этом возможно встраивание механизмов защиты в СУБД или использование их в виде отдельных компонент. Для большинства СУБД придание им дополнительных функций возможно только на этапе разработки СУБД. В эксплуатируемые системы управления базами данных дополнительные компоненты могут быть внесены путем расширения или модификации языка управления. Таким путем можно осуществлять наращивание возможностей, например, в СУБД CA-Clipper 5.0.

В современных базах данных довольно успешно решаются задачи разграничения доступа, поддержания физической целостности и логической сохранности данных. Алгоритмы разграничения доступа к записям и даже к полям записей в соответствии с полномочиями пользователя хорошо отработаны, и преодолеть эту защиту злоумышленник может лишь с помощью фальсификации полномочий или внедрения вредительских программ. Разграничение доступа к файлам баз данных и к частям баз данных осуществляется СУБД путем установления полномочий пользователей и контроля этих полномочий при допуске к объектам доступа.

Полномочия пользователей устанавливаются администратором СУБД. Обычно стандартным идентификатором пользователя является пароль, передаваемый в зашифрованном виде. В распределенных КС процесс подтверждения подлинности пользователя дополняется специальной процедурой взаимной аутентификации удаленных процессов. Базы данных, содержащих конфиденциальную информацию, хранятся на внешних запоминающих устройствах в зашифрованном виде.

Физическая целостность баз данных достигается путем использования отказоустойчивых устройств, построенных, например, по технологии RAID. Логическая сохранность данных означает невозможность нарушения структуры моде-

ли данных. Современные СУБД обеспечивают такую логическую целостность и непротиворечивость на этапе описания модели данных.

В базах данных, работающих с конфиденциальной информацией, необходимо дополнительно использовать криптографические средства закрытия информации. Для этой цели используется шифрование как с помощью единого ключа, так и с помощью индивидуальных ключей пользователей. Применение шифрования с индивидуальными ключами повышает надежность механизма разграничения доступа, но существенно усложняет управление.

Возможны два режима работы с зашифрованными базами данных. Наиболее простым является такой порядок работы с закрытыми данными, при котором для выполнения запроса необходимый файл или часть файла расшифровывается на внешнем носителе, с открытой информацией производятся необходимые действия, после чего информация на ВЗУ снова зашифровывается. Достоинством такого режима является независимость функционирования средств шифрования и СУБД, которые работают последовательно друг за другом. В то же время сбой или отказ в системе может привести к тому, что на ВЗУ часть базы данных останется записанной в открытом виде.

Второй режим предполагает возможность выполнения СУБД запросов пользователей без расшифрования информации на ВЗУ. Поиск необходимых файлов, записей, полей, групп полей не требует расшифрования. Расшифрование производится в ОП непосредственно перед выполнением конкретных действий с данными. Такой режим возможен, если процедуры шифрования встроены в СУБД. При этом достигается высокий уровень защиты от несанкционированного доступа, но реализация режима связана с усложнением СУБД. Придание СУБД возможности поддержки такого режима работы осуществляется, как правило, на этапе разработки СУБД.

При построении защиты баз данных необходимо учитывать ряд специфических угроз безопасности информации, связанных с концентрацией в базах данных большого количества разнообразной информации, а также с возможностью использования сложных запросов обработки данных. К таким угрозам относятся:

- инференция;
- агрегирование;
- комбинация разрешенных запросов для получения закрытых данных.

Под **инференцией** понимается получение конфиденциальной информации из сведений с меньшей степенью конфиденциальности путем умозаключений. Если учитывать, что в базах данных хранится информация, полученная из различных источников в разное время, отличающаяся степенью обобщенности, то аналитик может получить конфиденциальные сведения путем сравнения, дополнения и фильтрации данных, к которым он допущен. Кроме того, он обрабатывает информацию, полученную из открытых баз данных, средств массовой информации, а также использует просчеты лиц, определяющих степень важности и конфиденциальности отдельных явлений, процессов, фактов, полученных результатов. Такой способ получения конфиденциальных сведений, например, по материалам средств массовой информации, используется давно, и показал свою эффективность.

Близким к инференции является другой способ добывания конфиденциальных сведений - **агрегирование**. Под агрегированием понимается способ получения более важных сведений по сравнению с важностью тех отдельно взятых данных, на основе которых и получаются эти сведения. Так, сведения о деятельности одного отделения или филиала корпорации обладают определенным весом. Данные же за всю корпорацию имеют куда большую значимость.

Если инференция и агрегирование являются способами добывания информации, которые применяются не только в отношении баз данных, то способ специального **комбинирования запросов** используется только при работе с базами данных. Использование сложных, а также последовательности простых логически связанных запросов позволяет получать данные, к которым доступ пользователю закрыт. Такая возможность имеется, прежде всего, в базах данных, позволяющих получать статистические данные. При этом отдельные записи, поля, (индивидуальные данные) являются закрытыми. В результате запроса, в котором могут использоваться логические операции AND, OR, NOT, пользователь может получить такие величины как количество записей, сумма, максимальное или минимальное значение. Используя сложные перекрестные запросы и имеющуюся в его распоряжении дополнительную информацию об особенностях интересующей записи (поля), злоумышленник путем последовательной фильтрации записей может получить доступ к нужной записи (полю).

Противодействие подобным угрозам осуществляется следующими методами:

- блокировка ответа при неправильном числе запросов;
- искажение ответа путем округления и другой преднамеренной коррекции данных;
- разделение баз данных;
- случайный выбор записи для обработки;
- контекстно-ориентированная защита;
- контроль поступающих запросов.

Метод **блокировки ответа** при неправильном числе запросов предполагает отказ в выполнении запроса, если в нем содержится больше определенного числа совпадающих записей из предыдущих запросов. Таким образом, данный метод обеспечивает выполнение принципа минимальной взаимосвязи вопросов. Этот метод сложен в реализации, так как необходимо запоминать и сравнивать все предыдущие запросы.

Метод **коррекции** заключается в незначительном изменении точного ответа на запрос пользователя. Для того, чтобы сохранить приемлемую точность статистической информации, применяется так называемый свопинг данных. Сущность его заключается во взаимном обмене значений полей записи, в результате чего все статистики i-го порядка, включающие i атрибутов, оказываются защищенными для всех i, меньших или равных некоторому числу. Если злоумышленник сможет выявить некоторые данные, то он не сможет определить, к какой конкретно записи они относятся.

Применяется также метод **разделения баз данных** на группы. В каждую группу может быть включено не более определенного числа записей. Запросы разрешены к любому множеству групп, но запрещаются к подмножеству записей из

одной группы. Применение этого метода ограничивает возможности выделения данных злоумышленником на уровне не ниже группы записей. Метод разделения баз данных не нашел широкого применения из-за сложности получения статистических данных, обновления и реструктуризации данных.
Эффективным методом противодействия исследованию баз данных является метод **случайного выбора записей** для статистической обработки. Такая организация выбора записей не позволяет злоумышленнику проследить множество запросов.
Сущность **контекстно-ориентированной защиты** заключается в назначении атрибутов доступа (чтение, вставка, удаление, обновление, управление и т. д.) элементам базы данных (записям, полям, группам полей) в зависимости от предыдущих запросов пользователя. Например, пусть пользователю доступны в отдельных запросах поля: "идентификационные номера" и "фамилии сотрудников", а также "идентификационные номера" и "размер заработной платы". Сопоставив ответы по этим запросам, пользователь может получить закрытую информацию о заработной плате конкретных работников. Для исключения такой возможности пользователю следует запретить доступ к полю "идентификатор сотрудника" во втором запросе, если он уже выполнил первый запрос.
Одним из наиболее эффективных методов защиты информации в базах данных является **контроль поступающих запросов** на наличие "подозрительных" запросов или комбинации запросов. Анализ подобных попыток позволяет выявить возможные каналы получения несанкционированного доступа к закрытым данным.

3. ОБЪЕКТЫ И СРЕДСТВА ИССЛЕДОВАНИЯ

Предметная область в соответствии с вариантом задания. ПК типа IBM PC, дисплейный класс, операционная система Windows XP и выше.

4. ПОДГОТОВКА К РАБОТЕ

Ознакомиться с теоретическими положениями лабораторной работы.

5. ПРОГРАММА РАБОТЫ

По предложенному преподавателем варианту разработать алгоритм, моделирующий механизм защиты информации в базе данных. Оформить отчет.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

6.1. Назовите основные подходы к защите баз данных ?
6.2. Основные методы защиты?

Лабораторная работа № 13

УПРАВЛЕНИЕ ПРИВИЛЕГИЯМИ ДОСТУПА К БАЗАМ ДАННЫХ

1. ЦЕЛЬ РАБОТЫ

Изучение методов управления привилегиями доступа к базам данных.

2. ОСНОВЫ ТЕОРИИ

Привилегии пользователей назначаются им администратором базы данных и определяют какие действия над данными и над объектами схемы являются разрешенными. При контроле привилегий используется имя пользователя базы данных, называемое иногда идентификатором авторизации (Autorization ID). Некоторые СУБД идентифицируют понятие "пользователь" с понятием "учетная запись".

Все объекты пользователя БД входят в его схему. На практике один пользователь, как правило, ассоциируется с одной схемой, хотя стандарт подразумевает, что одному пользователю может принадлежать несколько схем, содержащих взаимосвязанные объекты.

После успешного завершения процедуры идентификации открывается сеанс пользователя и устанавливается соединение с базой данных.

Существуют привилегии двух типов:

- системные привилегии (system privileges), контролирующие общий доступ к базе данных;
- объектные привилегии (object privileges), контролирующие доступ к конкретным объектам базы данных.

Синтаксис, используемый для работы с привилегиями, на практике значительно шире стандарта, но в значительной степени зависит от архитектуры конкретной БД.

Для управления привилегиями определены следующие правила:

- объект принадлежит пользователю, его создавшему (если синтаксисом не указано создание объекта другого пользователя, конечно, при соответствующих полномочиях);
- владелец объекта, согласно стандарту, может изменять привилегии своего объекта (в коммерческих СУБД, таких как Oracle, уровни полномочий представляют собой более сложную иерархию);
- объектная привилегия всегда соотносится с конкретным объектом, а системная - с объектами вообще.

Язык SQL поддерживает следующие привилегии:

- ALTER - позволяет выполнять оператор ALTER TABLE;

- SELECT - позволяет выполнять оператор запроса;
- INSERT - позволяет выполнять добавление строк в таблицу;
- UPDATE - позволяет изменять значения во всей таблице или только в некоторых столбцах;
- DELETE - позволяет удалять строки из таблицы;
- REFERENCES - позволяет устанавливать внешний ключ с использованием в качестве родительского ключа любых столбцов таблицы или только некоторых из них;
- INDEX - позволяет создавать индексы (не входит в стандарт SQL-92);
- DROP - позволяет удалять таблицу из схемы базы данных.

Предоставление и снятие привилегий

Предоставление привилегии выполняется SQL-оператором GRANT, который имеет в стандарте SQL-92 следующее формальное описание:

```
ON { [TABLE] table_name
    | DOMAIN domain_name
    | COLLATION collation_name
    | CHARACTER SET set_name
    | TRANSLATION translation_name }
TO { user_name .,: } | PUBLIC
[ WITH GRANT OPTION ]
```

где privilege определяется как

```
{ ALL PRIVELEGES }
| SELECT
| DELETE
| INSERT [(field .,:)]
| UPDATE [(field .,:)]
| REFERENCES [(field .,:)]
| USAGE
```

После фразы GRANT через запятую можно перечислить список всех назначаемых привилегий.

Фраза ON определяет объект, для которого устанавливается привилегия.

Фраза TO указывает пользователя или пользователей, для которых устанавливается привилегия.

Так, оператор GRANT SELECT ON tbl1 TO PUBLIC; предоставляет доступ к выполнению оператора SELECT для таблицы tbl1 не только всем существующим пользователям, но и тем, которые позднее будут добавлены в базу данных.

Оператор GRANT UPDATE ON tbl1 TO user1; предоставляет пользователю user1 привилегию UPDATE на всю таблицу, а оператор GRANT UPDATE (f1,f2) ON tbl1 TO user1 предоставляет привилегию UPDATE для изменения только столбцов f1 и f2.

Фраза WITH GRANT OPTION предоставляет получающему привилегию пользователю дополнительную привилегию GRANT OPTION,позволяющую выполнять передачу полученных привилегий.

Отмена привилегии выполняется SQL-оператором REVOKE, который имеет в стандарте SQL-92 следующее формальное описание:

```
REVOKE [ GRANT OPTION FOR ]
   { ALL PRIVILEGES } | privilege
   ON { [TABLE] table_name
      | DOMAIN domain_name
```

```
    | COLLATION collation_name
    | CHARACTER SET set_name
    | TRANSLATION translation_name }
  FROM { PUBLIC | user_name .,: }
  [ CASCADE | RESTRICT ]
```

После фразы REVOKE через запятую можно перечислить список всех отменяемых привилегий.

Фраза ON определяет объект, для которого отменяется привилегия.

Фраза FROM указывает пользователя или пользователей, для которых отменяется привилегия.

Фраза GRANT OPTION FOR определяет отмену не самих привилегий, а только права их передачи другим пользователям.

Если одна привилегия вместе с опцией WITH GRANT OPTION была последовательно передана от одного пользователя другому несколько раз, то образуется цепочка зависимых привилегий. Фразы CASCADE и RESTRICT определяют, что будет происходить с этими привилегиями при отмене одного из звеньев этой цепочки.

Если при отмене зависимой привилегии для объекта не остается ни одной существующей привилегии, то такой объект называется несостоявшимся. Например, подобное может произойти с представлением, созданным как запрос к таблице, привилегия на которую была утрачена.

Если при отмене привилегии появляется несостоявшийся объект, то фраза RESTRICT предотвратит выполнение оператора REVOKE, и никакие привилегии отменены не будут.

Если указана фраза CASCADE и при отмене привилегии появляется несостоявшийся объект, то все несостоявшиеся объекты (представления) удаляются, а при наличии несостоявшихся ограничений в таблицах они отменяются автоматически выполнением оператора ALTER TABLE несостоявшиеся ограничения в доменах отменяются автоматически выполнением оператора ALTER DOMAIN.

Роли

Ролью называется именованный набор привилегий. Объединение привилегий в роли значительно упрощает процесс назначения и снятия привилегий. Если СУБД поддерживает управление ролями, то в SQL-операторах GRANT и REVOKE вместо имени пользователя можно указывать имя роли.

Многоуровневый контроль доступа в БД Oracle

Среди современных коммерческих СУБД базу данных Oracle можно считать одной из самых продвинутых в области контроля доступа. Все привилегии делятся на системные и объектные.

Синтаксис оператора GRANT, выполняющего предоставление пользователям или ролям системных полномочий и ролей, может быть представлен следующей схемой:

предоставление пользователям или ролям системных полномочий и ролей,

Предоставление пользователям или ролям привилегий над обычными объектами БД Oracle также может быть показано на примере следующей схемы:

Предоставление пользователям или ролям привилегий над обычными объектами БД Oracle

system_priv - предоставляемое системное полномочие.
role - предоставляемая роль.
TO -определяет пользователей или роли, которым предоставляются системные или объектные полномочия.
PUBLIC - указывает, что системные или объектные полномочия, определяемые оператором, предоставляются всем пользователям.
WITH ADMIN OPTION - позволяет пользователю, получившему системные или объектные полномочия или роль, предоставлять их в дальнейшем другим пользователям или ролям. Такое разрешение включает и возможность изменения или удаления роли. (Синтаксически данная опция отличается от стандарта SQL-92.)
object_priv - определяет предоставляемую привилегию, которая может быть указана одним из следующих значений:

- ALTER
- DELETE
- EXECUTE (только для процедур, функций и пакетов)
- INDEX (только для таблиц)
- INSERT
- REFERENCES (только для таблиц)
- SELECT
- UPDATE.

column - определяет столбец таблицы или представления, на который распространяется предоставляемая привилегия.
ON - определяет объект (таблицу, вид, хранимую процедуру, снимок), на который предоставляется привилегия.

Например:

```
GRANT SELECT, UPDATE ON tbl1 TO PUBLIC
GRANT REFERENCES  (f1), UPDATE (f1, f2, f3)
    ON user1.tbl1 TO user2
```

Приведем список предоставляемых оператором GRANT системных полномочий, который характеризует систему контроля доступа БД Oracle. К системным полномочиям относятся следующие:
ALTER ANY CLUSTER - разрешает получившему эти полномочия изменение любого кластера в любой схеме;
ALTER ANY INDEX - разрешает изменение любого индекса в любой схеме;
ALTER ANY PROCEDURE - разрешает изменение любой хранимой функции, процедуры или пакета в любой схеме;
ALTER ANY ROLE - разрешает изменение в базе данных любой роли;
ALTER ANY SEQUENCE - разрешает изменение в базе данных любой последовательности;
ALTER ANY SNAPSHOT - разрешает изменение в базе данных любого снимка;
ALTER ANY TABLE - разрешает изменение в схеме любой таблицы или вида;
ALTER ANY TRIGGER - позволяет разрешать, запрещать или компилировать любой триггер базы данных в любой схеме; изменение в базе данных любой роли;
ALTER DATABASE- разрешает изменение базы данных;
ALTER PROFILE - разрешает изменение профилей;
ALTER RESOURCE COST - разрешает устанавливать цену ресурсов сеанса работы пользователя;
ALTER ROLLBACK SEGMENT - разрешает изменение сегментов отката;
ALTER SESSION - разрешает выполнение оператора ALTER SESSION;

ALTER SYSTEM - разрешает выполнение оператора ALTER SYSTEM;
ALTER TABLESPACE - разрешает изменение табличных пространств;
ALTER USER - разрешает изменение параметров для любого пользователя (пароль, количество доступного табличного пространства, назначенный профиль и т.п.);
ANALYZE ANY - разрешает анализировать таблицу, кластер или индекс в любой схеме;
AUDIT ANY - разрешает выполнять аудит любого объекта в любой схеме;
AUDIT SYSTEM - разрешает выполнение SQL-оператора AUDIT;
BACKUP ANY TABLE - позволяет выполнять экспорт объектов из схемы других пользователей;
BECOME USER - позволяет становиться другим пользователем (требуется при импорте БД);
COMMENT ANY TABLE - разрешает получившему эти полномочия комментарий для любой таблицы, вида или столбца в любой схеме;
CREATE ANY CLUSTER ;
CREATE ANY INDEX ;
CREATE ANY LIBRARY ;
CREATE ANY PROCEDURE ;
CREATE ANY SEQUENCE ;
CREATE ANY SNAPSHOT ;
CREATE ANY SYNONYM ;
CREATE ANY TABLE ;
CREATE ANY TRIGGER ;
CREATE ANY VIEW ;
CREATE CLUSTER - разрешает создавать кластер в своей схеме (системное полномочие, не содержащее в названии фразу ANY, распространяется только на собственную схему пользователя);
CREATE DATABASE LINK - разрешает создавать линк базы данных в своей схеме;
CREATE PROCEDURE ;
CREATE PROFILE ;
CREATE PUBLIC DATABASE LINK - разрешает создавать общедоступные линки базы данных;
CREATE PUBLIC SYNONYM ;
CREATE ROLE - разрешает создание ролей;
CREATE ROLLBACK SEGMENT ;
CREATE LIBRARY ;
CREATE SEQUENCE;
CREATE SESSION - разрешает соединение с базой данных;
CREATE SNAPSHOT;
CREATE SYNONYM;
CREATE TABLE;
CREATE TABLESPACE ;
CREATE TRIGGER;
CREATE USER ;
CREATE VIEW ;
DELETE ANY TABLE ;
DROP ANY CLUSTER ;
DROP ANY INDEX - разрешает удаление любого индекса;
DROP ANY LIBRARY ;
DROP ANY PROCEDURE ;
DROP ANY ROLE ;

DROP ANY SEQUENCE ;
DROP ANY SNAPSHOT ;
DROP ANY SYNONYM ;
DROP ANY TABLE ;
DROP ANY TRIGGER ;
DROP ANY VIEW ;
DROP LIBRARY ;
DROP PROFILE ;
DROP PUBLIC DATABASE LINK ;
DROP PUBLIC SYNONYM ;
DROP ROLLBACK SEGMENT ;
DROP TABLESPACE ;
DROP USER ;
EXECUTE ANY PROCEDURE - разрешает выполнение процедур и функций, а также ссылки на общедоступные переменные пакетов в любой схеме;
FORCE ANY TRANSACTION - позволяет выполнять фиксацию или откат любой сомнительной распределенной транзакции на локальной базе данных, а также определять сбой распределенной транзакции;
FORCE TRANSACTION - позволяет выполнять фиксацию или откат собственной сомнительной распределенной транзакции на локальной базе данных;
GRANT ANY PRIVILEGE ;
GRANT ANY ROLE ;
INSERT ANY TABLE ;
LOCK ANY TABLE ;
MANAGE TABLESPACE - разрешает переключение табличного пространства из автономного режима в оперативный или обратно, а также разрешает выполнять копирование табличного пространства;
RESTRICTED SESSION ;
SELECT ANY SEQUENCE ;
SELECT ANY TABLE ;
UNLIMITED TABLESPACE - разрешает неограниченное использование любого табличного пространства.Предоставление этого полномочия перекрывает любые ограничения на количество доступного табличного пространства, ранее установленные для пользователя;
UPDATE ANY TABLE - разрешает изменение строк в таблицах и видах любой схемы.

При создании базы данных Oracle некоторые роли создаются автоматически. Следующая таблица содержит названия автоматически создаваемых Oracle ролей и список предоставляемых ими системных полномочий. Эти роли создают иерархию предоставляемых полномочий.

Роль	Предоставляемые системные полномочия и роли
CONNECT	ALTER SESSION CREATE CLUSTER CREATE DATABASE LINK CREATE SEQUENCE CREATE SESSION CREATE SYNONYM CREATE TABLE CREATE VIEW

RESOURCE	CREATE CLUSTER CREATE PROCEDURE CREATE SEQUENCE CREATE TABLE CREATE TRIGGER
DBA	Все системные полномочия WITH ADMIN OPTION EXP_FULL_DATABASE (роль) IMP_FULL_DATABASE (роль)
EXP_FULL_DATABASE	SELECT ANY TABLE BACKUP ANY TABLE INSERT, UPDATE, DELETE ON sys.incexp, sys.incvid, sys.incfil

Для просмотра предоставленных привилегий администратор базы данных может использовать следующие системные представления словаря данных:

Системное представление	Описание
ALL_COL_PRIVS	Содержит список привилегий, предоставленных для столбцов таблицы другим пользователям или PUBLIC. ; Содержит столбцы: GRANTOR (кто предоставляет привилегию) GRANTEE (кому предоставляется привилегия) TABLE_SCHEMA (схема объекта) TABLE_NAME COLUMN_NAME PRIVILEGE
ALL_COL_PRIVS_MADE	Содержит список привилегий столбцов, которыми владеет пользователь или предоставляет на них привилегии. Содержит столбцы: GRANTEE OWNER GRANTOR TABLE_NAME COLUMN_NAME PRIVILEGE
ALL_TAB_PRIVS	Содержит список привилегий, предоставленных для таблицы другим пользователям или PUBLIC. Содержит столбцы: GRANTOR GRANTEE TABLE_NAMEPRIVILEGE
DBA_PROFILES	Содержит описания всех профилей базы данных и определяемых ими ограничений. Содержит столбцы: PROFILE RESOURCE_NAME

	LIMIT
DBA_ROLES	Содержит список имен всех существующих в базе данных ролей. Содержит столбцы: ROLE PASSWORD_REQUIRED
DBA_ROLE_PRIVS	Содержит список ролей, предоставляемых другим пользователям или ролям. Содержит столбцы: GRANTEE (кто получает полномочия) GRANTED_ROLE (предоставляемая роль)
DBA_SYS_PRIVS	Содержит список системных полномочий, предоставленных пользователям и ролям. Содержит столбцы: GRANTEE (кто получает полномочия) PRIVILEGE (название системного полномочия) ADMIN_OPTION
DBA_TAB_PRIVS	Содержит список всех предоставленных полномочий для объектов базы данных. Содержит столбцы: GRANTEE OWNER TABLE_NAME GRANTOR PRIVILEGE
DBA_USERS	Содержит информацию обо всех пользователях базы данных. Содержит столбцы: USERNAME USER_ID PASSWORD DEFAULT_TABLESPACE PROFILE
ROLE_SYS_PRIVS	Содержит информацию о предоставленных ролям системных полномочиях. Содержит столбцы: ROLE PRIVILEGE
ROLE_TAB_PRIVS	Содержит информацию о предоставленных ролям привилегиях для таблиц и столбцов. Содержит столбцы: ROLE OWNER TABLE_NAME COLUMN_NAME PRIVILEGE
SYSTEM_PRIVILEGE_MAP	Содержит список всех системных полномочий
TABLE_PRIVILEGE_MAP	Содержит информацию о кодах привилегий доступа к таблицам и столбцам.

USER_ROLE_PRIVS	Содержит список ролей, предоставленных пользователю. Содержит столбцы: USERNAME GRANTED_ROLE
USER_SYS_PRIVS	Содержит список всех системных полномочий, предоставленных пользователю. Содержит столбцы: USERNAME PRIVILEGE
USER_TAB_PRIVS	Содержит список привилегий для объектов, где пользователь является владельцем, получателем или лицом, предоставляющим привилегии. Содержит столбцы: GRANTEE (кому предоставляется привилегия) OWNER TABLE_NAME (имя объекта) GRANTOR (кто предоставляет привилегию)
USER_TAB_PRIVS_MADE	Содержит список всех предоставлений привилегий для объектов, принадлежащих пользователю. Содержит столбцы: GRANTEE GRANTOR TABLE_NAME PRIVILEGE
USER_TAB_PRIVS_RECD	Содержит список всех привилегий для объектов, где пользователь является получателем привилегии. Содержит столбцы: OWNER (владелец объекта) TABLE_NAME (имя объекта) GRANTOR (имя пользователя, предоставившего привилегию) PRIVILEGE

Эти системные представления позволяют администратору БД Oracle полностью контролировать назначение, передачу и взаимозависимость системных и объектных привилегий.

3. ОБЪЕКТЫ И СРЕДСТВА ИССЛЕДОВАНИЯ

Предметная область в соответствии с вариантом задания. ПК типа IBM PC, дисплейный класс, операционная система Windows XP и выше.

4. ПОДГОТОВКА К РАБОТЕ

Ознакомиться с теоретическими положениями лабораторной работы.

5. ПРОГРАММА РАБОТЫ

По предложенному преподавателем варианту разработать алгоритм, моделирующий механизм защиты информации в базе данных. Оформить отчет.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

6.1. Назовите основные подходы к защите баз данных?
6.2. Основные методы защиты?

Лабораторная работа № 14

АУДИТ ДАННЫХ С ПОМОЩЬЮ СРЕДСТВ СУБД

1. ЦЕЛЬ РАБОТЫ

Изучение методов аудита данных.

2. ОСНОВЫ ТЕОРИИ

В большинстве современных информационных систем протоколирование операций пользователей решается с помощью журнала событий. Однако бывают события, которые в таком журнале не отражаются, но способны нарушить целостность критически важной информации.

Большинство информационных систем строятся по схеме «клиент – сервер приложений – база данных», причем задача протоколирования выполняемых пользователями операций решается на уровне сервера приложений. Для этого, например, в прикладной системе может быть реализован служебный компонент для ведения журнала событий (операций), в котором регистрируются все значимые действия по просмотру и модификации информации. Преимущество такого подхода в том, что он позволяет регистрировать события в терминах предметной области, а не на уровне SQL-операций над таблицами базы данных. Казалось бы, подобный компонент, фиксирующий действия пользователей, решает все проблемы мониторинга операций и нет необходимости включения дополнительных средств на уровне СУБД, однако это не так. В крупных компаниях и организациях, имеющих большое количество взаимосвязанных систем, возникает острая необходимость контролировать критические изменения в данных не только по журналу событий, отражающих действия пользователя, но и на уровне операций, производимых в базе данных посредством SQL-команд или путем загрузки данных из разных источников.

Мониторинг на уровне СУБД

У любой прикладной системы обычно имеется администратор, сопровождающий ее с помощью SQL-интерфейса, и его действия, как правило, в журнале событий прикладной системы не отражаются. В результате на уровне приложения не остается никаких следов, позволяющих обнаружить возможные несанкционированные действия. Кроме этого, в самой прикладной системе могут быть уязвимости, позволяющие ее взломать, например, при помощи популярного среди хакеров метода SQL-инъекций, что может привести к непредсказуемым изменениям в базе, которые не будут зафиксированы в журнале событий. Наконец, часто возникает необходимость проведения прямых изменений в базе посредством SQL-команд, минуя интерфейс прикладной системы. Например, в задачу некоторых прикладных систем входит ежедневное выполнение расчетов определенных показателей (финансовых обязательств, биржевых котировок, и т. п.) на основе нормативных или оперативных данных, загружаемых из внешних источников. Бывают ситуации, когда перед расчетом выясняется, что поступившие в систему данные некорректны, а времени на их повторную загрузку уже нет, и в этом случае ошибки обычно исправляются вручную путем прямых изменений в базе. Очевидно, что такие действия необходимо контролировать, поскольку велика вероятность того, что исправление будет не-

верным. При этом важно отслеживать, что все изменения произведены до запуска расчета, а не в процессе его выполнения или окончания.

Все эти примеры свидетельствуют о том, что журнала событий прикладной системы недостаточно — необходим аудит изменений данных на уровне базы. Несмотря на инструментальную роль, отводимую базам данных в архитектуре современных информационных систем, в условиях, когда подавляющее большинство пользователей не употребляют SQL и вообще ничего не знают о схеме базы и структуре таблиц, с которыми они работают, применение средств мониторинга изменений на уровне базы оказывается полезным дополнительным инструментом для обеспечения безопасности и исключения ошибок при эксплуатации системы. Основным критерием для выбора механизма, обеспечивающего мониторинг, является минимальное влияние внедряемого решения на производительность системы.

Мониторинг с помощью триггеров

Наиболее известным способом мониторинга изменений является использование механизма триггеров, с помощью которых можно осуществлять перехват и протоколирование операций, производимых в базе, — модификацию данных (операции DML (Data Manipulation Language): insert, update, delete) и модификацию схемы (операции DDL (Data Definition Language): create table, alter table add column и т. п.). Этот метод также действует во встроенных механизмах аудита современных СУБД (например, при использовании команды AUDIT в Oracle), где система в автоматическом режиме создает и актуализирует набор триггеров, соответствующий текущим правилам контроля объектов базы, типов операций и т. п.

Основным недостатком применения триггеров и встроенного аудита для мониторинга изменений является падение производительности системы, поскольку в этом случае каждая операция в транзакции дополнительно сопровождается добавлением записи в таблицу аудита. Реально использовать встроенный аудит СУБД возможно только в том случае, когда удается заранее установить селективные условия аудита (например, настроить фильтр на протоколирование данных из небольшого подмножества таблиц базы, чтобы обеспечить обозримый объем выборки записей для анализа). Если же подключить к аудиту все операции над базой или их большую часть, то это сильно повлияет на производительность прикладной системы, и выполнять подобные операции на «боевой» системе, находящейся под постоянной или временами пиковой нагрузкой, обычно неприемлемо.

Мониторинг по журналу транзакций

С учетом названных ограничений получил развитие альтернативный подход к реализации мониторинга изменений, основанный на возможности извлечения и последующей обработки информации об операциях DML и DDL, имеющейся в журнале транзакций базы. Для основных СУБД (Oracle, Microsoft SQL Server и IBM DB2) созданы промышленные системы аудита, предлагаемые в виде отдельных продуктов: Oracle Audit Vault, Apex SQL Log и IBM Audit Management Expert. Эти системы обладают возможностями захвата изменений из журналов транзакций и позволяют аудитору удаленно подключаться к журналам транзакций целевых баз (включая архивы журналов), задавать разнообразные фильтры на выборку интересующих его записей, получать удобные для анализа отчеты в различных форматах и пр. Общим недостатком этих систем, несмотря на различия в их архитектуре, является то, что захват изменений осуществляется непосредственно на целевой базе — этот процесс достаточно «тяжелый», поскольку связан с парсингом и трансформацией данных.

По сравнению с подключением встроенного аудита СУБД способ захвата нагружает систему не так значительно, однако это не всегда можно делать на исходной базе, особенно когда прикладная система активно выполняет свою целевую функцию. Дополнительно следует отметить, что мониторинг, выполняемый по журналу транзакций, позволяет извлекать информацию о давно произведенных операциях в базе. Если изменения по каким-то объектам до этого не отслеживались, то аудитор тем не менее может запросить историю модификации любого из них за прошлый период по журналу транзакций. Очевидно, что эта функциональность полезна и востребована, но ее поддержка на сервере прикладной системы в критической степени сказывается на производительности системы, поскольку исполнение такого рода запросов вызывает необходимость выполнения ресурсоемкого сканирования большого количества (несколько тысяч) архивных файлов журнала транзакций.

Архитектура сервера контроля изменений

Перспективным можно считать вариант, когда серверы с базами данных информационной системы полностью избавляются от «непрофильной» работы, а выполнение задач по захвату, обработке и сохранению информации об изменениях осуществляется на специально выделенном сервере. На рис. 1 приведена предлагаемая архитектура системы такого класса.

Рис. 1. Архитектура сервера контроля изменений

Центральным звеном архитектуры является **сервер контроля изменений** (СКИ), на котором выполняется процесс захвата и обработки записей из архивных журналов, поступающих с серверов, содержащих исходные базы. Предполагается, что любая СУБД поддерживает механизм архивирования журналов транзакций, применяемый для решения задач резервного копирования и восстановления данных. Именно эти архивные журналы передаются на СКИ и обрабатываются, формируя структурированное представление о прочитанном из журнала изменении, которое в виде отдельной записи (или группы записей) может быть сохранено в базе СКИ.

СКИ может обслуживать несколько исходных баз, для каждой из которых порождается свой процесс захвата и обработки изменений. Основная особенность предложенной архитектуры заключается в том, что сами исходные базы на сервер СКИ не перемещаются, а метаданные о структуре базы, составляющие содержимое словаря данных (место размещения названий таблиц, полей и т. п.), могут быть получены из архивного журнала или каким-то иным способом переданы на СКИ. Не

имея этих метаданных (а они являются неотъемлемой частью любой базы данных и используются при чтении файлов журнала), анализировать записи журналов невозможно — они закодированы, и, например, вместо имен полей используются их номера.

К числу дополнительных преимуществ, вытекающих из данной архитектуры, относятся: экономическая целесообразность (для реализации подобной системы достаточно одного сервера); возможность получения интегральной информации (например, в виде отчетов) об изменениях, произведенных в нескольких источниках; организация централизованного долговременного хранения файлов архивных журналов и т. п.

СКИ для СУБД Oracle

На сегодняшний день Oracle, пожалуй, одна из немногих компаний, обеспечивающих разработчиков набором штатных инструментальных средств для решения задач мониторинга. Алгоритмы консолидации изменений, происходящих и транслируемых из исходных баз данных, основываются на механизме Oracle Streams версии 11g R2, предназначенном для распространения данных на базе очередей сообщений Oracle Advanced Queuing (AQ). Этот механизм достаточно универсален, с его помощью можно выполнять репликацию данных; резервное копирование; загрузку хранилищ данных из прикладных систем; управление событиями и т. п.

В СУБД Oracle используется журнал транзакций фиксированного размера, называемый также оперативным журналом повтора (online redo log). Рабочий объем журнала определяется при конфигурировании базы путем указания количества файлов журнала (не менее двух) и их размера. Выделение свободного места для записи генерируемого системой потока транзакций осуществляется по циклической схеме. Записи последовательно помещаются в один из файлов журнала (активный файл), по достижении конца которого производится переключение (logfile switching) на следующий по порядку файл либо на начало первого файла, если был заполнен последний из файлов журнала.

Для реализации возможности сохранения информации о транзакциях за продолжительный период времени, а также для передачи этой информации в другие системы необходимо включить режим ARCHIVELOG, активирующий механизм генерации архивных журналов (archived logs). При работе в таком режиме система осуществляет копирование заполненных файлов оперативного журнала в архивные журналы. Гарантируется, что до перезаписи файла система сделает копию его предыдущего содержимого.

В Oracle для любой базы может быть порождено несколько архивных журналов (не более 10) для решения различных прикладных задач. Каждый архивный журнал представляет собой папку в файловой системе, где сохраняются копии заполненных файлов оперативного журнала транзакций. Папки архивных журналов могут размещаться не только в локальной файловой системе, но и на удаленных устройствах. Обычно архивные журналы создаются для резервного копирования или для зеркального отображения данных на резервный сервер. Для передачи файлов журнала между основным и резервным серверами предназначен встроенный сервис Log Transport Service (процесс, осуществляющий транспортировку файлов журнала), который выполняет синхронизацию между серверами путем копирования журнальных файлов в определенную папку на резервном компьютере. Для подключения базы к Oracle Streams следует создать отдельный архивный журнал, который

будет передаваться на целевой сервер СКИ. Это рекомендуется сделать, чтобы избежать конфликтов со штатными процедурами архивирования, если таковые имеются.

При организации подключения базы к Oracle Streams возникает вопрос, связанный с выбором способа транспортировки файлов журнала на целевой сервер СКИ. Для этого можно использовать либо штатный сервис Log Transport Service, либо реализовать свое решение для передачи файлов.

Транспортировка архивных журналов

Опыт работы с Log Transport Service показал, что этот сервис имеет ряд принципиальных недостатков, ставящих под сомнение целесообразность его применения в качестве транспортного средства для передачи файлов журнала транзакций. Во-первых, метод аутентификации, используемый сервисом, требует совпадения паролей для учетной записи SYS на исходной и целевой базе данных. Данное требование необходимо при использовании сервиса транспортировки для организации зеркалирования, поскольку очевидно, что в этом случае вся учетная информация должна быть идентична. С другой стороны, при подключении к целевому серверу СКИ это ограничение нелогично и не отвечает требованиям безопасности.

Кроме того, Log Transport Service не обеспечивает гарантированной доставки файлов, в результате чего в журнале на целевом сервере могут возникать разрывы в последовательности отгруженных файлов. Это объясняется push-режимом, лежащим в основе архитектуры данного процесса. Если по каким-то причинам не удается передать файл (например, целевой компьютер выключен или с ним разорвано соединение), то процесс после нескольких неудачных попыток просто «забывает» про передаваемый файл и переключается на следующий. Такие разрывы приходится устранять вручную с помощью копий журнала, в которых следует отыскать все потерянные файлы и скопировать их на целевой сервер СКИ.

Можно рекомендовать альтернативный подход для организации транспортировки файлов журнала на целевой сервер СКИ, который основан на использовании pool-технологии. При подключении исходной базы к системе мониторинга в ней необходимо создать отдельный архивный журнал в папке файловой системы этого сервера. Эту папку следует сделать разделяемой и открыть к ней доступ по сети. На целевом сервере СКИ должен быть реализован и запущен процесс Log_File_Reader, который опрашивает удаленную папку и «забирает» из нее новые файлы. При подключении других внешних источников с архивными журналами каждый из этих журналов будет копироваться в свою папку на целевой сервер СКИ. Таким образом, любой архивный журнал исходной базы будет представлен своей копией, размещаемой в отдельном каталоге целевого сервера СКИ. После того как файл успешно передан, он удаляется из папки первоисточника. Разумеется, что для работоспособности описанной схемы у разделяемой папки-первоисточника должны быть установлены права на чтение/запись/удаление файлов для процесса Log_File_Reader. Изложенная схема по транспортировке файлов журнала обладает одним важным преимуществом — она устраняет возможность появления разрывов в последовательности загружаемых файлов.

Захват изменений

После того как решен вопрос с транспортировкой данных, требуется описать основные понятия и процессы Oracle Streams, которые позволяют выполнять обра-

ботку файлов журнала транзакций. Oracle Streams состоит из трех основных процессов:

- *Capture Process* — захват изменений в рабочих и архивных журналах транзакций, выбор из них записей об изменениях в исходной базе (Change Record, CR) и формирование логических записей изменений (Logical Change Record, LCR), помещаемых в очередь сообщений AQ;
- *Propagate Process* — передача сообщений из очереди исходной базы в очередь на целевой базе;
- *Apply Process* — применение изменений из очереди LCR к таблицам целевой базы либо их передача специальной обрабатывающей программе для выполнения необходимых преобразований.

В Oracle 11g R2 реализован комбинированный способ обработки журналов транзакций Combined Capture Apply, позволяющий через служебную очередь организовать взаимодействие между процессами Capture и Apply (рис. 2).

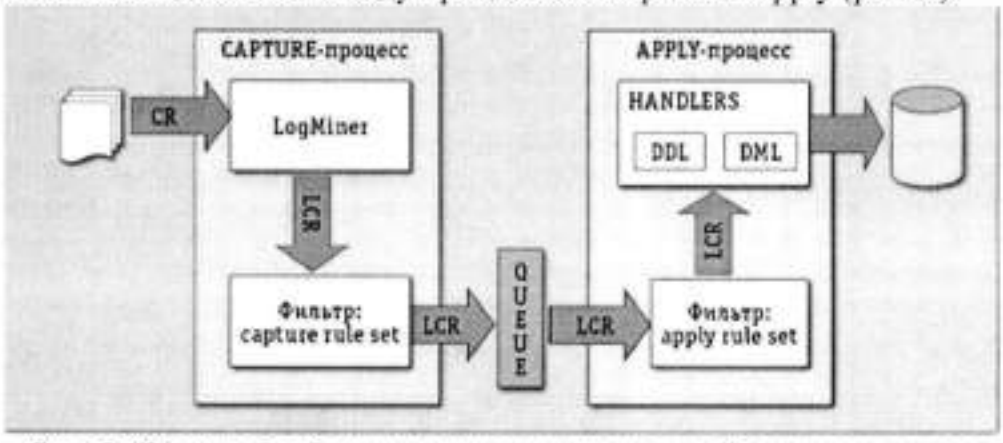


Рис. 2. Схема обработки журналов транзакций с помощью механизма Combined Capture Apply

Процесс захвата изменений осуществляет обработку поступающих log-файлов журнала при помощи вызова штатной утилиты Oracle LogMiner, которая считывает записи об изменениях (CR) и преобразует их в логические записи об изменениях — LCR. Для выполнения этой трансформации LogMiner использует словарь данных исходной базы, выгружаемый в журнал транзакций специальной процедурой (BUILD). На уровне процесса захвата изменений задаются правила, с помощью которых определяются схема и набор таблиц баз данных, подлежащих мониторингу. Соответственно, в очередь процесса Apply будут передаваться только изменения объектов, перечисленные в правилах. Набор правил может изменяться путем добавления в них новых объектов, подлежащих аудиту, или исключения тех, мониторинг которых утратил актуальность. Далее процесс Apply извлекает из очереди входящих сообщений LCR-записи и передает их обрабатывающей прикладной программе. Предварительно выбираемые записи могут быть отфильтрованы по аналогичным правилам, которые задаются для процесса захвата изменений. Обрабатывающая программа разбирает LCR, распознавая указанный там тип операции, и сохраняет ее в базе данных СКИ.

На наш взгляд, главным ограничением существующей версии Streams является отсутствие штатного доступа к словарю данных, который был получен из журнала транзакций. Из-за этого при построении пользовательских сервисов над базой СКИ

приходится обращаться к недокументированным системным таблицам для выбора объектов схемы (названий таблиц, полей и т. п.) из списка.

Управление мониторингом

Описанная технология позволяет реализовать на сервере СКИ сервис, предоставляющий конечным пользователям возможность самостоятельно управлять процессом аудита путем явного указания объектов, подлежащих мониторингу. На основании подключенных к аудиту баз данных пользователи сервиса смогут определять состав таблиц, для которых будут протоколироваться операции по изменению данных и схемы. С этой целью каждый аудитор создает свой профиль, содержащий набор правил, которые определяют интересующие его изменения. Система объединяет правила, указанные во всех профилях, формируя интегральный фильтр, который будет применяться для отбора данных из архивных журналов при захвате изменений.

При задании конкретного правила в своем профиле пользователь может дополнительно указать дату, начиная с которой он хочет получить соответствующие изменения, произведенные в базе прикладной системы. Указание такой даты «инструктирует» систему о необходимости перезагрузки файлов журнала задним числом, но не ранее момента подключения базы к мониторингу. Это возможно, если считать, что все архивные журналы-первоисточники сохранены в файловой системе СКИ.

Интересы аудиторов могут меняться в процессе эксплуатации системы мониторинга — в нее могут добавляться новые правила или удаляться существующие. Система должна отслеживать эти модификации, перестраивая интегральный фильтр, и если оказывается, что часть сохраненных в СКИ данных больше не востребована, они удаляются из базы. Кроме того, при определении любого правила можно указать количество дней до текущей даты, в течение которых будет храниться отобранная по данному правилу информация в базе СКИ.

Роль администратора при такой организации сводится лишь к подключению исходных баз к системе мониторинга, а все задачи управления номенклатурой загружаемой в СКИ информации полностью возлагаются на пользователей-аудиторов, поскольку именно они обладают знанием предметной области и могут точно определить, какие изменения необходимо контролировать.

***

Описанный подход к организации мониторинга изменений баз данных позволяет создать корпоративный сервис централизованного аудита всех критических изменений в прикладных системах компании или организации как на уровне данных, так и их структур. Такой аудит особенно важен в случае взаимодействия большого количества прикладных систем, в которые вносятся изменения, способные оказать влияние на логическую целостность и информационную безопасность. Успешное тестирование и апробация изложенных механизмов были проведены в рамках пилотного проекта в компании ОАО «АТС» — коммерческого оператора оптового рынка электроэнергии в России. Одной из побудительных причин для данного проекта стала возникшая в компании потребность в создании единого универсального механизма аудита на уровне СУБД действий пользователей, программ-загрузчиков, SQL-скриптов и т. п., которые затрагивают изменения в схеме базы и самих данных. Программный комплекс состоит из нескольких сотен взаимодействующих модулей, в которые постоянно вносятся изменения в соответствии с изменениями правил рынка. Подобный механизм аудита дает мощное средство обеспечения ло-

гической целостности и информационной безопасности всего программного обеспечения. Представленное в статье решение по организации аудита на основе журнала транзакций универсально и может быть реализовано на любой платформе СУБД во многих крупных компаниях и организациях.

3. ОБЪЕКТЫ И СРЕДСТВА ИССЛЕДОВАНИЯ

Предметная область в соответствии с вариантом задания. ПК типа IBM PC, дисплейный класс, операционная система Windows XP и выше.

4. ПОДГОТОВКА К РАБОТЕ

Ознакомиться с теоретическими положениями лабораторной работы.

5. ПРОГРАММА РАБОТЫ

Провести анализ методов аудита в СУБД. Оформить отчет.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

6.1. Назовите основные подходы к аудиту баз данных?
6.2. Основные методы аудита?

Лабораторная работа № 15

РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ БАЗ ДАННЫХ

1. ЦЕЛЬ РАБОТЫ

Изучение методов резервного копирования и восстановления баз данных.

2. ОСНОВЫ ТЕОРИИ

Модели восстановления

Перед тем как браться за настройку резервного копирования, следует выбрать модель восстановления. Для оптимального выбора следует оценить требования к восстановлению и критичность потери данных, сопоставив их с накладными расходами на реализацию той или иной модели.

Как известно, база данных MS SQL состоит из двух частей: собственно, базы данных и лога транзакций к ней. База данных содержит пользовательские и служебные данные на текущий момент времени, лог транзакций включает в себя историю всех изменений базы данных за определенный период, располагая логом транзакций мы можем откатить состояние базы на любой произвольный момент времени.

Для использования в производственных средах предлагается две модели восстановления: простая и полная. Существует также модель с неполным протоколированием, но она рекомендуется только как дополнение к полной модели на период крупномасштабных массовых операций, когда нет необходимости восстановления базы на определенный момент времени.

Простая модель предусматривает резервное копирование только базы данных, соответственно восстановить состояние БД мы можем только на момент создания резервной копии, все изменения в промежуток времени между созданием последней резервной копии и сбоем будут потеряны. В тоже время простая схема имеет небольшие накладные расходы: вам необходимо хранить только копии базы данных, лог транзакций при этом автоматически усекается и не растет в размерах. Также процесс восстановления наиболее прост и не занимает много времени.

Полная модель позволяет восстановить базу на любой произвольный момент времени, но требует, кроме резервных копий базы, хранить копии лога транзакций за весь период, для которого может потребоваться восстановление. При активной работе с базой размер лога транзакций, а, следовательно, и размер архивов, могут достигать больших размеров. Процесс восстановления также гораздо более сложен и продолжителен по времени.

При выборе модели восстановления следует сравнить затраты на восстановление с затратами на хранение резервных копий, также следует принять во внимание наличие и квалификацию персонала, который будет выполнять восстановление. Восстановление при полной модели требует от персонала

определенной квалификации и знаний, тогда как при простой схеме достаточно будет следовать инструкции.
Для баз с небольшим объемом добавления информации может быть выгоднее использовать простую модель с большой частотой копий, которая позволит быстро восстановиться и продолжить работу, введя потерянные данные вручную. Полная модель в первую очередь должна использоваться там, где потеря данных недопустима, а их возможное восстановление сопряжено со значительными затратами.
Виды резервных копий
Полная копия базы данных - как следует из ее названия, представляет собой содержимое базы данных и часть активного лога транзакций за то время, которое формировалась резервная копия (т.е. сведения обо всех текущих и незавершенных транзакциях). Позволяет полностью восстановить базу данных на момент создания резервной копии.
Разностная копия базы данных - полная копия имеет один существенный недостаток, она содержит всю информацию базы данных. Если резервные копии нужно делать довольно часто, то сразу возникает вопрос неэкономного использования дискового пространства, так как большую часть хранилища будут занимать одинаковые данные. Для устранения этого недостатка можно использовать разностные копии базы данных, которые содержат только изменившуюся со времени последнего полного копирования информацию.
Обращаем внимание, разностная копия - это данные от момента последнего полного копирования, т.е. каждая последующая разностная копия содержит в себе данные предыдущей (но при этом они могут быть изменены) и размер копии будет постоянно расти. Для восстановления достаточно одной полной и одной разностной копии, обычно последней. Количество разностных копий следует выбирать исходя из прироста их размера, как только размер разностной копии сравнится с размером половины полной, имеет смысл сделать новую полную копию.
Резервная копия журнала транзакций - применяется только при полной модели восстановления и содержит копию журнала транзакций начиная с момента создания предыдущей копии.

Важно помнить следующий момент - копии журнала транзакций никак не связаны с копиями базы данных и не содержат информацию предыдущих копий, поэтому для восстановления базы вам необходимо иметь непрерывную цепочку копий того периода, в течении которого вы хотите иметь возможность откатывать состояние базы. При этом момент последнего успешного копирования должен быть внутри этого периода.
Посмотрим на рисунок выше, если будет утрачена первая копия файла журнала, то вы сможете восстановить состояние базы только на момент полного копирования, что будет аналогично простой модели восстановления, восстановить состояние базы на любой момент времени вы сможете только после следующего разностного (или полного) копирования, при условии, что це-

почка копий журналов начиная с предшествующего копированию базы и далее будет непрерывна (на рисунке - от третьего и далее).

Журнал транзакций

Для понимания процессов восстановления и назначения разных видов резервных копий следует более подробно рассмотреть устройство и работу журнала транзакций. Транзакция - это минимально возможная логическая операция, которая имеет смысл и может быть выполнена только полностью. Такой подход обеспечивает целостность и непротиворечивость данных при любых ситуациях, так как промежуточное состояние операции недопустимо. Для контроля над любыми изменениями в базе предназначен журнал транзакций.

При выполнении любой операции в журнал транзакций добавляется запись о начале транзакции, каждой записи присваивается уникальный номер (LSN) из неразрывной последовательности, при любом изменении данных в журнал вносится соответствующая запись, а после завершения операции в журнале появляется отметка о закрытии (фиксации) транзакции.

При каждом запуске система анализирует журнал транзакций и откатывает все незафиксированные транзакции, одновременно с этим происходит накат изменений, которые зафиксированы в журнале, но не были записаны на диск. Это дает возможность использовать кэширование и отложенную запись, не опасаясь за целостность данных даже при отсутствии систем резервного питания.

Та часть журнала, которая содержит активные транзакции и используется для восстановления данных называется активной частью журнала. Она начинается с номера, который называется минимальным номером восстановления (MinLSN).

В простейшем случае MinLSN - это номер записи первой незавершенной транзакции. Если посмотреть на рисунок выше, то открыв синюю транзакцию мы получим MinLSN равную 321, после ее фиксации в записи 324, номер MinLSN изменится на 323, что будет соответствовать номеру зеленой, еще не зафиксированной, транзакции.

На практике все немного сложнее, например, данные закрытой синей транзакции могут быть еще не сброшены на диск и перемещение MinLSN на 323 сделает восстановление этой операции невозможной. Для того, чтобы избежать таких ситуации было введено понятие контрольной точки. Контрольная точка создается автоматически при наступлении следующих условий:

При явном выполнении инструкции CHECKPOINT. Контрольная точка срабатывает в текущей базе данных соединения.

При выполнении в базе данных операции с минимальной регистрацией, например, при выполнении операции массового копирования для базы данных, на которую распространяется модель восстановления с неполным протоколированием.

При добавлении или удалении файлов баз данных с использованием инструкции ALTER DATABASE.

При остановке экземпляра SQL Server с помощью инструкции SHUTDOWN или при остановке службы SQL Server (MSSQLSERVER). И в том, и в другом случае будет создана контрольная точка каждой базы данных в экземпляре SQL Server.
Если экземпляр SQL Server периодически создает в каждой базе данных автоматические контрольные точки для сокращения времени восстановления базы данных.
При создании резервной копии базы данных.
При выполнении действия, требующего отключения базы данных. Примерами могут служить присвоение параметру AUTO_CLOSE значения ON и закрытие последнего соединения пользователя с базой данных или изменение параметра базы данных, требующее перезапуска базы данных.
В зависимости от того, какое событие произошло раньше, MinLSN будет присвоено значение либо номера записи контрольной точки, либо начала самой старой незавершенной транзакции.
Усечение журнала транзакций
Журнал транзакций, как и любой журнал, требует периодической очистки от устаревших записей, иначе он разрастется и займет все доступное место. Учитывая, что при активной работе с базой размер лога транзакций может значительно превышать размер базы, то этот вопрос актуален для многих администраторов.
Физически файл журнала транзакций является контейнером для виртуальных журналов, которые последовательно заполняются по мере роста лога. Логический журнал, содержащий запись MinLSN является началом активного журнала, предшествующие ему логические журналы являются неактивными и не требуются для автоматического восстановления базы.
Если выбрана простая модель восстановления, то при достижении логическими журналами размера равного 70% физического файла происходит автоматическая очистка неактивной части журнала, т.н. усечение. Однако это не приводит к уменьшению физического файла журнала, усекаются только логические журналы, которые после этой операции могут использоваться повторно.

Если количество транзакций велико и к моменту достижения 70% размера физического файла не окажется неактивных логических журналов, то размер физического файла будет увеличен.
Таким образом файл лога транзакций при простой модели восстановления будет расти согласно активности работы с базой до тех пор, пока не будет надежно вмещать всю активную часть журнала. После чего его рост прекратится.
При полной модели неактивную часть журнала нельзя усечь до тех пор, пока она полностью не попадет в резервную копию. Усечение журнала производится при условии, что выполнена резервная копия журнала транзакций, после чего была создана контрольная точка.

Неправильная настройка резервного копирования журнала транзакций при полной модели способно привести к неконтролируемому росту файла журнала, что часто составляет проблему для неопытных администраторов. Также часто попадаются советы по ручному усечению журнала транзакций. При полной модели восстановления делать этого не следует категорически, так как тем самым вы нарушите целостность цепочки копий журнала и сможете восстановить базу только на момент создания копий, что будет соответствовать простой модели.

В этом случае самое время вспомнить то, о чем мы говорили в начале статьи, если затраты на полную модель превышают затраты на восстановление следует отдать предпочтение простой модели.

Простая модель восстановления

Теперь, после получения необходимого минимума знаний, можно перейти к более подробному рассмотрению моделей восстановления. Начнем с простой. Допустим, на момент сбоя у нас имеется одна полная и две разностные копии:

Резервное копирование выполнялось раз в сутки и последняя копия была создана ночью с 21-го на 22-е. Сбой происходит вечером 22-го до создания очередной копии. В этом случае нам потребуется последовательно восстановить полную и последнюю разностные копии, при этом данные за последний рабочий день будут утеряны. Если по каким-либо причинам копия от 21-го также окажется повреждена, то мы можем восстановить предыдущую копию, потеряв еще день работы, в тоже время повреждение копии за 20-е число никак не помешает успешно восстановить данные на вечер 21-го, при наличии соответствующей копии.

Полная модель восстановления

Рассмотрим аналогичную ситуацию, но с применением полной модели восстановления. Резервные копии у нас также делаются ежесуточно, по принципу полная + разностные, а также несколько раз в сутки копируется лог транзакций.

Процесс восстановления в этом случае будет более сложен. Прежде всего потребуется создать вручную резервную копию заключительного фрагмента журнала (показана красным), т.е. часть журнала с момента прошлого создания копии и до аварии.

Если этого не сделать, то восстановить базу можно будет только до состояния на момент создания последней копии журнала транзакций.

При этом повреждение файла копии журнала за предыдущий день не помешает нам восстановить актуальное состояние базы, но ограничит нас моментом создания последней копии, т.е. текущими сутками.

Затем последовательно восстанавливаем полную и разностную копию и цепочку копий журнала, созданную после последнего резервного копирования, последней восстанавливаем копию заключительного фрагмента журнала, что даст нам возможность восстановить базу прямо на момент аварии или произвольный, предшествовавший ему.

Если последняя разностная копия будет повреждена, то в случае с простой моделью это приведет к потере еще одного рабочего дня, полная модель позволяет восстановить предпоследнюю копию, после чего нужно будет восстановить всю цепочку копий лога транзакций от момента предпоследней копии и до сбоя. Глубина восстановления зависит только от глубины непрерывной цепочки логов.

С другой стороны, если одна из копий лога транзакций будет повреждена, скажем, предпоследняя, то восстановить данные мы сможем только на момент последней резервной копии + период в неповрежденной цепочке копий журналов. Например, если журналы делались в 12, 14 и 16 часов и поврежден журнал, созданный в 14 часов, то располагая суточной копией мы сможем восстановить базу до момента окончания непрерывной цепочки, т.е. до 12 часов.

3. ОБЪЕКТЫ И СРЕДСТВА ИССЛЕДОВАНИЯ

Предметная область в соответствии с вариантом задания. ПК типа IBM PC, дисплейный класс, операционная система Windows XP и выше.

4. ПОДГОТОВКА К РАБОТЕ

Ознакомиться с теоретическими положениями лабораторной работы.

5. ПРОГРАММА РАБОТЫ

Изучить модели копирования и восстановления баз данных. Оформить отчет.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

6.1. Назовите основные подходы к копированию и восстановлению баз данных?

6.2. Основные методы восстановления баз данных?

ПРИЛОЖЕНИЕ

ВАРИАНТЫ ПРИМЕРНЫХ ПРЕДМЕТНЫХ ОБЛАСТЕЙ ДЛЯ ВЫДАЧИ ЗАДАНИЙ ЛАБОРАТОРНЫХ РАБОТ ПО БД

ЗАДАНИЕ 1:
ИНФОРМАЦИОННАЯ СИСТЕМА "ДЕТСКАЯ ПОЛИКЛИНИКА"

Система хранит информацию о врачах, пациентах, заболеваниях, детских учреждениях. Информационная система должна предоставлять данные по отдельным запросам:

- информацию о враче (фамилия, специализация, стаж, оклад, совместительство);
- данные о больном (фамилия, возраст, адрес, детское учреждение, место работы родителей, хронические заболевания, прививки, последнее обращение к врачу);
- данные о детском учреждении (наименование, адрес, количество детей, наличие карантина, выявленные инфекционные заболевания, дата последнего профилактического обследования).

Кроме того периодически должны выдаваться следующие ведомости:

- список детских учреждений, в которых зафиксированы инфекционные заболевания;
- статистический отчет по заболеваниям;
- отчет по заболеваемости в детских учебных заведениях.

ЗАДАНИЕ 2:
ИНФОРМАЦИОННАЯ СИСТЕМА "УПРАВЛЕНИЕ ДОРОЖНЫМ ДВИЖЕНИЕМ"

Система хранит информацию об автомобилях, зарегистрированных в области, владельцах автомашин, о нарушениях правил дорожного движения. Системой представляется следующая информация по отдельным запросам:

- информация об автомобиле (регистрационный номер, марка, номер двигателя, дата последнего техосмотра, владелец);
- информация о владельце (номер и марка автомобиля, количество предупреждений в талоне, дата медицинского обследования, фамилия, адрес);
- информация о водителях некоторого автохозяйства.

Периодически и по запросу должны выдаваться следующие ведомости:

- список нарушителей за указанный период;
- список водителей автохозяйств, направленных на медицинскую комиссию;

- статистика нарушений по типам, маркам автомобилей, по автохозяйствам, владельцам.

ЗАДАНИЕ 3:
ИНФОРМАЦИОННАЯ СИСТЕМА "ТЕЛЕАТЕЛЬЕ"

Хранит информацию о типах телевизоров, комплектующих деталях, обслуживающих бригадах, о качестве ремонта. В ателье реализована бригадная форма работы: задание выдается на бригаду, каждая бригада отдельно обеспечивается деталями.

Отдельные запросы, реализуемые системой, могут касаться:

- конкретного телевизора (номер приемной квитанции, тип телевизора, адрес владельца, дата приемки, срок исполнения заказа);
- комплектующей детали (название, маркировка, возможная замена, количество деталей на складе);
- работа отдельной бригады (фамилия бригадира, количество членов бригады, количество выполненных заказов, выполнение плана в стоимостном выражении, рекламация и возврат).

Система должна также обеспечить выдачу следующих выходных документов:

- отчет о выполнении планового задания; состояние склада телеателье;
список заказов с истекшим сроком исполнения.

ЗАДАНИЕ 4:
ИНФОРМАЦИОННАЯ СИСТЕМА "ЗООПАРК"

Хранит информацию о животных, занимаемых ими помещениях, о рационе животных, об обслуживаемом персонале. Система должна представлять информацию по следующим запросам:

- информацию о конкретном животном (кличка, возраст, вид, размещение, состояние здоровья);
- данные о работниках зоопарка (фамилия, возраст, специальность, отделение, стаж, адрес);

Кроме того периодически должны выдаваться следующие документы:

- рекомендуемый рацион для всех животных;
- список заболевших животных;
- список животных с адресами по видам;
- состояние животных, обслуживаемых отдельными работниками;
- ведомость заполнения помещений зоопарка.

ЗАДАНИЕ 5:
ИНФОРМАЦИОННАЯ СИСТЕМА "АБИТУРИЕНТ"

Создается информационная система АБИТУРИЕНТ для автоматизации работы приемной комиссии вуза. Один из фрагментов этой предметной области требует обработки анкетных данных абитуриентов. Для этого фрагмента разработать программу :

- меню и средства диалога; ввода и изменения данных; подготовки печатных форм.

Анкета включает следующие данные об абитуриенте :

- регистрационный номер; фамилия, имя, отчество; дата рождения;
- окончание среднего учебного заведения; дата окончания;
- наличие красного диплома или золотой медали;
- адрес (город, улица, N дома, квартиры, телефон);
- выбранная специальность.

Исходными документами для заполнения анкеты является аттестат или диплом о среднем образовании, заявление абитуриента. В вузе определен список специальностей, который может изменяться ежегодно. По каждой специальности вуза определен список сдаваемых предметов, например:

- математика (П), - математика (У), - родной язык (П), и т.д.

ИС должна обеспечивать выполнение следующих функций :

- ввод и коррекцию анкетных данных;
- просмотр анкетных данных по специальностям в алфавитном порядке;
- ввод, коррекцию и просмотр специальностей и сдаваемых предметов;
- вывод на печать анкетных данных абитуриентов, имеющих красный диплом или медаль;
- вывод на печать всех инициалов абитуриентов по специальностям в алфавитном порядке с указанием сдаваемых предметов;
- вывод на печать анкетных данных по специальностям.

ЗАДАНИЕ 6:
ИНФОРМАЦИОННАЯ СИСТЕМА "НАЧИСЛЕНИЯ ЗАРПЛАТЫ

Создается информационная подсистема НАЧИСЛЕНИЯ ЗАРПЛАТЫ для автоматизации начисления заработной платы в бухгалтерии.

Зарплата начисляется работникам вуза, имеющим установленные оклады (сдельных работ нет). На каждого работника хранятся следующие данные: личный номер; ф.и.о.; должность; оклад; семейное положение и число детей; данные о невыходе на работу по болезни (даты заболевания и выздоровления) и т.д.

В период болезни работнику начисляется 50 % зарплаты; 100 % начисляется лишь членам профсоюза. Работникам могут начисляться премии и другие надбавки. С общей суммы зарплаты снимается подоходный налог 8 % и налог за бездетность 6 %.

ИС должна обеспечивать:

- ввод, изменение анкетных данных работников, сведения о болезнях, надбавках; ежемесячный перерасчет зарплаты с выдачей ведомости на экран и на печать.

Разработать программы:

меню и выдачи печатных форм; ввода и изменения данных.

ЗАДАНИЕ 7:
КОНТРОЛЬ ИСПОЛНЕНИЯ ПОРУЧЕНИЙ

Создается информационная подсистема КОНТРОЛЬ ИСПОЛНЕНИЯ ПОРУЧЕНИЙ для некоторой организации. В качестве исходной информации используются данные:

- порядковый номер поручения, название поручения, содержание поручения, дата выдачи поручения, срок исполнения, дата фактического исполнения, исполнитель, кто выдал поручение.

Поручения могут выдавать руководитель организации и руководители подразделений. Ввод всех данных в персональную ЭВМ выполняет один оператор.

ИС должна обеспечивать:

- ввод и коррекцию данных о поручениях, просмотр поручений по некоторой дате, ежедневную печать поручений с текущей датой исполнения (для руководителя организации).

Разработать программы:

меню и выдачи печатных форм, ввода, изменения и просмотра данных.

ЗАДАНИЕ 8
ИНФОРМАЦИОННАЯ СИСТЕМА "СНАБЖЕНИЕ"

Информационная система создается для оптовой базы. Основным назначением оптовой базы является снабжение сети магазинов различными товарами. Отдел снабжения в каждый момент времени должен иметь точные данные о названии товаров, их количестве на складе, о названии магазинов, о названии и количестве каждого вида товара в каждом магазине, о заявках магазинов на текущий год.

Отдел снабжения должен иметь возможность проделывать следующие операции:

- включить новый товар в список товаров на складе; удалить ненужный товар из складского списка; включить новый магазин в список магазинов; удалить ненужный магазин из списка; выполнить поступление некоторого товара на склад; просмотреть информацию о товарах на складе; просмотреть информацию о товарах по магазинам; провести инвентаризацию склада и каждого магазина; выдать магазину товар со склада и отпечатать накладную; ввести заявку магазина на текущий год.

ЗАДАНИЕ 9:
ИНФОРМАЦИОННАЯ СИСТЕМА "ДЕКАНАТ"

Создается информационная система ДЕКАНАТ для автоматизации работы деканата факультета вуза. Пусть максимальное число кафедр на факультете равно 10, кафедры готовят студентов по 12 специальностям.

По каждой специальности имеется учебный план, который содержит список всех предметов, изучаемых на этой специальности, с указанием общего количества лекционных, практических, лабораторных часов, раскладку предметов и курсовых работ по семестрам с указанием количества часов, форму сдачи предмета (зачет/экзамен).

По окончании вступительных экзаменов приёмная комиссия вуза передает в деканат личные дела и списки зачисленных абитуриентов. На каждого студента 1-го курса заводится учебная карточка, в которую заносятся его точные данные, а также список предметов, подлежащих сдаче согласно учебному плану специальности. По мере сдачи предметов и перехода с курса на курс учебная карточка заполняется соответствующими оценками.

По окончании вуза копия учебной карточки выдаётся студенту как приложение к диплому. Учебная карточка выдаётся также при переводе в другой вуз.

Система должна обеспечивать ввод и обработку учебных планов специальностей, учебных карточек студентов, выдачу списков студентов по различным выборкам, должна исключать дублирование данных о предметах. Пользователями ИС являются декан и секретарь деканата. Систему можно расширить, включив обработку данных о преподавателях, их нагрузке. Система ДЕКАНАТ может включаться в ИС всего вуза. Каждое подразделение, кафедра, учебный отдел, научная часть, бухгалтерия, от дел кадров и пр. могут иметь собственные информационные под системы.

ЗАДАНИЕ 10:
ИНФОРМАЦИОННАЯ СИСТЕМА "КАДРЫ"

Разработать ИС КАДРЫ для автоматизации работы отдела кадров предприятия с числом сотрудников до 1000 человек. Система должна функционировать в двух режимах: первичной загрузки данных и текущей обработки информации.

В режиме загрузки базы данных система должна предоставлять ввод данных из личных карточек работающих с контролем вводимой информации. В режиме текущей обработки система должна реализовывать функции:

- обработку данных по движению кадров: прием, увольнение, перемещение;
- получение статистической отчетной информации по уволен ным и работающим в различных разрезах;

- получение справочной информации по данным, содержащимся в личной картотеке;

- ведение табельного учета по отсутствующим на местах.

ЗАДАНИЕ 11:
ИНФОРМАЦИОННАЯ СИСТЕМА "ЗАРПЛАТА"

Разработать ИС "Зарплата" для автоматизации учета труда и заработной платы.

Система должна предоставлять следующие функциональные возможности:

а) начисление аванса за первую половину месяца; основной заработной платы рабочим-повременщикам и распределение премии с учетом коэффициента трудового участия КТУ; основной заработной платы по установленному окладу или тарифу; различного рода премий, доплат, надбавок к основной заработной плате;

б) расчет сдельного заработка рабочим-сдельщикам, работающим по бригадному подряду и индивидуально; пенсии работающим пенсионерам;

в) исчисление налогов;

г) удержание по исполнительным листам; за товары, купленные в кредит; различных ссуд Госбанка; профсоюзных взносов;

д) перечисление сумм организациям Госстраха; в сберегательный банк;

е) формирование расчетно-платежной документации.

ЗАДАНИЕ 12
ИНФОРМАЦИОННАЯ СИСТЕМА "КОММЕРЦИЯ"

В вашем городе создается компьютерный центр коммерческой информации. Его функцией является сбор сведений о предприятиях, фирмах, кооперативах и пр., о производимых ими товарах и услугах, систематизация этих данных по различным параметрам, издание ежеквартальных бюллетеней о сведениях, зарегистрированных за прошедший квартал, выдача интересующей информации по заказу отдельных лиц и организаций.

Информация собирается из периодической печати (рекламные объявления в газетах и журналах),а также может предоставляться самой регистрируемой организацией. Еженедельная порция вводимых данных составляет до 10 новых организаций и до 100 организаций, в которых обновляется информация. Хранимая в базе данных информация об организации должна включать следующие сведения: точное название организации; страна, город и точный адрес, телефон, телекс, телефакс; основные виды деятельности или отрасли производства; вид или наименование производимых товаров или услуг.

Оперативная информация: что приобретается, продается, кто требуется на работу и пр. Данные в базе данных не должны дублироваться.

ИС должна обеспечивать выборку информации по различным критериям, например: "выдать список фирм, занимающихся производством бытовой электроники", далее "в этом списке выбрать список фирм, производящих электронные часы" и т.д.

ЗАДАНИЕ 13
ИНФОРМАЦИОННАЯ СИСТЕМА "КОМИССИОННЫЙ МАГАЗИН"

Информационная система создается для автоматизации работы комиссионного магазина. Основным назначением системы является учет товарооборота магазина.

Система должна предоставлять по запросу выдачу данных о названии товаров, их количестве, о товарах, подлежащих переоценке в связи с истечением срока и обеспечивать выполнение следующие операции: включить новый товар в список товаров; удалить реализованный товар из списка; просмотреть информацию о товарах по магазину; провести инвентаризацию магазина; выдать ведомость товаров сданных в магазин на определенную дату.

ЗАДАНИЕ 14
ИНФОРМАЦИОННАЯ СИСТЕМА "АВТОСЕРВИС"

Создается система для хранения и выдачи информации об автомобилях, находящихся на обслуживании, владельцах автомобилей, неисправностях и о наличии деталей на складе.

Информация об автомобиле должна содержать данные: номер квитанции, марку автомобиля, неисправность, детали, необходимые для ремонта. Информация о владельцах должна содержать следующие данные: Ф.И.О., номер квитанции, марку автомобиля.

Информация о запчастях должна содержать данные: код детали, название, стоимость, расценка ее замены.

Периодически по запросу должны выдаваться следующие сведения:

- перечень Ф.И.О. клиентов с указанием марки автомобиля, вида неисправности, номера накладной и перечня используемых деталей. Запрос о наличии определенной детали на складе. Результаты инвентаризации склада.

ЗАДАНИЕ 15
ИНФОРМАЦИОННАЯ СИСТЕМА "УНИВЕРМАГ"

Информационная система хранит информацию о товарах, поставщиках товаров, отделах универмага и сотрудниках, работающих в универмаге.

Система должна представлять информацию по следующим запросам:

- информация о поставщиках товаров,
- информация о товарах.

Система должна обеспечивать выдачу следующих выходных документов:

- список фирм, поставляющих товар; список товаров, находящихся на реализации; список цен на товары; список стран-производителей товаров; адреса фирм-поставщиков.

ЗАДАНИЕ 16
ИНФОРМАЦИОННАЯ СИСТЕМА "АВТОХОЗЯЙСТВО"

Система хранит информацию об автомобилях, зарегистрированных в данном автохозяйстве, шоферах автомашин, о нарушениях правил дорожного движения. Системой представляется следующая информация по отдельным запросам:

- информация об автомобиле (регистрационный номер, марка, номер двигателя, дата последнего техосмотра, владелец); информация о шофере автохозяйства (номер и марка автомобиля, количество предупреждений в талоне, дата медицинского обследования, фамилия, адрес);

Периодически и по запросу должны выдаваться следующие ведомости:

- список нарушителей за указанный период; список водителей автохозяйства, направленных на медицинскую комиссию; статистика нарушений по типам, маркам автомобилей.

ЗАДАНИЕ 17
ИНФОРМАЦИОННАЯ СИСТЕМА "ДЕТСКАЯ БОЛЬНИЦА"

Система хранит информацию о врачах, пациентах, заболеваниях, детских учреждениях. Информационная система должна предоставлять данные по отдельным запросам:

информацию о враче (фамилия, специализация, стаж, оклад, совместительство); данные о больном (фамилия, возраст, адрес, детское учреждение, место работы родителей, хронические заболевания, прививки, последнее обращение к врачу); информацию о лекарствах для приобретения их в аптеке.

Кроме того периодически должны выдаваться следующие ведомости:

- список больных детских учреждений, в которых зафиксированы инфекционные заболевания; перечень дефицитных лекарств, отсутствующих в больнице.

ЗАДАНИЕ 18
ИНФОРМАЦИОННАЯ СИСТЕМА «АЭРОПОРТ»

Создается система для хранения и выдачи информации о самолетах, о расписании самолетов, о сотрудниках аэропорта.

Периодически по запросу должны выдаваться следующие сведения:

- информацию о самолетах (тип самолета, номер самолета, номер обслуживающей самолет бригады);
- информацию о расписании (номер рейса, пункт отправления, пункт назначения, время отправления, время полета, стоимость билета),
- Информацию о летном составе (Ф.И.О., должность, стаж работы, адрес, оклад).

ЗАДАНИЕ 19
ИНФОРМАЦИОННАЯ СИСТЕМА "СКЛАД"

Информационная система создается для объекта склад.

Периодически по запросу должны выдаваться данные о названии товаров, их количестве на складе.

Оператор ЭВМ должен иметь возможность проделывать следующие операции обработки данных:

- включение нового товара в перечень товаров на складе;
- удаление ненужный товар из списка;
- просмотреть информацию о наличии товаров;

а также выдачу документов: инвентаризации склада; накладную на получение товаров со склада; ведомость пересортировки товаров.

ЗАДАНИЕ 20
ИНФОРМАЦИОННАЯ СИСТЕМА "СПОРТ"

Создается информационная система СПОРТ для автоматизации обработки данных о проводимых соревнованиях и чемпионатах.

Перечень информационных требований к системе:

- выдача данных о результатах чемпионата по определенному виду спорта с указанием страны, занятого места и количества набранных баллов,
- выдача информации о спортсменах-победителях чемпионата,
- перечень чемпионатов, проводимых в стране с указанием места, времени, вида спорта, стран участниц

ЗАДАНИЕ 21
ИНФОРМАЦИОННАЯ СИСТЕМА "АВТОБУСНОЕ ДВИЖЕНИЕ"

Информационная система создается для автоматизации процесса обработки и хранения данных о проверке состояния машин и возможности ответов на запросы, представленные следующим набором требований:

- ведомость контроля за состоянием машин, закрепленных за бригадой;
- расписание рейсов автобусов с указанием направления движения и временем отправления,
- расписание отправления автобусов из АТП (для диспетчера);
- ведомость выполнения расписания движения.

Периодически и по запросу должны выдаваться следующие ведомости:

- данных о водителях бригады, с указанием стажа работы, количества нарушений, виды поощрения;
- данные об авариях и невыхода транспорта с указанием причины.

ЗАДАНИЕ 22
ИНФОРМАЦИОННАЯ СИСТЕМА "ЖЕЛЕЗНОДОРОЖНЫЙ ТРАНСПОРТ"

Информационная система создается для автоматизации процесса обработки и хранения оперативных данных и справочной информации о работе железнодорожного транспорта. Она должна предусматривать ответы на некоторые запросы и должна отвечать следующим требованиям:

- оперативный ввод информации о заявках на перевозку грузов и пассажиров;
- печать графика движения железнодорожных составов пассажирских и грузовых) исходя из пропускной способности транспортного узла.
- печать графика работы железнодорожных бригад (машинистов), учитывая болезни, отпуска, выполнения обязанностей и т.д.)
- расчет графика профилактики , ремонта подвижного состава, транспортных путей;
- выдавать ведомость о перевозках пассажиров по дням, месяцам;
- выдавать ведомость состава обслуживающего персонала локомотивных бригад.

Периодически и по запросу предусмотреть выдачу оперативных данных:

- о нарушении графика движения, с указанием причин и виновных,
- о наличии неиспользуемого транспортного парка.

ЗАДАНИЕ 23
ИНФОРМАЦИОННАЯ СИСТЕМА "ВОЕНКОМАТ"

Создать информационную систему для автоматизации процесса обработки и хранения данных о результатах прохождения призывной комиссии и

возможности ответов на некоторые вопросы, представленные следующими требованиями:

- распечатка учетно-послужных карточек призывников, содержащих анкетные данные о призывнике,
- распечатка медицинских ведомостей призывников, с указанием состава врачей приемной комиссии и результатах пригодности призывника к военной службе;
- распечатка ведомостей призывной медицинской комиссии с указанием ФИО призывника, кода учетно-призывной карточки и рода войск;
- запрос данных о составе медицинской комиссии с указанием специальностей врачей, номера больницы и других анкетных данных;
- запрос данных о распределении призывников по родам войск с указание количества призывников в разрезе по роду войск.

ЗАДАНИЕ 24
ИНФОРМАЦИОННАЯ СИСТЕМА "КНИГОХРАНИЛИЩЕ"

Создать информационную систему для автоматизации процесса обработки и хранения данных об имеющихся в данном книгохранилище книгах и возможности ответов на некоторые запросы, представленные следующим перечнем информационных требований:

- результаты инвентаризации книг по отделам с указанием степени изношенности;
- сведения о новых поступлениях в книгохранилище;
- сведения о списанных книгах;
- запрос данных о наличии произведений конкретного автора;
- запрос данных об имеющихся в книгохранилище книгах по конкретной тематике (тематический каталог).

ЗАДАНИЕ 25
ИНФОРМАЦИОННАЯ СИСТЕМА "ОТЕЛЬ"

Создать информационную систему для автоматизации процесса обработки и данных о работе отеля.

Система должна обрабатывать и хранить данные о служащих отеля, данные о количестве свободных номеров и стоимости проживания в номере в зависимости от категории, а также вести оперативный учет о проживающих, отъезжающих жильцах.

Информационные требования и запросы к системе:

- распечатка данных о проживающих с указанием номера и срока проживания;
- список забронированных номеров с указание срока действия брони;
- данные об иностранцах, проживающих в отеле;

- запрос данных о служащих отеля с указанием стажа работы в отеле,
- запрос об экскурсоводах, их анкетные данные и профиль экскурсий.

ЗАДАНИЕ 26
ИНФОРМАЦИОННАЯ СИСТЕМА "АПТЕКОУПРАВЛЕНИЕ"

Создается система для хранения и выдачи информации для автоматизации процесса обработки данных о работе аптекоуправления.

Система должна обеспечивать выдачу следующих выходных документов:

- Ведомость наличия в аптеках города особо дефицитных лекарств с указанием адреса аптеки, вида упаковки (таблетки, раствор, для инъекций) и их стоимости.

- Сведения о поставщиках импортных лекарств, с указанием наименования лекарства, номера лицензии на изготовление лекарства, объема поставок, срока поставки и вида расчета (рубли, конвертируемая валюта).

- Запрос на поставку лекарств в конкретную аптеку с указанием наименования, количества и срока поставки.

Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Тульский государственный университет»
Технический колледж им. С.И. Мосина

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ САМОСТОЯТЕЛЬНЫХ РАБОТ

по дисциплине «Базы данных»

специальности
09.02.01 Компьютерные системы и комплексы

Тула 2022

УТВЕРЖДЕНЫ
на заседании цикловой комиссии информационных технологий
Протокол от 13 января 2022г. № 6
Председатель цикловой комиссии
_____ И.В Миляева

САМОСТОЯТЕЛЬНАЯ РАБОТА ПО ПРОЕКТИРОВАНИЮ ИНФОЛОГИЧЕСКОЙ МОДЕЛИ ДАННЫХ

1 ОСНОВЫ ТЕОРИИ

Этапы построение модели:

определение информационных элементов

- перечень информационных элементов каждого информационного требования

определение сущностей

- перечень объектов предметной области

определение атрибутов сущностей

- перечень атрибутов объектов на основе информационных элементов требований

определение идентификаторов

- для каждого объекта список ключевых атрибутов

определение связей между сущностями

- спецификация связей между объектами (имя, тип, класс принадлежности)

построение концептуальной инфологической модели

- графическое отражение выделенных сущностей и связей между ними

Идентификатор (имя сущности)	
Первичный ключ	**PK**
атрибут 1 ... атрибут N	

Типы атрибутов

Назначение
- Идентификатор
- Реквизит-признак (качественный)
- Реквизит-основание (количественный)
- Дата/время

Изменяемость свойств
- Динамическое
- Статическое

Обязательность
- Обязательное
- Необязательное

Множественность
- Единичное
- Множественное

Формат
- Символьное
- Числовое
- Дата/время
- Изображение
- Логическое

Способ получения
- Датчики/счетчики
- Внешние
- Производственная

Простой объект в базовой модели

Пример ER - диаграммы

2 ЗАДАНИЕ:

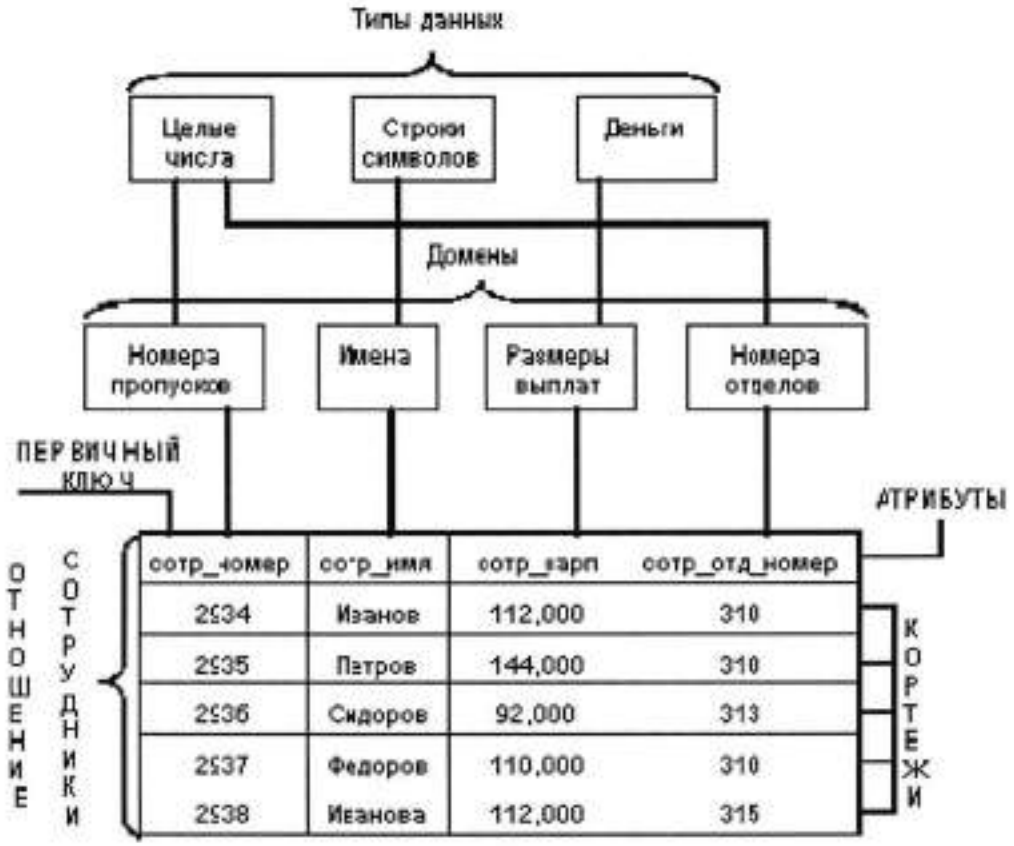
По предложенному преподавателем варианту разработать ER-модель.

САМОСТОЯТЕЛЬНАЯ РАБОТА
ПО ПРОЕКТИРОВАНИЮ, НОРМАЛИЗАЦИИ БАЗ ДАННЫХ

ОСНОВЫ ТЕОРИИ

1 ПРОЕКТИРОВАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХ

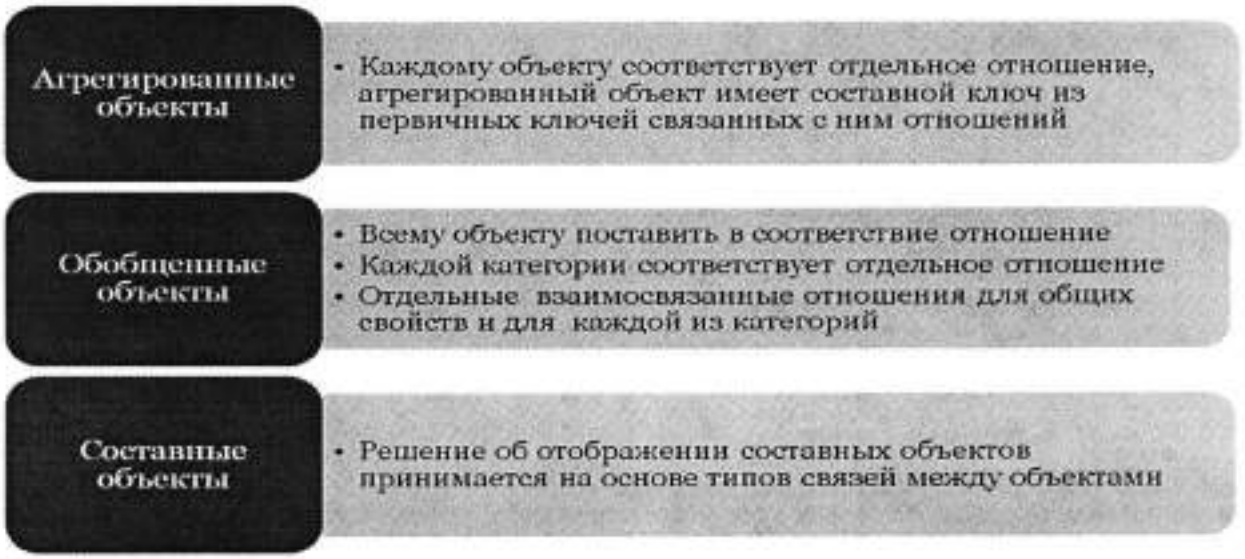
Структура реляционной базы данных



Алгоритм перехода от ER-модели к реляционной модели данных

Простые объекты и единичные свойства	• Строится отношение, в состав которого входят идентификатор и реквизиты, соответствующие единичным свойствам
Множественные свойства объектов	• Строится отношение с составным ключем – первичный ключ сущности и соответствующий реквизит
Условные свойства объектов	• Включается как обычный атрибут • Строится отдельное отношение с составным ключем
Составные свойства объектов	• Всему свойству ставится в соответствие один атрибут • Каждому из составляющих элементов ставится в соответствие одно поле

Отображение связи «многие ко многим»	• Введение промежуточного отношения с составным ключем, включающим первичные ключи исходных сущностей
Отображение связи «один ко многим»	• Введение неключевого атрибута, соответствующего первичному ключу единичного отношения, в множественное отношение
Отображение связи «один к одному»	• Одно обобщающее отношение с первичным ключем одной из сущности • Введение неключевой атрибут в одно из отношений, соответствующий первичному ключу другого отношений



2 ОСНОВЫ ТЕОРИИ ПРИМЕНЕНИЕ ПРОЦЕССА НОРМАЛИЗАЦИИ

2 ЗАДАНИЕ

1 По разработанной в предыдущей работе разработать структуру БД на основе описанного алгоритма.

2 Применить к разработанной структуре БД схему нормализации.

САМОСТОЯТЕЛЬНАЯ РАБОТА:
ПО ТЕМЕ «ПОСТРОЕНИЕ ЗАПРОСОВ»

По предложенному преподавателем варианту разработать несколько вариантов часто используемых запросов по выборке данных из таблиц. Использовать в запросе одно или несколько вычислимых полей. Разработать один или несколько вариантов многотабличных запросов с использованием группировки полей.

ВАРИАНТЫ ЗАДАНИЙ: Выбирается преподавателем из списка заданий (Приложение).

ПРИЛОЖЕНИЕ

ВАРИАНТЫ ПРИМЕРНЫХ ПРЕДМЕТНЫХ ОБЛАСТЕЙ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

ЗАДАНИЕ 1:
ИНФОРМАЦИОННАЯ СИСТЕМА "ДЕТСКАЯ ПОЛИКЛИНИКА"

Система хранит информацию о врачах, пациентах, заболеваниях, детских учреждениях. Информационная система должна предоставлять данные по отдельным запросам:

- информацию о враче (фамилия, специализация, стаж, оклад, совместительство);
- данные о больном (фамилия, возраст, адрес, детское учреждение, место работы родителей, хронические заболевания, прививки, последнее обращение к врачу);
- данные о детском учреждении (наименование, адрес, количество детей, наличие карантина, выявленные инфекционные заболевания, дата последнего профилактического обследования).

Кроме того периодически должны выдаваться следующие ведомости:

- список детских учреждений, в которых зафиксированы инфекционные заболевания;
- статистический отчет по заболеваниям;
- отчет по заболеваемости в детских учебных заведениях.

ЗАДАНИЕ 2:
ИНФОРМАЦИОННАЯ СИСТЕМА "УПРАВЛЕНИЕ ДОРОЖНЫМ ДВИЖЕНИЕМ"

Система хранит информацию об автомобилях, зарегистрированных в области, владельцах автомашин, о нарушениях правил дорожного движения. Системой представляется следующая информация по отдельным запросам:

- информация об автомобиле (регистрационный номер, марка, номер двигателя, дата последнего техосмотра, владелец);
- информация о владельце (номер и марка автомобиля, количество предупреждений в талоне, дата медицинского обследования, фамилия, адрес);
- информация о водителях некоторого автохозяйства.

Периодически и по запросу должны выдаваться следующие ведомости:

- список нарушителей за указанный период;
- список водителей автохозяйств, направленных на медицинскую комиссию;

- статистика нарушений по типам, маркам автомобилей, по автохозяйствам, владельцам.

ЗАДАНИЕ 3:
ИНФОРМАЦИОННАЯ СИСТЕМА "ТЕЛЕАТЕЛЬЕ"

Хранит информацию о типах телевизоров, комплектующих деталях, обслуживающих бригадах, о качестве ремонта. В ателье реализована бригадная форма работы: задание выдается на бригаду, каждая бригада отдельно обеспечивается деталями.

Отдельные запросы, реализуемые системой, могут касаться:

- конкретного телевизора (номер приемной квитанции, тип телевизора, адрес владельца, дата приемки, срок исполнения заказа);
- комплектующей детали (название, маркировка, возможная замена, количество деталей на складе);
- работа отдельной бригады (фамилия бригадира, количество членов бригады, количество выполненных заказов, выполнение плана в стоимостном выражении, рекламация и возврат).

Система должна также обеспечить выдачу следующих выходных документов:

- отчет о выполнении планового задания; состояние склада телеателье;
список заказов с истекшим сроком исполнения.

ЗАДАНИЕ 4:
ИНФОРМАЦИОННАЯ СИСТЕМА "ЗООПАРК"

Хранит информацию о животных, занимаемых ими помещениях, о рационе животных, об обслуживаемом персонале. Система должна представлять информацию по следующим запросам:

- информацию о конкретном животном (кличка, возраст, вид, размещение, состояние здоровья);
- данные о работниках зоопарка (фамилия, возраст, специальность, отделение, стаж, адрес);

Кроме того периодически должны выдаваться следующие документы:

- рекомендуемый рацион для всех животных;
- список заболевших животных;
- список животных с адресами по видам;
- состояние животных, обслуживаемых отдельными работниками;
- ведомость заполнения помещений зоопарка.

ЗАДАНИЕ 5:
ИНФОРМАЦИОННАЯ СИСТЕМА "АБИТУРИЕНТ"

Создается информационная система АБИТУРИЕНТ для автоматизации работы приемной комиссии вуза. Один из фрагментов этой предметной

области требует обработки анкетных данных абитуриентов. Для этого фрагмента разработать программу :

- меню и средства диалога; ввода и изменения данных; подготовки печатных форм.

Анкета включает следующие данные об абитуриенте :

- регистрационный номер; фамилия, имя, отчество; дата рождения;
- окончание среднего учебного заведения; дата окончания;
- наличие красного диплома или золотой медали;
- адрес (город, улица, N дома, квартиры, телефон);
- выбранная специальность.

Исходными документами для заполнения анкеты является аттестат или диплом о среднем образовании, заявление абитуриента. В вузе определен список специальностей, который может изменяться ежегодно. По каждой специальности вуза определен список сдаваемых предметов, например:

- математика (П), - математика (У), - родной язык (П), и т.д.

ИС должна обеспечивать выполнение следующих функций :

- ввод и коррекцию анкетных данных;
- просмотр анкетных данных по специальностям в алфавитном порядке;
- ввод, коррекцию и просмотр специальностей и сдаваемых предметов;
- вывод на печать анкетных данных абитуриентов, имеющих красный диплом или медаль;
- вывод на печать всех инициалов абитуриентов по специальностям в алфавитном порядке с указанием сдаваемых предметов;
- вывод на печать анкетных данных по специальностям.

ЗАДАНИЕ 6:
ИНФОРМАЦИОННАЯ СИСТЕМА «НАЧИСЛЕНИЯ ЗАРПЛАТЫ»

Создается информационная подсистема НАЧИСЛЕНИЯ ЗАРПЛАТЫ для автоматизации начисления заработной платы в бухгалтерии.

Зарплата начисляется работникам вуза, имеющим установленные оклады (сдельных работ нет). На каждого работника хранятся следующие данные: личный номер; ф.и.о.; должность; оклад; семейное положение и число детей; данные о невыходе на работу по болезни (даты заболевания и выздоровления) и т.д.

В период болезни работнику начисляется 50 % зарплаты; 100 % начисляется лишь членам профсоюза. Работникам могут начисляться премии и другие надбавки. С общей суммы зарплаты снимается подоходный налог 8 % и налог за бездетность 6 %.

ИС должна обеспечивать:

- ввод, изменение анкетных данных работников, сведения о болезнях, надбавках; ежемесячный перерасчет зарплаты с выдачей ведомости на экран и на печать.

Разработать программы:

меню и выдачи печатных форм; ввода и изменения данных.

ЗАДАНИЕ 7:
КОНТРОЛЬ ИСПОЛНЕНИЯ ПОРУЧЕНИЙ

Создается информационная подсистема КОНТРОЛЬ ИСПОЛНЕНИЯ ПОРУЧЕНИЙ для некоторой организации. В качестве исходной информации используются данные:

- порядковый номер поручения, название поручения, содержание поручения, дата выдачи поручения, срок исполнения, дата фактического исполнения, исполнитель, кто выдал поручение.

Поручения могут выдавать руководитель организации и руководители подразделений. Ввод всех данных в персональную ЭВМ выполняет один оператор.

ИС должна обеспечивать:

- ввод и коррекцию данных о поручениях, просмотр поручений по некоторой дате, ежедневную печать поручений с текущей датой исполнения (для руководителя организации).

Разработать программы:

меню и выдачи печатных форм, ввода, изменения и просмотра данных.

ЗАДАНИЕ 8
ИНФОРМАЦИОННАЯ СИСТЕМА "СНАБЖЕНИЕ"

Информационная система создается для оптовой базы. Основным назначением оптовой базы является снабжение сети магазинов различными товарами. Отдел снабжения в каждый момент времени должен иметь точные данные о названии товаров, их количестве на складе, о названии магазинов, о названии и количестве каждого вида товара в каждом магазине, о заявках магазинов на текущий год.

Отдел снабжения должен иметь возможность проделывать следующие операции:

- включить новый товар в список товаров на складе; удалить ненужный товар из складского списка; включить новый магазин в список магазинов; удалить ненужный магазин из списка; выполнить поступление некоторого товара на склад; просмотреть информацию о товарах на складе; просмотреть информацию о товарах по магазинам; провести инвентаризацию склада и каждого магазина; выдать магазину товар со склада и отпечатать накладную; ввести заявку магазина на текущий год.

ЗАДАНИЕ 9:
ИНФОРМАЦИОННАЯ СИСТЕМА "ДЕКАНАТ"

Создается информационная система ДЕКАНАТ для автоматизации работы деканата факультета вуза. Пусть максимальное число кафедр на факультете равно 10, кафедры готовят студентов по 12 специальностям.

По каждой специальности имеется учебный план, который содержит список всех предметов, изучаемых на этой специальности, с указанием общего количества лекционных, практических, лабораторных часов, раскладку предметов и курсовых работ по семестрам с указанием количества часов, форму сдачи предмета (зачет/экзамен).

По окончании вступительных экзаменов приёмная комиссия вуза передает в деканат личные дела и списки зачисленных абитуриентов. На каждого студента 1-го курса заводится учебная карточка, в которую заносятся его точные данные, а также список предметов, подлежащих сдаче согласно учебному плану специальности. По мере сдачи предметов и перехода с курса на курс учебная карточка заполняется соответствующими оценками.

По окончании вуза копия учебной карточки выдаётся студенту как приложение к диплому. Учебная карточка выдаётся также при переводе в другой вуз.

Система должна обеспечивать ввод и обработку учебных планов специальностей, учебных карточек студентов, выдачу списков студентов по различным выборкам, должна исключать дублирование данных о предметах. Пользователями ИС являются декан и секретарь деканата. Систему можно расширить, включив обработку данных о преподавателях, их нагрузке. Система ДЕКАНАТ может включаться в ИС всего вуза. Каждое подразделение, кафедра, учебный отдел, научная часть, бухгалтерия, от дел кадров и пр. могут иметь собственные информационные под системы.

ЗАДАНИЕ 10:
ИНФОРМАЦИОННАЯ СИСТЕМА "КАДРЫ"

Разработать ИС КАДРЫ для автоматизации работы отдела кадров предприятия с числом сотрудников до 1000 человек. Система должна функционировать в двух режимах: первичной загрузки данных и текущей обработки информации.

В режиме загрузки базы данных система должна предоставлять ввод данных из личных карточек работающих с контролем вводимой информации. В режиме текущей обработки система должна реализовывать функции:

- обработку данных по движению кадров: прием, увольнение, перемещение;
- получение статистической отчетной информации по уволен ным и работающим в различных разрезах;

- получение справочной информации по данным, содержащимся в личной картотеке;

- ведение табельного учета по отсутствующим на местах.

ЗАДАНИЕ 11:
ИНФОРМАЦИОННАЯ СИСТЕМА "ЗАРПЛАТА"

Разработать ИС "Зарплата" для автоматизации учета труда и заработной платы.

Система должна предоставлять следующие функциональные возможности:

а) начисление аванса за первую половину месяца; основной заработной платы рабочим-повременщикам и распределение премии с учетом коэффициента трудового участия КТУ; основной заработной платы по установленному окладу или тарифу; различного рода премий, доплат, надбавок к основной заработной плате;

б) расчет сдельного заработка рабочим-сдельщикам, работающим по бригадному подряду и индивидуально; пенсии работающим пенсионерам;

в) исчисление налогов;

г) удержание по исполнительным листам; за товары, купленные в кредит; различных ссуд Госбанка; профсоюзных взносов;

д) перечисление сумм организациям Госстраха; в сберегательный банк;

е) формирование расчетно-платежной документации.

ЗАДАНИЕ 12
ИНФОРМАЦИОННАЯ СИСТЕМА "КОММЕРЦИЯ"

В вашем городе создается компьютерный центр коммерческой информации. Его функцией является сбор сведений о предприятиях, фирмах, кооперативах и пр., о производимых ими товарах и услугах, систематизация этих данных по различным параметрам, издание ежеквартальных бюллетеней о сведениях, зарегистрированных за прошедший квартал, выдача интересующей информации по заказу отдельных лиц и организаций.

Информация собирается из периодической печати (рекламные объявления в газетах и журналах),а также может предоставляться самой регистрируемой организацией. Еженедельная порция вводимых данных составляет до 10 новых организаций и до 100 организаций, в которых обновляется информация. Хранимая в базе данных информация об организации должна включать следующие сведения: точное название организации; страна, город и точный адрес, телефон, телекс, телефакс; основные виды деятельности или отрасли производства; вид или наименование производимых товаров или услуг.

Оперативная информация: что приобретается, продается, кто требуется на работу и пр. Данные в базе данных не должны дублироваться.

ИС должна обеспечивать выборку информации по различным критериям, например: "выдать список фирм, занимающихся производством бытовой электроники", далее "в этом списке выбрать список фирм, производящих электронные часы" и т.д.

ЗАДАНИЕ 13
ИНФОРМАЦИОННАЯ СИСТЕМА "КОМИССИОННЫЙ МАГАЗИН"

Информационная система создается для автоматизации работы комиссионного магазина. Основным назначением системы является учет товарооборота магазина.

Система должна предоставлять по запросу выдачу данных о названии товаров, их количестве, о товарах, подлежащих переоценке в связи с истечением срока и обеспечивать выполнение следующие операции: включить новый товар в список товаров; удалить реализованный товар из списка; просмотреть информацию о товарах по магазину; провести инвентаризацию магазина; выдать ведомость товаров сданных в магазин на определенную дату.

ЗАДАНИЕ 14
ИНФОРМАЦИОННАЯ СИСТЕМА "АВТОСЕРВИС"

Создается система для хранения и выдачи информации об автомобилях, находящихся на обслуживании, владельцах автомобилей, неисправностях и о наличии деталей на складе.

Информация об автомобиле должна содержать данные: номер квитанции, марку автомобиля, неисправность, детали, необходимые для ремонта. Информация о владельцах должна содержать следующие данные: Ф.И.О., номер квитанции, марку автомобиля.

Информация о запчастях должна содержать данные: код детали, название, стоимость, расценка ее замены.

Периодически по запросу должны выдаваться следующие сведения:

- перечень Ф.И.О. клиентов с указанием марки автомобиля, вида неисправности, номера накладной и перечня используемых деталей.

Запрос о наличии определенной детали на складе. Результаты инвентаризации склада.

ЗАДАНИЕ 15
ИНФОРМАЦИОННАЯ СИСТЕМА "УНИВЕРМАГ"

Информационная система хранит информацию о товарах, поставщиках товаров, отделах универмага и сотрудниках, работающих в универмаге.

Система должна представлять информацию по следующим запросам:

- информация о поставщиках товаров,
- информация о товарах.

Система должна обеспечивать выдачу следующих выходных документов:

- список фирм, поставляющих товар; список товаров, находящихся на реализации; список цен на товары; список стран-производителей товаров; адреса фирм-поставщиков.

ЗАДАНИЕ 16
ИНФОРМАЦИОННАЯ СИСТЕМА "АВТОХОЗЯЙСТВО"

Система хранит информацию об автомобилях, зарегистрированных в данном автохозяйстве, шоферах автомашин, о нарушениях правил дорожного движения. Системой представляется следующая информация по отдельным запросам:

- информация об автомобиле (регистрационный номер, марка, номер двигателя, дата последнего техосмотра, владелец); информация о шофере автохозяйства (номер и марка автомобиля, количество предупреждений в талоне, дата медицинского обследования, фамилия, адрес);

Периодически и по запросу должны выдаваться следующие ведомости:

- список нарушителей за указанный период; список водителей автохозяйства, направленных на медицинскую комиссию; статистика нарушений по типам, маркам автомобилей.

ЗАДАНИЕ 17
ИНФОРМАЦИОННАЯ СИСТЕМА "ДЕТСКАЯ БОЛЬНИЦА"

Система хранит информацию о врачах, пациентах, заболеваниях, детских учреждениях. Информационная система должна предоставлять данные по отдельным запросам:

информацию о враче (фамилия, специализация, стаж, оклад, совместительство); данные о больном (фамилия, возраст, адрес, детское учреждение, место работы родителей, хронические заболевания, прививки, последнее обращение к врачу); информацию о лекарствах для приобретения их в аптеке.

Кроме того периодически должны выдаваться следующие ведомости:

- список больных детских учреждений, в которых зафиксированы инфекционные заболевания; перечень дефицитных лекарств, отсутствующих в больнице.

ЗАДАНИЕ 18
ИНФОРМАЦИОННАЯ СИСТЕМА «АЭРОПОРТ»

Создается система для хранения и выдачи информации о самолетах, о расписании самолетов, о сотрудниках аэропорта.

Периодически по запросу должны выдаваться следующие сведения:

- информацию о самолетах (тип самолета, номер самолета, номер обслуживающей самолет бригады);

- информацию о расписании (номер рейса, пункт отправления, пункт назначения, время отправления, время полета, стоимость билета),

- Информацию о летном составе (Ф.И.О., должность, стаж работы, адрес, оклад).

ЗАДАНИЕ 19
ИНФОРМАЦИОННАЯ СИСТЕМА "СКЛАД"

Информационная система создается для объекта склад.

Периодически по запросу должны выдаваться данные о названии товаров, их количестве на складе.

Оператор ЭВМ должен иметь возможность проделывать следующие операции обработки данных:

- включение нового товара в перечень товаров на складе;
- удаление ненужный товар из списка;
- просмотреть информацию о наличии товаров;

а также выдачу документов: инвентаризации склада; накладную на получение товаров со склада; ведомость пересортировки товаров.

ЗАДАНИЕ 20
ИНФОРМАЦИОННАЯ СИСТЕМА "СПОРТ"

Создается информационная система СПОРТ для автоматизации обработки данных о проводимых соревнованиях и чемпионатах.

Перечень информационных требований к системе:

- выдача данных о результатах чемпионата по определенному виду спорта с указанием страны, занятого места и количества набранных баллов,
- выдача информации о спортсменах-победителях чемпионата,
- перечень чемпионатов, проводимых в стране с указанием места, времени, вида спорта, стран участниц

ЗАДАНИЕ 21
ИНФОРМАЦИОННАЯ СИСТЕМА "АВТОБУСНОЕ ДВИЖЕНИЕ"

Информационная система создается для автоматизации процесса обработки и хранения данных о проверке состояния машин и возможности ответов на запросы, представленные следующим набором требований:

- ведомость контроля за состоянием машин, закрепленных за бригадой;
- расписание рейсов автобусов с указанием направления движения и временем отправления,
- расписание отправления автобусов из АТП (для диспетчера);
- ведомость выполнения расписания движения.

Периодически и по запросу должны выдаваться следующие ведомости:

- данных о водителях бригады, с указанием стажа работы, количества нарушений, виды поощрения;
- данные об авариях и невыхода транспорта с указанием причины.

ЗАДАНИЕ 22
ИНФОРМАЦИОННАЯ СИСТЕМА "ЖЕЛЕЗНОДОРОЖНЫЙ ТРАНСПОРТ"

Информационная система создается для автоматизации процесса обработки и хранения оперативных данных и справочной информации о работе железнодорожного транспорта. Она должна предусматривать ответы на некоторые запросы и должна отвечать следующим требованиям:

- оперативный ввод информации о заявках на перевозку грузов и пассажиров;
- печать графика движения железнодорожных составов пассажирских и грузовых) исходя из пропускной способности транспортного узла.
- печать графика работы железнодорожных бригад (машинистов), учитывая болезни, отпуска, выполнения обязанностей и т.д.)
- расчет графика профилактики , ремонта подвижного состава, транспортных путей;
- выдавать ведомость о перевозках пассажиров по дням, месяцам;
- выдавать ведомость состава обслуживающего персонала локомотивных бригад.

Периодически и по запросу предусмотреть выдачу оперативных данных:

- о нарушении графика движения, с указанием причин и виновных,
- о наличии неиспользуемого транспортного парка.

ЗАДАНИЕ 23
ИНФОРМАЦИОННАЯ СИСТЕМА "ВОЕНКОМАТ"

Создать информационную систему для автоматизации процесса обработки и хранения данных о результатах прохождения призывной комиссии и возможности ответов на некоторые вопросы, представленные следующими требованиями:

- распечатка учетно-послужных карточек призывников, содержащих анкетные данные о призывнике,
- распечатка медицинских ведомостей призывников, с указанием состава врачей приемной комиссии и результатах пригодности призывника к военной службе;
- распечатка ведомостей призывной медицинской комиссии с указанием ФИО призывника, кода учетно-призывной карточки и рода войск;

- запрос данных о составе медицинской комиссии с указанием специальностей врачей, номера больницы и других анкетных данных;

- запрос данных о распределении призывников по родам войск с указание количества призывников в разрезе по роду войск.

ЗАДАНИЕ 24
ИНФОРМАЦИОННАЯ СИСТЕМА "КНИГОХРАНИЛИЩЕ"

Создать информационную систему для автоматизации процесса обработки и хранения данных об имеющихся в данном книгохранилище книгах и возможности ответов на некоторые запросы, представленные следующим перечнем информационных требований:

- результаты инвентаризации книг по отделам с указанием степени изношенности;
- сведения о новых поступлениях в книгохранилище;
- сведения о списанных книгах;
- запрос данных о наличии произведений конкретного автора;
- запрос данных об имеющихся в книгохранилище книгах по конкретной тематике (тематический каталог).

ЗАДАНИЕ 25
ИНФОРМАЦИОННАЯ СИСТЕМА "ОТЕЛЬ"

Создать информационную систему для автоматизации процесса обработки и данных о работе отеля.

Система должна обрабатывать и хранить данные о служащих отеля, данные о количестве свободных номеров и стоимости проживания в номере в зависимости от категории, а также вести оперативный учет о проживающих, отъезжающих жильцах.

Информационные требования и запросы к системе:

- распечатка данных о проживающих с указанием номера и срока проживания;
- список забронированных номеров с указание срока действия брони;
- данные об иностранцах, проживающих в отеле;
- запрос данных о служащих отеля с указанием стажа работы в отеле,
- запрос об экскурсоводах, их анкетные данные и профиль экскурсий.

ЗАДАНИЕ 26
ИНФОРМАЦИОННАЯ СИСТЕМА "АПТЕКОУПРАВЛЕНИЕ"

Создается система для хранения и выдачи информации для автоматизации процесса обработки данных о работе аптекоуправления.

Система должна обеспечивать выдачу следующих выходных документов:

- Ведомость наличия в аптеках города особо дефицитных лекарств с указанием адреса аптеки, вида упаковки (таблетки, раствор, для инъекций) и их стоимости.
- Сведения о поставщиках импортных лекарств, с указанием наименования лекарства, номера лицензии на изготовление лекарства, объема поставок, срока поставки и вида расчета (рубли, конвертируемая валюта).
- Запрос на поставку лекарств в конкретную аптеку с указанием наименования, количества и срока поставки.