РОСЖЕЛДОР

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Ростовский государственный университет путей сообщения»

(ФГБОУ ВО РГУПС)

Е.В. Голубенко, А.Г. Кулькин

УПРАВЛЕНИЕ ДАННЫМИ

Учебно-методическое пособие
для лабораторных работ

Ростов-на-Дону
2017

УДК 681.3(07) + 06

Рецензент – доктор технических наук, профессор М.А. Бутакова

Голубенко, Е.В.

Управление данными: учебно-методическое пособие для лабораторных работ / Е.В. Голубенко, А.Г. Кулькин; ФГБОУ ВО РГУПС. – Ростов н/Д, 2017. – 55 с.

В учебно-методическом пособии приведены задания и методика выполнения лабораторных работ по дисциплине «Управление данными», а также примеры выполнения лабораторных работ.

Учебно-методическое пособие предназначено для студентов направления подготовки «Информационные системы и технологии» и «Информатика и вычислительная техника», а также для студентов всех специальностей, изучающих дисциплины «Управление данными», «Базы данных».

Одобрено к изданию кафедрой «Вычислительная техника и автоматизированные системы управления».

Содержание

ЛАБОРАТОРНАЯ РАБОТА 1. ИНФОЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

ЗАДАНИЕ 1

Создание инфологической и логической моделей базы данных.

1. Разработайте информационно-логическую модель реляционной базы данных.

2. Разработайте логическую модель реляционной базы данных

ТЕХНОЛОГИЯ РАБОТЫ

1. Перед разработкой информационно-логической модели реляционной базы данных рассмотрим, из каких информационных объектов должна состоять эта база данных. Можно выделить три объекта, которые не будут обладать избыточностью, - Студенты, Дисциплины и Преподаватели. Представим состав реквизитов этих объектов в виде "название объекта (перечень реквизитов)": Студенты (код студента, фамилия, имя, отчество, номер группы, дата рождения, стипендия, оценки). Дисциплины (код дисциплины, название дисциплины), Преподаватели (код преподавателя, фамилия, имя, отчество, дата рождения, телефон, заработная плата).

Рассмотрим связь между объектами Студенты и Дисциплины. Студент изучает несколько дисциплин, что соответствует многозначной связи и отражено на рис. 1 двойной стрелкой. Понятно, что каждая дисциплина изучается множеством студентов. Это тоже многозначная связь, обозначаемая двойной стрелкой (связь "один" обозначена одинарной стрелкой). Таким образом, связь между объектами Студенты и Дисциплины - Многие-ко-многим (M : N).

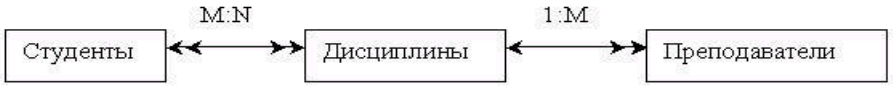

Рис.1. Типы связей между объектами Студенты, Дисциплины и Преподаватели

Множественные связи усложняют управление базой данных, например, в СУБД Access 2010 при множественных связях нельзя использовать механизм каскадного обновления. Поэтому использовать такие связи нежелательно и нужно строить реляционную модель, не содержащую связей типа Многие-ко-многим. В Access 2010 для контроля целостности данных с возможностью каскадного обновления и удаления данных необходимо создать вспомогательный объект связи, который состоит из ключевых реквизитов связываемых объектов и который может быть дополнен описательными реквизитами. В нашем случае таким новым объектом для связи служит объект

Оценки, реквизитами которого являются код студента, код дисциплины и оценки. Каждый студент имеет оценки по нескольким дисциплинам, поэтому связь между объектами Студенты и Оценки будет Один-ко-многим (1:M). Каждую дисциплину сдает множество студентов, поэтому связь между объектами Дисциплины и Оценки также будет Один-ко-многим (1:M). В результате получаем информационно-логическую модель базы данных, приведенную на рис. 2.

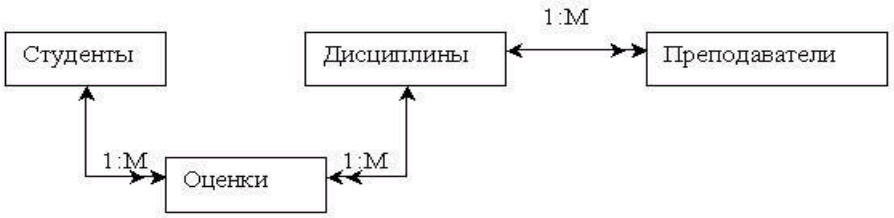

Рис. 2. Информационно-логическая модель реляционной базы данных

2. В реляционной базе данных в качестве объектов рассматриваются отношения, которые можно представить в виде таблиц. Таблицы между собой связываются посредством общих полей, т.е. одинаковых по форматам и, как правило, по названию, имеющихся в обеих таблицах. Рассмотрим, какие общие поля надо ввести в таблицы для обеспечения связности данных. В таблицах Студенты и Оценки таким полем будет "Код студента", в таблицах Дисциплины и Оценки - "Код дисциплины", в таблицах Преподаватели и Дисциплины - "Код дисциплины". Выбор цифровых кодов вместо фамилий или названий дисциплин обусловлен меньшим объемом информации в таких полях: например, число "2". по количеству символов значительно меньше слова "математика". В соответствии с этим логическая модель базы данных представлена на рис. 3, где жирными буквами выделены ключевые поля.

Рис. 3. Логическая модель базы данных

ЗАДАНИЕ 2

Создание реляционной базы данных.

1.Создайте базу данных Деканат.

2.Создайте структуру таблицы Студенты.

3.Создайте структуру таблицы Дисциплины.
4. Измените структуру таблицы Преподаватели.
5. Создайте структуру таблицы Оценки.
6. Разработайте схему данных, т.е. создайте связи между таблицами.

ТЕХНОЛОГИЯ РАБОТЫ

1. Создайте базу данных Деканат, выполнив следующие действия:

– загрузите Access, в появившемся окне выберите пункт Новая база данных, затем щелкните по кнопке <ОК>;

– в окне <Файл новой базы данных> задайте имя (пункт Имя файла) и выберите папку (пункт Папка), где ваша база будет находиться. По умолчанию Access предлагает имя базы dbl, а тип файла - Базы данных Access. Имя задайте Деканат, а тип файла оставьте прежним, так как другие типы файлов нужны в специальных случаях;

– щелкните по кнопке <Создать>

2. Создайте структуру таблицы Студенты. Для этого:

– в окне базы данных выберите вкладку Таблицы, а затем щелкните по кнопке <Создать>;

– в окне "Новая таблица" выберите пункт Конструктор и щелкните по кнопке <ОК>. В результате проделанных операций открывается окно таблицы в режиме конструктора, в котором следует определить поля таблицы;

Таблица 1

Таблица 4.3.

Имя поля	**Тип данных**	**Размер поля**
Код студента	Числовой	Целое
Фамилия	Текстовый	15
Имя	Текстовый	12
Отчество	Текстовый	15
Номер группы	Числовой	Целое
Телефон	Текстовый	9
Стипендия	Логический	Да/Нет

- определите поля таблицы в соответствии с табл.3;
- в качестве ключевого поля задайте "Код студента". Для этого щелкните по полю "Код студента" и по кнопке на панели инструментов или выполните команду Правка, Ключевое поле;
- закройте таблицу, задав ей имя Студенты.

Примечание. Заполнять таблицу данными пока не требуется, это будет сделано в режиме формы.

3. Создайте структуру таблицы Дисциплины аналогично п. 2 в соответствии с табл. 2.

Таблица 2

Имя поля	**Тип данных**	**Размер поля**
Код дисциплины	Числовой	Целое
Название дисциплины	Текстовый	30

В качестве ключевого поля задайте "Код дисциплины". Заполняться эта таблица будет также в режиме формы.

4. Структура таблицы Преподаватели уже создана в работе 1 и заполнена данными, этому для работы используйте эту таблицу с одним лишь изменением - в соответствии с рис. 3 в структуру таблицы надо добавить поле "Код дисциплины" и заполнить его в соответствии с данными табл. 2.

5. Создайте структуру таблицы Оценки аналогично п. 2 в соответствии с табл. 3.

Таблица 3

Имя поля	Тип данных	Размер поля
Код студента	Числовой	Целое
Код дисциплины	Числовой	Целое
Оценки	Числовой	Байт

В этой таблице задавать ключевое поле не надо, так как данные во всех полях могут повторяться. Эта таблица, аналогично предыдущим, будет заполняться в режиме формы.

6. Разработайте схему данных, т.е. создайте связи между таблицами. Для этого:

– щелкните по кнопке на панели инструментов или выполните команду Сервис, Схема данных. На экране появится окно "Схема данных";

– щелкните по кнопке на панели инструментов или выполните команду Связи, Добавить таблицу;

– в появившемся окне будет выделено название одной таблицы. Щелкните по кнопке <Добавить>;

– переведите выделение на имя следующей таблицы и щелкните по кнопке <Добавить>. Аналогично добавьте оставшиеся две таблицы;

– закройте окно, щелкнув по кнопке <Закрыть>;

– создайте связь между таблицами Дисциплины и Оценки. Для этого подведите курсор мыши к полю "Код дисциплины" в таблице Дисциплины щелкните левой кнопкой мыши и, не отпуская ее, перетащите курсор на поле "Код дисциплины" в таблицу Оценки, а затем отпустите кнопку мыши. На экране откроется окно "Связи";

– установите флажок ("галочку") в свойстве Обеспечение целостности данных, щелкнув по нему;

– установите флажок в свойстве Каскадное обновление связанных полей и Каскадное удаление связанных записей.

Примечание. Задание каскадного обновления связанных полей и каскадного удаления связанных записей позволит вам отредактировать записи только в таблице Дисциплины, а в таблице Оценки эти действия будут со связанными записями выполняться автоматически. Например, если вы удалите из таблицы Дисциплины один предмет, то в таблице Оценки удалятся все строки, связанные с этим предметом.

– щелкните по кнопке <Создать>. Связь будет создана;

– аналогично создайте связи между полем "Код дисциплины" в таблице Дисциплины и полем "Код дисциплины" в таблице Преподаватели, а также между полем "Код студента" в таблице Студенты и полем "Код студента" в таблице Оценки. Результат представлен на рис. 4;

– закройте окно схемы данных, ответив ДА на вопрос о сохранении макета.

Рис. 4. Структура таблицы Студенты

ЗАДАНИЕ 3

Создание форм для ввода данных в таблицы.

1. Создайте форму Студенты.
2. Заполните данными таблицу Студенты посредством формы Студенты.
3. Создайте форму Дисциплины.
4. Заполните данными таблицу Дисциплины посредством формы Дисциплины.
5. Создайте форму Оценки.
6. Заполните данными таблицу Оценки посредством формы Оценки.

ТЕХНОЛОГИЯ РАБОТЫ

1. Для создания формы Студенты:

– откройте вкладку Формы;

– щелкните по кнопке <Создать>;

– в открывающемся списке выберите таблицу Студенты,

– выберите пункт Автоформа: ленточная;

– щелкните по кнопке <ОК>. Форма для ввода данных создана;

Примечание. Если вас не удовлетворяет макет, вы можете перейти в режим конструктора и изменить макет, передвигая и изменяя размеры элементов - заголовков полей и ячеек для ввода данных. Достаточно щелкнуть по элементу - он выделяется прямоугольной рамкой, и вы можете изменять размеры и двигать элемент. Если вы хотите изменить другие параметры элемента, надо, по выделенному элементу щелкнуть правой клавишей мыши, и откроется окно свойств элемента. В силу ограниченности объема раздела описать все свойства нет возможности, но их можно изучить самостоятельно по справочной системе, а многие свойства понятны уже из своего названия.

2. Заполните данными, приведенными в табл. 6, таблицу Студенты посредством формы.

Таблица 4

Код студента	Фамилия	Имя	Отчество	Номер группы	Телефон	Стипендия
1	Арбузов	Николай	Николаевич	151	260-15-63	Да
2	Киршин	Петр	Валерьевич	151	110-67-82	Да
3	Кривинский	Сергей	Николаевич	151	172-97-21	Нет
4	Крылова	Елена	Петровна	151	130-31-87	Да
5	Кульчий	Григорий	Викторович	151	269-53-75	Да
6	Патрикеев	Олег	Борисович	152	234-11-63	Нет
7	Перлов	Кирилл	Николаевич	152	312-21-33	Нет
8	Соколова	Наталия	Петровна	152	166-87-24	Нет
9	Степанская	Ольга	Витальевна	152	293-43-77	Да
10	Тимофеев	Сергей	Трофимович	152	260-11-57	Да

Примечание. Переход между ячейками лучше выполнять клавишей <Tab> либо мышью. Существуют и другие варианты перехода по строкам или полям с помощью различных клавиш и их комбинаций. Обычно их используют опытные пользователи, не любящиеработать с мышью.

Закройте форму, задав ей имя Студенты.

3. Создайте форму Дисциплины аналогично п.1.

4. Заполните данными, приведенными в табл. 5, таблицу Дисциплины посредством, формы и закройте форму, задав ей имя Дисциплины.

5. Создайте форму Дисциплины аналогично п. 1.

6. Заполните данными, приведенными в табл. 6, таблицу Оценки. Посредством формы закройте форму, задав ей имя Оценки.

Таблица 5

Код дисциплины	Название дисциплины
1	Информатика
2	Математика
3	Физика
4	Экономика

Таблица 6

Код студента	Код дисциплины	Оценки	Код студента	Код дисциплины	Оценки
1	1	4	6	1	5
1	2	5	6	2	4
1	3	4	6	3	5
1	4	4	6	4	4
2	1	5	7	1	4
2	2	5	7	2	3
2	3	4	7	3	4
2	4	4	7	4	3
3	1	3	8	1	3
3	2	5	8	2	5
3	3	4	8	3	5
3	4	3	8	4	4
4	1	4	9	1	4
4	2	4	9	2	4
4	3	5	9	3	4
4	4	4	9	4	4
5	1	5	10	I	5
5	2	5	10	2	5
5	3	5	10	3	5
5	4	5	10	4	5

ЛАБОРАТОРНАЯ РАБОТА 2. ДАТАЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ. СОЗДАНИЕ РЕЛЯЦИОННЫХ ТАБЛИЦ

ЗАДАНИЕ 1

Создание базы данных.

1. Создайте новую базу данных.
2. Создайте таблицу базы данных.
3. Определите поля таблицы в соответствии с табл.7.
4. Сохраните созданную таблицу.

Таблица 7. Таблица данных Преподаватели

Имя поля	Тип данных	Размер поля
Код преподавателя	Счетчик	
Фамилия	Текстовый	15
Имя	Текстовый	15
Отчество	Текстовый	15
Дата рождения	Дата/время	Краткий
Должность	Текстовый	9
Дисциплина	Текстовый	11
Телефон	Текстовый	9
Зарплата	Денежный	

ТЕХНОЛОГИЯ РАБОТЫ

Для создания новой базы данных:

– загрузите Access, в появившемся окне выберите пункт Новая база данных;

– в окне "Файл новой базы данных" задайте имя вашей базы (пункт Имя Файла) и выберите папку (пункт Папка), где ваша база данных будет находиться. По умолчанию Access предлагает вам имя базы db1, а тип файла - Базы данные Access. Имя задайте Преподаватели, а тип файла оставьте прежним, так как другие типы файлов нужны в специальных случаях;

– щелкните по кнопке <Создать>.

Для создания таблицы базы данных:

– в окне базы данных выберите вкладку Таблицы, а затем щелкните по кнопке <Создать>;

– в окне "Новая таблица" выберите пункт Конструктор и щелкните по кнопке <ОК>. В результате проделанных операций открывается окно

таблицы в режим конструктора (рис. 5), в котором следует определить поля таблицы.

Для определения полей таблицы:

– введите в строку столбца Имя поля имя первого поля Код преподавателя;

– в строке столбца "Тип данных" щелкните по кнопке списка и выберите тип данных Счетчик. Поля вкладки Общие оставьте такими, как предлагает Access.

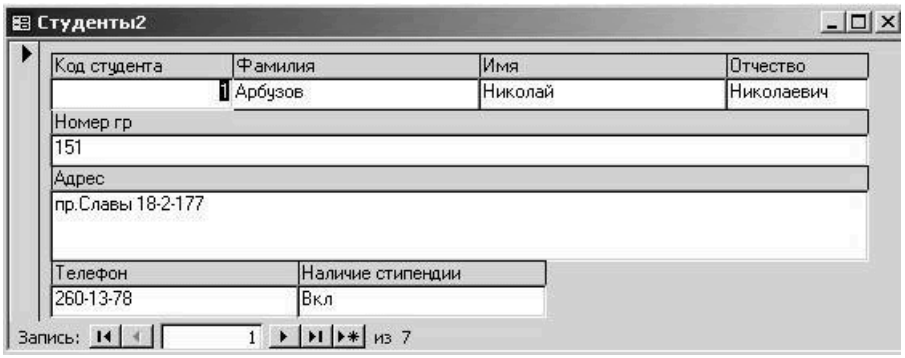

Рис. 5. Окно таблицы в режиме конструктора - в этом режиме вводятся имена и типы полей таблицы

Примечание. Заполнение строки столбца "Описание" необязательно и обычно используется для внесения дополнительных сведений о поле.

Для определения всех остальных полей таблицы базы данных Преподаватели в соответствии с табл. 1 выполните действия, аналогичные указанным выше.

Внимание! Обратите внимание на вкладку *Общие* в нижней части экрана. Советуем изменить данные в пункте Размер поля, а остальные пункты оставить по умолчанию (их функции рассмотрим далее). Например, для текстового типа данных Access предлагает по умолчанию длину 50 символов. Но вряд ли поле "Фамилия" будет содержать более 15 символов, хотя лучше точно подсчитать, сколько символов в самой длинной фамилии. Не бойтесь ошибиться – в дальнейшем можно скорректировать длину поля. Для числового типа Access предлагает Длинное целое, но ваши данные могут быть либо небольшие целые числа (в диапазоне от -32768 до 32767) – тогда надо выбрать Целое, либо дробные числа – тогда надо выбрать с плавающей точкой. Для выбора необходимого параметра надо щелкнуть по полю, а затем нажать появившуюся кнопку списка и выбрать необходимые данные. В результате ваша таблица будет иметь более компактный вид, а объем базы данных уменьшится.

4. Для сохранения таблицы:

– выберите пункт меню Файл, Сохранить;

– в диалоговом окне "Сохранение" введите имя таблицы Преподаватели',

– щелкните по кнопке <ОК>.

Примечание. В результате щелчка по кнопке <ОК> Access предложит вам задать ключевое поле (поле первичного ключа), т.е. поле, однозначно

идентифицирующее каждую запись. Для однотабличной базы данных это не столь актуально, как для многотабличной, поэтому щелкните по кнопке <Нет>.

ЗАДАНИЕ 2

Заполнение базы данных.

1. Введите ограничения на данные, вводимые в поле "Должность"; должны вводиться только слова Профессор, Доцент или Ассистент.

2. Задайте текст сообщения об ошибке, который будет появляться на экране при вводе неправильных данных в поле "Должность".

3. Задайте значение по умолчанию для поля "Должность" в виде слова Доцент.

4. Введите ограничения на данные в поле <Код>; эти данные не должны повторяться.

5. Заполните таблицу данными в соответствии с табл. 8 и проверьте реакцию системы на ввод неправильных данных в поле "Должность".

6. Измените ширину каждого поля таблицы в соответствии с шириной данных.

7. Произведите поиск в таблице преподавателя Миронова.

8.Произведите замену данных: измените заработную плату ассистенту Сергеевой с 10000 р, на 11000 р.

9. Произведите сортировку данных в поле "Год рождения" по убыванию,

10. Произведите фильтрацию данных по полям "Должность" и "Дисциплина".

Просмотрите созданную таблицу, как она будет выглядеть на листе бумаги при печати.

Таблица 8

Код	**Фамилия**	**Имя**	**Отчество**	**Дата рожд.**	**Должность**	**Дисциплина**	**Телефон**	**Зар. плата**
1	Истомин	Ремир	Евгеньевич	23.10.54	Доцент	Информатика	110-44-68	16000
2	Миронов	Павел	Юрьевич	25.07.40	Профессор	Экономика	312-21-40	24000
3	Гришин	Евгений	Сергеевич	05.12.67	Доцент	Математика	260-23-65	15000
4	Сергеева	Ольга	Ивановна	12.02.72	Ассистент	Математика	234-85-69	10000
5	Емец	Татьяна	Ивановна	16.02.51	Доцент	Экономика	166-75-33	16000
6	Игнатова	Татьяна	Павловна	30.05.66	Доцент	Информатика	210-36-98	15000
7	Миронов	Алексей	Николаевич	30.07.48	Доцент	Физика	166-75-33	16000

ТЕХНОЛОГИЯ РАБОТЫ

Для задания условия на значение для вводимых данных:

– войдите в режим Конструктор для проектируемой таблицы. Если вы находитесь в окне базы данных, то выберите вкладку Таблицы и щелкните по кнопке <Конструктор>. Если вы находитесь в режиме таблицы, то щелкните по кнопке на панели инструментов или выполните команду Вид, Конструктор;

– в верхней части окна щелкните по полю "Должность";

– в нижней части окна щелкните по строке параметра Условие на значение;

– щелкните по кнопке для определения условий на значение при помощи построителя выражений;

– в появившемся окне напишите слово Профессор, затем щелкните по кнопке (эта кнопка выполняет функцию ИЛИ), напишите Доцент, снова щелкните по этой же кнопке, напишите Ассистент и щелкните по кнопке <ОК>. Таким образом, вы ввели условие, при котором в поле "Должность" могут вводиться только указанные значения.

2. В строке Сообщение об ошибке введите предложение "Такой должности нет, правильно введите данные".

3. В строке Значение по умолчанию введите слово "Доцент".

4. Введите ограничения на данные в поле "Код". Здесь ограничения надо вводить не совсем обычным способом. Дело в том, что коды преподавателей не должны повторяться, а также должна быть обеспечена возможность их изменения (из-за последнего условия в этом поле нельзя использовать тип данных Счетчик, в котором данные не повторяются). Для выполнения второго условия пришлось задать в поле "Код" тип данных Числовой, а для выполнения первого условия сделайте следующее:

– щелкните по строке параметра Индексированное поле;

> Примечание. Индекс – это средство Access, ускоряющее поиск и сортировку данных в таблице. Ключевое поле (поле первичного ключа) таблицы индексируется автоматически. Не допускается создание индексов для полей типа MEMO и Гиперссылка vim полей объектов OLE. Свойство Индексированное поле определяет индекс, создаваемый по одному полю. Индексированное поле может содержать как уникальные, так и повторяющиеся значения. Допускается создание произвольного количества индексов.

– выберите в списке пункт Да (совпадения не допускаются);

– перейдите в режим Таблица, щелкнув по кнопке на панели инструментов или выполнив команду Вид, Режим таблицы. На вопрос о сохранении таблицы щелкните по кнопке <Да>.

5. Введите данные в таблицу в соответствии с табл. 3. Попробуйте в поле <Должность> любой записи ввести слово Лаборант. Посмотрите, что

получилось. На экране должно появиться сообщение; "Такой должности нет, правильно введите данные". Введите правильное слово.

6. Для изменения ширины каждого поля таблицы в соответствии с шириной данных:

– щелкните в любой строке поля "Код";

– выполните команду Формат, Ширина столбца;

– в появившемся окне щелкните по кнопке <По ширине данных>. Ширина поля изменится; проделайте эту операцию с остальными полями.

7. Для поиска в таблице преподавателя Миронова:

– переведите курсор в первую строку поля "Фамилия";

– выполните команду Правка, Найти;

– в появившейся строке параметра Образец введите Миронов;

– в строке параметра Просмотр должно быть слово ВСЕ (имеется в виду искать по всем записям);

– в строке параметра Совпадение выберите из списка С любой частью поля;

– в строке параметра Только в текущем поле установите флажок (должна стоять галочка);

– щелкните по кнопке <Найти>. Курсор перейдет на вторую запись и выделит слово Миронов;

– щелкните по кнопке <Найти далее>. Курсор перейдет на седьмую запись и также выделит слово Миронов;

– щелкните по кнопке <Закрыть> для выхода из режима поиска.

8. Для замены заработной платы ассистенту Сергеевой с 10000 р. на 11000 р.:

– переведите курсор в первую строку поля "Зарплата";

– выполните команду Правка, Заменить;

– в появившемся окне в строке Образец введите 10000;

– в строке Заменить на введите 11000. Обратите внимание на остальные опции - вам надо вести поиск по всем записям данного поля;

– щелкните по кнопке <Найти далее>. Курсор перейдет на четвертую запись, но здесь не нужно менять данные, поэтому снова щелкните по кнопке <Найти далее>. Курсор перейдет на девятую запись - это то, что нам надо;

– щелкните по кнопке <Заменить>. Данные будут изменены;

Примечание. Чтобы заменить сразу все данные, надо воспользоваться кнопкой <Заменить все>. щелкните по кнопке <Закрыть>.

9. Для сортировки данных в поле "Год рождения" по убыванию:

– щелкните по любой записи поля "Год рождения";

– щелкните по кнопке на панели управления или выполните команду Записи, Сортировка, Сортировка по убыванию. Все данные в таблице будут отсортированы в соответствии с убыванием значений в поле "Год рождения".

Для фильтрации данных по полям "Должность" и "Дисциплина":

– щелкните по записи Доцент поля "Должность";

– щелкните по кнопке или выполните команду Записи, Фильтр, Фильтр по выделенному. В таблице останутся только записи о преподавателях - доцентах;

– щелкните по записи Информатика поля "Дисциплина";

– щелкните по кнопке или выполните команду Записи, Фильтр, Фильтр по выделенному. В таблице останутся только записи о преподавателях - доцентах кафедры информатики;

– для отмены фильтрации щелкните по кнопке на панели инструментов или выполните команду Записи, Удалить фильтр. В таблице появятся все данные.

Для просмотра созданной таблицы:

– щелкните по кнопке или выполните команду Файл, Предварительный, просмотр. Вы увидите таблицу как бы на листе бумаги;

– закройте окно просмотра.

Примечание. Если вы захотите изменить поля или ориентацию таблицы на листе бумаги, выполните команду Файл, Параметры страницы. В открывшемся окне можете изменять указанные параметры.

Если у вас есть принтер, то созданную страницу можете распечатать.

ЗАДАНИЕ 3

Ввод и просмотр данных посредством формы.

1. С помощью Мастера форм создайте форму Состав преподавателей (тип - форма один столбец).

2. Найдите запись о доценте Гришине, находясь в режиме формы.

3. Измените зарплату ассистенту Сергеевой с 1000 р. на 1100 р.

4. Произведите сортировку данных в поле "Фамилия" по убыванию.

5. Произведите фильтрацию данных по полю "Должность".

6. Измените название поля "Дисциплина" на "Преподаваемая дисциплина".

7. Просмотрите форму с точки зрения того, как она будет выглядеть на листе бумаги.

ТЕХНОЛОГИЯ РАБОТЫ

1. Для создания формы Состав преподавателей:

– откройте вкладку Формы в окне базы данных;

– щелкните по кнопке <Создать>;

– в появившемся окне выберите (подведите курсор мыши и щелкните левой кнопкой) пункт Мастер форм;

– щелкните по значку списка в нижней части окна;

– выберите из появившегося списка таблицу Преподаватель;

– щелкните по кнопке <ОК>;

– в появившемся окне выберите поля, которые будут присутствовать в форме. В данном примере присутствовать будут все поля, поэтому щелкните по кнопке ;
– щелкните по кнопке <Далее>;
– в появившемся окне уже выбран вид Форма в один столбец, поэтому щелкните по кнопке <Далее>;
– в появившемся окне выберите стиль оформления. Для этого щелкните по словам, обозначающим стили, либо перемещайте выделение стрелками вверх или вниз на клавиатуре. После выбора стиля щелкните по кнопке <Далее>;
– в появившемся окне задайте имя формы, набрав на клавиатуре параметр Состав преподавателей. Остальные параметры в окне оставьте без изменений;
– щелкните по кнопке <Готово>.

Перед вами откроется форма в один столбец. Столбец слева – это названия полей, столбец справа - данные первой записи (в нижней части окна в строке параметра Запись стоит цифра "1"). Для перемещения по записям надо щелкнуть по кнопке (в сторону записей с большими номерами) или (в сторону записей с меньшими номерами). Кнопка – это переход на первую запись, кнопка – переход на последнюю запись.

Для поиска преподавателя Миронова:
– переведите курсор в первую строку поля "Фамилия';
– выполните команду Правка, Найти;
– в появившемся окне в строке Образец введите фамилию Миронов;
– в строке параметра Просмотр должно быть слово ВСЕ (имеется в виду искать по всем записям);
– в строке параметра Совпадение выберите из списка параметр С любой частью поля;
– в строке параметра Только в текущем поле установите флажок (должна стоять "галочка");
– щелкните по кнопке <Найти>. Курсор перейдет на вторую запись и выделит слово Миронов;
– щелкните по кнопке <Найти далее>. Курсор перейдет на седьмую запись и также выделит слово Миронов;
– щелкните по кнопке <Закрыть> для выхода из режима поиска;

Для замены зарплаты ассистенту Сергеевой с 1000 р. на 1100 р.:
– переведите курсор в первую строку поля "Зарплата";
– выполните команду Правка, Заменить;
– в появившемся окне в строке параметра Образец введите 1000 р.;
– в строке параметра Заменить на введите 1100 р. Обратите внимание на остальные опции - вам надо вести поиск по всем записям данного поля;
– щелкните по кнопке <Найти далее>. Курсор перейдет на четвертую запись, но здесь не нужно менять данные, поэтому снова щелкните по

кнопке <Найти далее>. Курсор перейдет на девятую запись - это то, что нам надо;

– щелкните по кнопке <Заменить>. Данные будут изменены;

– щелкните по кнопке <Закрыть>.

4. Дня сортировки данных в поле "Год рождения" по убыванию:

– щелкните по любой записи поля "Год рождения";

– щелкните по кнопке на панели управления или выполните команду Записи, Сортировка, Сортировка по убыванию. Все данные в таблице будут отсортированы в соответствии с убыванием значений в поле "Год рождения".

5. Для фильтрации данных по полю "Должность":

– щелкните по записи Доцент поля "Должность";

– щелкните по кнопке или выполните команду Записи, Фильтр, Фильтр по выделенному. В форме останутся только записи о преподавателях - доцентах;

– щелкните по записи Информатика поля "Дисциплина";

– щелкните по кнопке или выполните команду Записи, Фильтр, Фильтр по выделенному. В форме останутся только записи о преподавателях - доцентах кафедры информатики;

– для отмены фильтра щелкните по кнопке на панели инструментов или выполните команду Записи, Удалить фильтр. В таблице появятся все данные.

6. Измените название поля "Дисциплина" на "Преподаваемая дисциплина". Дли этого:

– перейдите в режим конструктора, щелкнув по кнопке на панели инструментов или выполнив команду Вид, Конструктор;

– щелкните правой кнопкой мыши в поле "Дисциплина" (на названии поля - оно слева, а строка справа с именем Дисциплина - это ячейка для данных, свойства которых мы не будем менять). В появившемся меню выберите пункт Свойства. На экране откроется окно свойств для названия поля "Дисциплина";

– щелкните по строке с именем Подпись, т.е. там, где находится слово Дисциплина;

– сотрите слово "Дисциплина" и введите "Преподаваемая дисциплина";

– для просмотра результата перейдите в режим формы, выполнив команду Вид, Режим формы.

7.Для просмотра созданной формы:

– щелкните по кнопке или выполните команду Файл, Предварительный просмотр. Вы увидите форму как бы на листе бумаги;

– закройте окно просмотра.

Примечание. Не удивляйтесь полученному результату, так как на листе поместилось несколько страниц формы. Распечатывать форму не будем, потому

что основное назначение подобной формы – удобный построчный ввод и просмотр данных, а не сохранение данных в виде бумажного документа.

ЛАБОРАТОРНАЯ РАБОТА 3. НОРМАЛИЗАЦИЯ ДАТАЛОГИЧЕСКОЙ МОДЕЛИ МЕТОДОМ ДЕКОМПОЗИЦИИ

Рассмотрим в качестве примера проектирование и создание информационной системы для ввода данных о посещаемости хозрасчетной поликлиники пациентами. Данный пример носит исключительно учебный характер и ни в коем случае не претендует на роль образцовой информационной системы для данной предметной области - в реальной информационной системе подобного назначения число таблиц и набор полей таблиц должно быть существенно большим, да и автоматизированные рабочие места обычно выглядят несколько по-другому.

Целью данной информационной системы является хранение данных о посещениях пациентами поликлиники с целью дальнейшей статистической обработки, или, иными словами, регистрация статистических талонов, содержащих набор сведений, сходный с представленным на рис.6

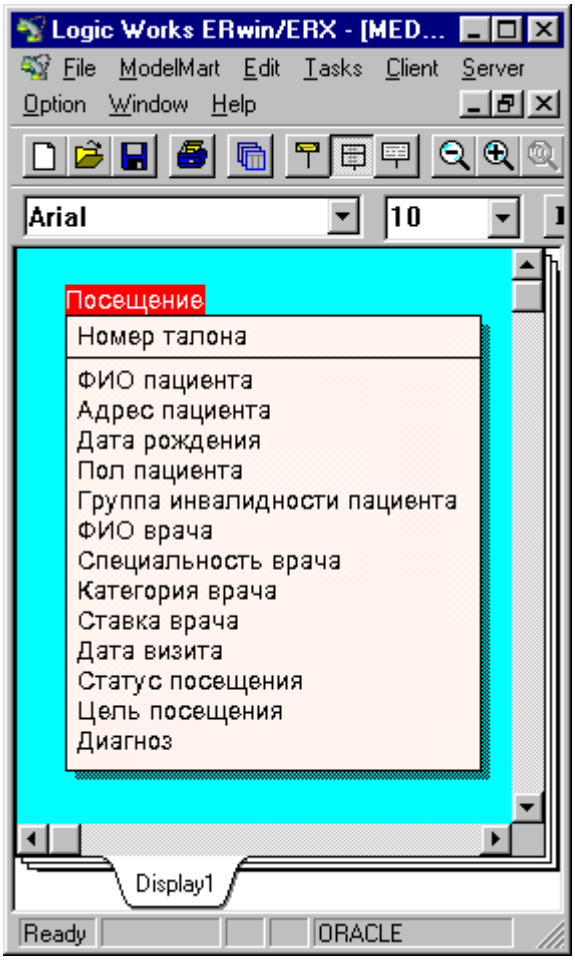

Рис.6. Примерное содержание статистического талона

Итак, центральной сущностью будущей модели данных является посещение врача пациентом. И врач, и пациент участвуют в посещении. В исходной сущности помимо атрибутов, идентифицирующих врача и пациента (которые, следовательно, могут повторяться) присутствуют атрибуты, характеризующие конкретный экземпляр данной сущности - дата посещения, цель посещения, статус (первичное или повторное это посещение), диагноз. При проектировании данных желательно создать также уникальный первичный ключ (номер талона). Это более удобно, чем использование комбинации первичных ключей сущностей "Врач" и "Пациент", так как визитов одного и того же пациента к одному и тому же врачу может быть несколько(рис.7).

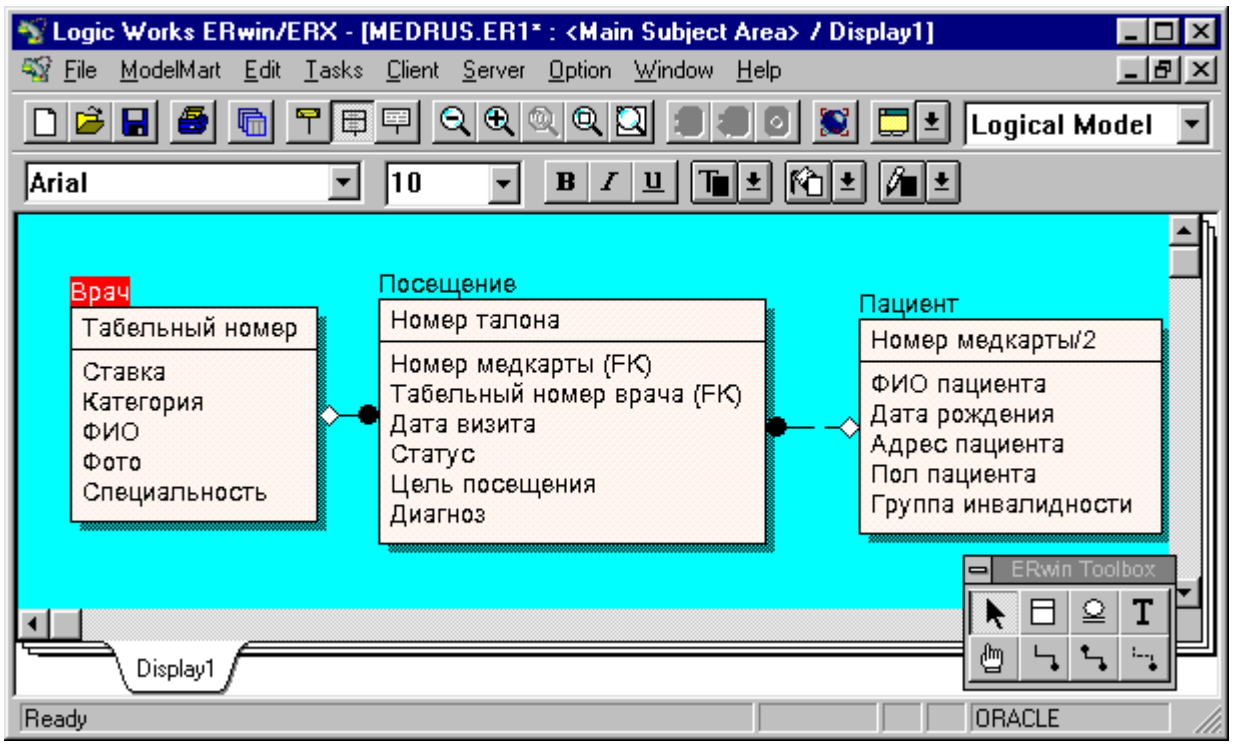

Рис.7. Первый вариант модели данных - выделяем сущности "Врач" и "Пациент"

Учитывая возможную повторяемость значений атрибутов, характеризующих диагноз, цель посещения, статус, создадим соответствующие дополнительные сущности, выполняющие роль справочников (вспомним о том, что модель данных должна удовлетворять первой нормальной форме).

Далее рассмотрим сущность "Врач". Очевидно, в поликлинике может быть несколько врачей, имеющих одну и ту же специальность, поэтому имеет смысл выделить понятие "специальность" в отдельную сущность и связать ее посредством внешнего ключа с сущностью "Врач". В результате получим модель данных, похожую на изображенную на рис. 8.

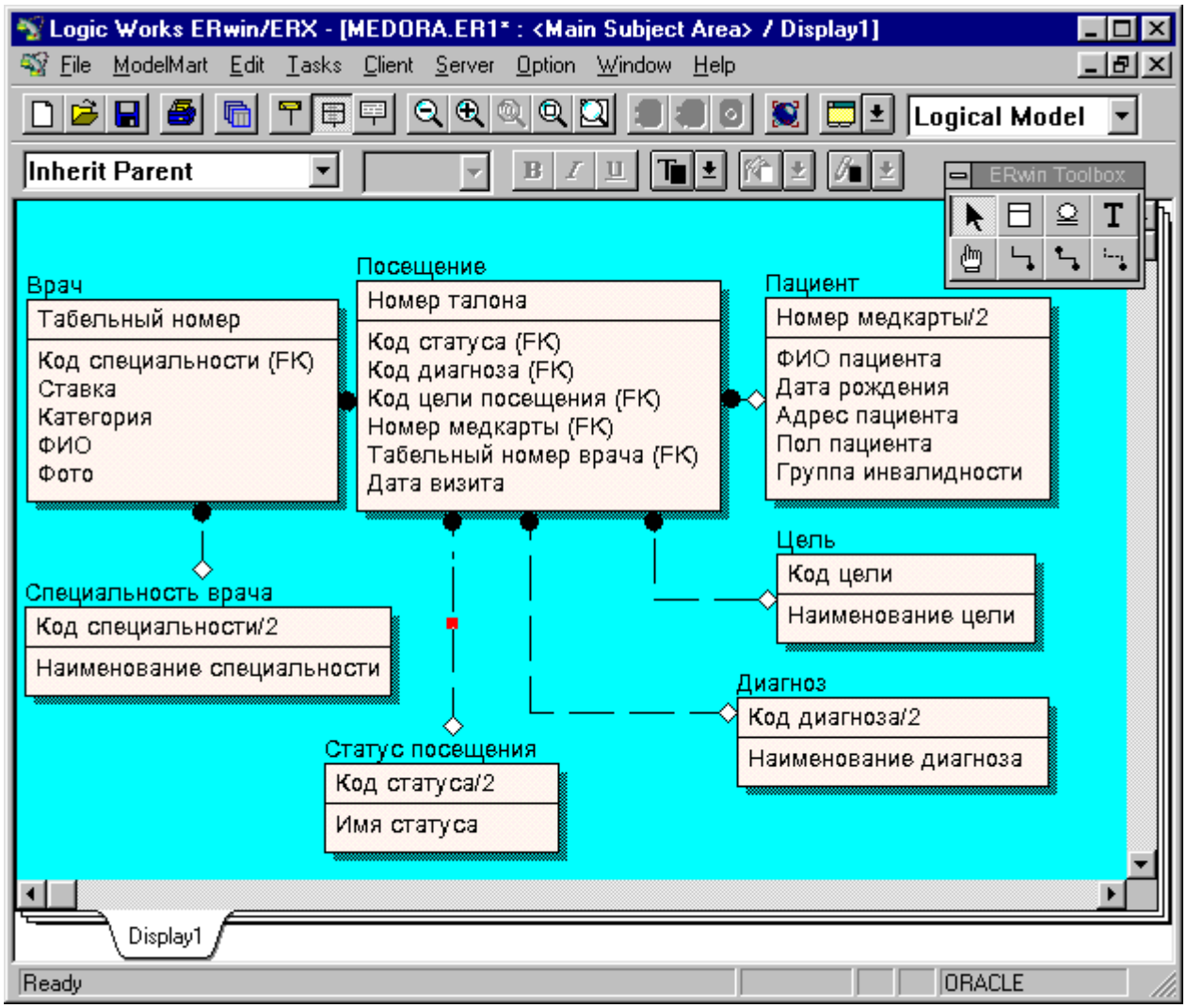

Рис.8. Модель данных после нормализации.

Возникает вопрос: а как быть с остальными атрибутами этой сущности? Ведь очевидно, что атрибуты "Ставка" и "Категория" тоже повторяются. Однако атрибут "Категория" - это обычно целое число, и поэтому нет смысла выносить его в отдельный справочник. Что касается атрибута "Ставка", характеризующего продолжительность рабочего времени, то это обычно дробное число (например, 1.25). В принципе можно было бы вынести его в отдельный справочник, но из соображений разумной достаточности в данном примере мы этого делать не будем (нормализуя модель, можно дойти до такого состояния, когда выполнение запросов к таблицам окажется весьма длительным из-за их большого числа).

Теперь рассмотрим сущность "Пациент". Исходя из требования атомарности атрибутов, следует обдумать, является ли таковым адрес пациента. Отметим, что ответ на этот вопрос неоднозначен. Если, например, речь идет о районной поликлинике с делением на участки, очевидно, что следует как минимум выделить в отдельные атрибут номер квартиры и остальную часть адреса (улица, дом, номер корпуса), так как эта остальная часть адреса указывает на принадлежность к определенному участку, а также, возможно, создать сущности "Участок" и "Часть участка" (со сведениями об улице, доме и

номере корпуса), связав их между собой и с сущностью "Пациент". С другой стороны, если речь идет о хозрасчетной или ведомственной поликлинике, где адрес может потребоваться главным образом для отправки по почте счетов за услуги, то вполне возможно, что в этом случае адрес может считаться атомарным атрибутом (если, конечно, в дальнейшем не потребуется какая-либо специфическая статистическая обработка данных, связанная с анализом распределения посещений по районам проживания пациентов). Кроме того, если речь идет о районной поликлинике, возможна повторяемость не только частей адресов, но и самих адресов (в случае, если поликлиника обслуживает нескольких жителей одной квартиры). В любом случае, проектируя данные, следует ставить подобные вопросы перед заказчиками и будущими пользователями информационной системы еще на этапе проектирования данных, так как исправления в модели данных гораздо более удобно делать до разработки автоматизированных рабочих мест, нежели исправлять вместе с моделью готовую базу данных и приложения.

Итак, мы создали логическую модель данных. До этого момента можно было думать о предметной области, не вдаваясь в подробности физической реализации разрабатываемой модели. Теперь же следует подумать о физическом проектировании данных, а именно о выборе СУБД, создании таблиц, соответствующих созданным сущностям, типах полей для хранения атрибутов, а также о создании других объектов базы данных, предназначенных для облегчения поиска в таблицах, контроля ссылочной целостности данных, обработки данных, таких, как индексы, триггеры, хранимые процедуры.

Как уже отмечалось в данном цикле статей, использование серверной СУБД значительно облегчает разработку информационных систем по сравнению с использованием плоских таблиц типа dBase или Paradox, так как в последнем случае реализация правил ссылочной целостности, а также механизмов авторизации пользователя и методов ограничения доступа к данным может быть реализована только в приложении (фактически разработчик вынужден "изобретать велосипед при наличии самолетов", создавая в своих приложениях код, уже реализованный в серверах баз данных). Поэтому для реализации данной информационной системы воспользуемся какой-либо серверной СУБД (например, Oracle).Для создания физической модели выберем соответствующую платформу в используемом CASE-средстве и опишем характеристики таблиц, соответствующих описанным в логической модели сущностям, и содержащихся в них полей, соответствующих их атрибутам (рис. 9)

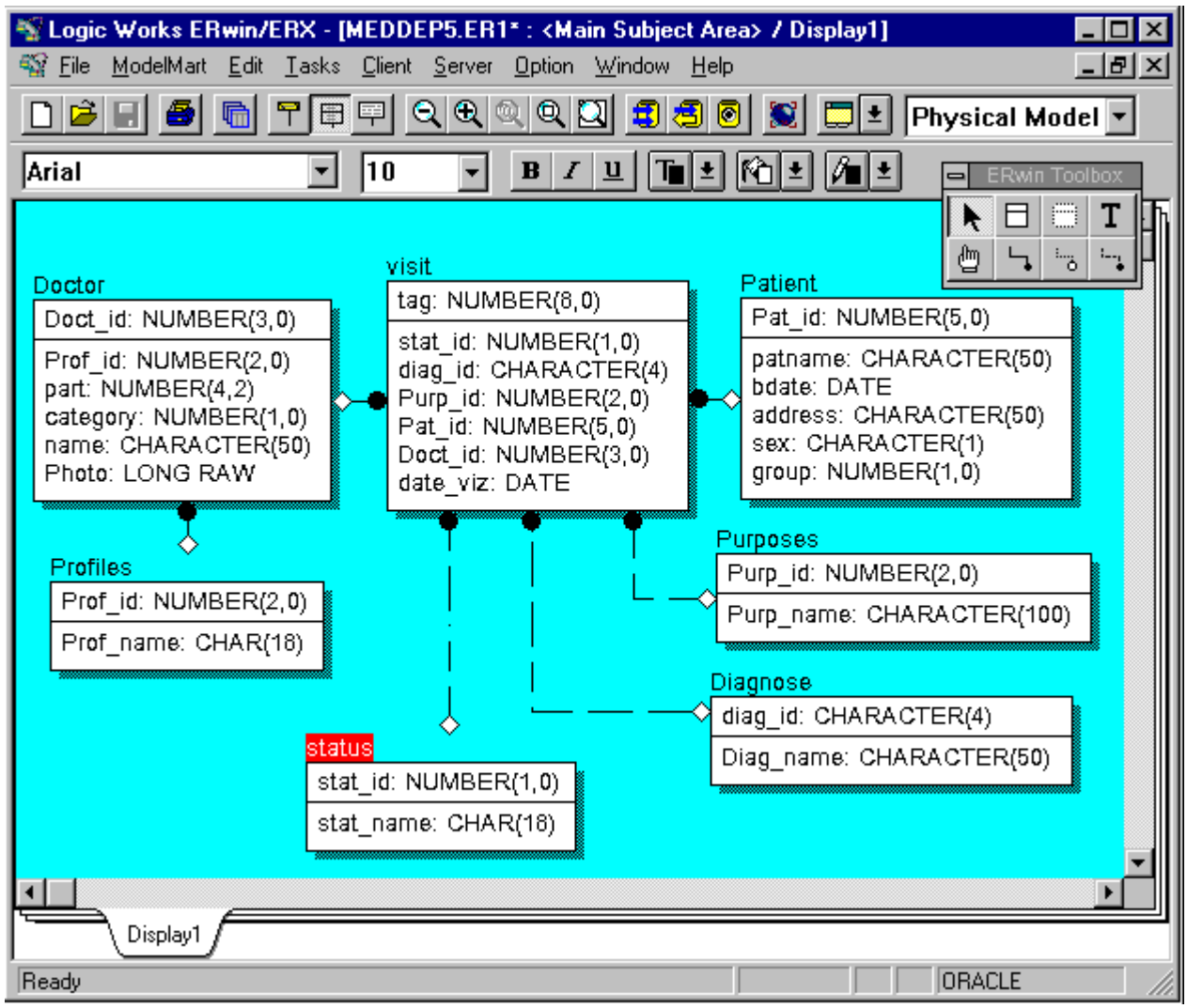

Рис.9. Физическая модель данных

Далее опишем свойства связей между таблицами, в частности, реакцию сервера на попытки нарушения ссылочной целостности со стороны клиентского приложения (рис. 10): При желании можно также отредактировать тексты сообщений триггеров сервера, передаваемых клиентскому приложению.

После этого можно создать на сервере схему базы данных, либо сгенерировать DDL-сценарий создания базы данных, представляющий собой набор предложений на процедурном расширении языка SQL данной серверной СУБД) с целью его последующего выполнения. В результате выполнения DDL-сценария на сервере должны быть созданы пустые таблицы, а также иные объекты базы данных, которые можно просмотреть с помощью утилиты Database Explorer (рис.11).

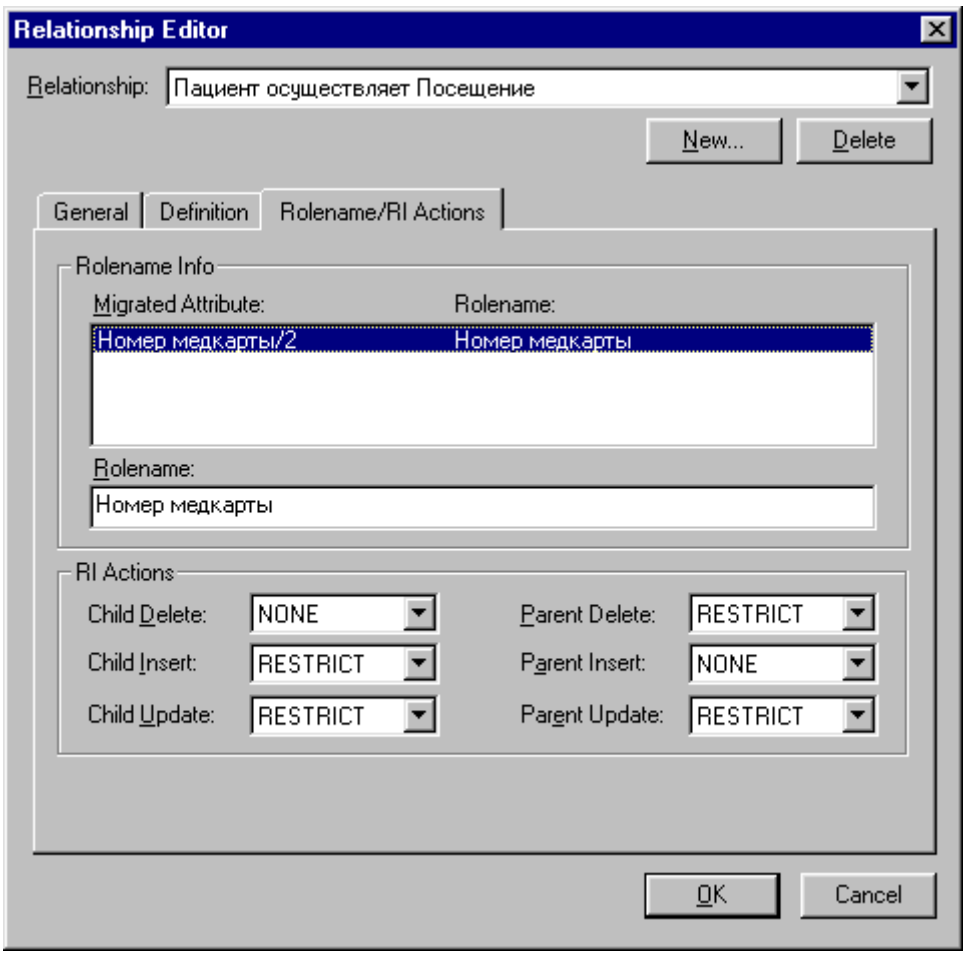

Рис. 10. Описание реакции сервера на попытки нарушения ссылочной целостности данных

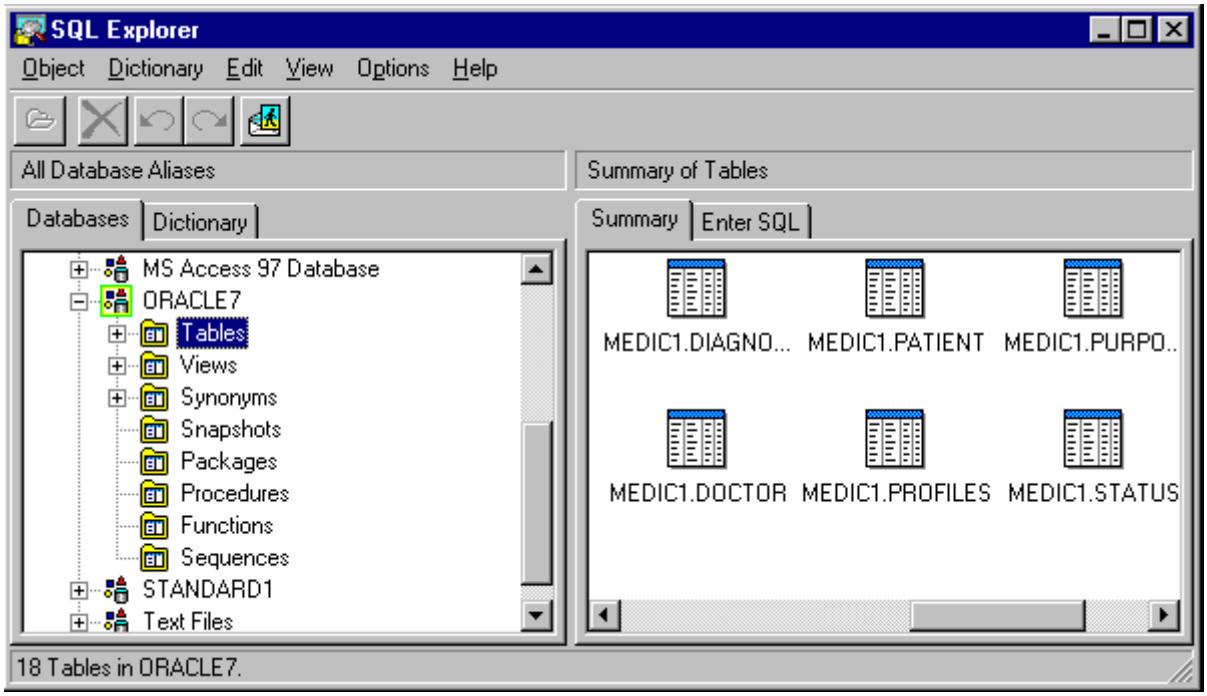

Рис.11. Созданные на сервере объекты базы данных

ЛАБОРАТОРНАЯ РАБОТА 4. СОЗДАНИЕ ЗАПРОСОВ К БАЗЕ ДАННЫХ.

ЗАДАНИЕ 1

Формирование запросов на выборку.

1. На основе таблицы Преподаватели создайте простой запрос на выборку, в котором должны отображаться фамилии, имена, отчества преподавателей и их должность.
2. Данные запроса отсортируйте по должностям.
3. Сохраните запрос.
4. Создайте запрос на выборку с параметром, в котором должны отображаться фамилии, имена, отчества преподавателей и преподаваемые ими дисциплины, а в качестве параметра задайте фамилию преподавателя и выполните этот запрос для преподавателя Гришина.

ТЕХНОЛОГИЯ РАБОТЫ

1. Для создания простого запроса:
– в окне базы данных откройте вкладку Запросы;
– в открывшемся окне щелкните по кнопке <Создать>;
– из появившихся пунктов окна "Новый запрос" выберите Простой запрос и щелкните по кнопке <ОК>;
– в появившемся окне в строке Таблицы/запросы выберите таблицу Преподаватели (если других таблиц или запросов не было создано, она будет одна в открывающемся списке);
– в окне "Доступные поля" переведите выделение на параметр Фамилия,
– щелкните по кнопке. Слово Фамилия перейдет в окно "Выбранные поля";
– аналогично в окно "Выбранные поля" переведите поля "Имя", "Отчество", "Должность" (порядок важен - в таком порядке данные и будут выводиться);
– щелкните по кнопке. <Далее>;
– в строке параметра Задайте имя запроса введите новое имя Должности преподавателей;
– щелкните по кнопке <Готово>. На экране появится таблица с результатами запроса.
2. Для сортировки данных:
– щелкните в любой строке поля "Должность";
– отсортируйте данные по убыванию. Для этого щелкните по кнопке  на панели инструментов или выполните команду Записи, Сортировка, Сортировка по убыванию.
3. Для сохранения запроса:
– щелкните по кнопке  или выполните команду Файл, Сохранить;

– закройте окно запроса.

4. Для создания запроса на выборку с параметром:

– создайте запрос на выборку для следующих полей таблицы Преподаватели: "Фамилия", "Имя", "Отчество", "Преподаваемая дисциплина". Запрос создавайте аналогично тому, как это делалось в п.1;

– задайте имя запросу Преподаваемые дисциплины;

– щелкните по кнопке <Готово>. На экране появится таблица с результатами запроса;

– перейдите в режиме конструктора, щелкнув по кнопке или выполнив команду Вид, Конструктор;

– в строке параметра Условия отбора для поля "Фамилия" введите фразу (скобки тоже вводить): [Введите фамилию преподавателя]

– выполните запрос, щелкнув по кнопке на панели инструментов или выполнив команду Запрос, Запуск.

Примечание. Вышеописанным способом запрос выполняется только в режиме конструктора. Для того чтобы выполнить запрос из другого режима, надо открыть вкладку Запросы, выделить требуемый запрос и щелкнуть по кнопке <Открыть>.

– в появившемся окне введите фамилию Гришин и щелкните по кнопке <ОК>. На экране появится таблица с данными о преподавателе Гришине - его имя, отчество и преподаваемая им дисциплина;

– сохраните запрос;

– закройте окно запроса.

ЗАДАНИЕ 2

На основе таблицы Преподаватели создайте отчет с группированием данных по должностям.

ТЕХНОЛОГИЯ РАБОТЫ

Для создания отчета:

– откройте вкладку Отчеты и щелкните по кнопке <Создать>;

– в открывшемся окне выберите пункт Мастер отчетов;

– щелкните по значку раскрывающегося списка в нижней части окна;

– выберите из появившегося списка таблицу Преподаватели;

– щелкните по кнопке <ОК>, В появившемся окне выберите поля, которые будут присутствовать в форме. В данном примере присутствовать будут все поля из таблицы, поэтому щелкните по кнопке

– щелкните по кнопке <Далее>;

– в появившемся окне присутствует перечень полей. Переведите выделение на поле "Должность";

– щелкните по кнопке. Таким образом вы задаете группировку данных по должности;

– щелкните по кнопке <Далее>;

– параметры появившегося окна оставим без изменений, поэтому щелкните по кнопке <Далее>;

– в появившемся окне выберите стиль оформления отчета;

– щелкните по кнопке. <Далее>;

– в появившемся окне введите название отчета Преподаватели;

– щелкните по кнопке <Готово>. На экране появится сформированный отчет;

– просмотрите, а затем закройте отчет.

Формирование сложных запросов

ЗАДАНИЕ 3

1.Разработайте запрос с параметрами о студентах заданной группы, в котором при вводе в окно параметров номера группы (в примере это 151 или 152) на экран должен выводиться состав этой группы.

2. Создайте запрос, в котором выводятся оценки студентов заданной группы по заданной дисциплине.

3. Создайте перекрестный запрос, в результате которого создастся выборка, отражающая средний балл по дисциплинам в группах.

4. Разработайте запрос на увеличение на 10% заработной платы тех преподавателей, кто получает менее 5000 руб.

5. Создайте запрос на удаление отчисленных студентов.

6. Разработайте запрос на создание базы данных отличников.

7. Для всех созданных вами запросов разработайте формы.

ТЕХНОЛОГИЯ РАБОТЫ

1. Для создания запроса с параметрами о студентах заданной группы:

– откройте вкладку Запросы',

– щелкните по кнопке <Создать>;

– в появившемся окне выберите Простои запрос и щелкните по кнопке <ОК>;

– в появившемся окне в строке Таблицы/запросы выберите из списка таблицу Студенты;

– перенесите все поля из окна "Доступные поля" в окно "Выбранные поля";

– щелкните по кнопке <Далее>. Выводить надо все поля, поэтому еще раз щелкните по кнопке <Далее>;

– в появившемся окне введите имя запроса Группа;

– щелкните по кнопке <Готово>. На экране появится таблица с данными запроса. Но вам надо, чтобы при выполнении запроса выяснялся номер группы. Для этого перейдите в режим конструктора;

– в строке Условия отбора для поля "Номер группы" введите фразу (скобки то же вводить): [Введите номер группы];

– выполните запрос, щелкнув по кнопке на панели инструментов, или выполните команду Запрос, Запуск;

– в появившемся окне введите 151 и щелкните по кнопке <ОК>. На экране появится таблица с данными о студентах 151-й группы;

– сохраните запрос и закройте таблицу запроса.

2. Для создания запроса, в котором выводятся оценки студентов заданной группы по заданной дисциплине:

– на вкладке Запросы щелкните по кнопке <Создать>;

– выберите Простой запрос и щелкните по кнопке <ОК>;

– выберите таблицу Студенты и перенесите поля "Фамилия", "Имя", "Отчество", "Номер группы" в окно "Выделенные поля" (выделяя нужное поле и щелкая по кнопке).

– Внимание! В дальнейшем под фразой В таблице ... выберите поле ... будем понимать выбор таблицы, выбор поля и перенос его в окно "Выделенные поля".

– в таблице Дисциплины выберите поле "Название дисциплины";

– в таблице Оценки выберите поле "Оценки". Вы сформировали шесть полей запроса - они связаны между собой посредством схемы данных;

– щелкните по кнопке <Далее>, затем в появившемся окне снова щелкните по кнопке <Далее>;

– в появившемся окне введите имя запроса Оценки группы, затем щелкните по ячейке Изменение структуры запроса (в ней должна появиться черная точка) - это позволит сразу перейти в режим конструктора;

– щелкните по кнопке <Готово>;

– в строке Условия отбора для поля "Номер группы" введите фразу: [Введите номер группы],

– в строке Условия отбора для поля "Название дисциплины" введите фразу: [Введите название дисциплины]

– выполните запрос;

– в первом появившемся окне введите 152, затем щелкните по кнопке <ОК>, во втором - введите Информатика и щелкните по кнопке <ОК>. На экране появится таблица со списком 152-й группы и оценками по информатике;

– сохраните запрос и закройте таблицу запроса.

– 3. Создайте перекрестный запрос о среднем балле в группах по дисциплинам. Но такой запрос строится на основе одной таблицы или одного запроса, в связи с чем надо сначала сформировать запрос, в

котором были бы поля "Номер группы", "Название дисциплины" и "Оценки". Для этого:

– на вкладке Запросы щелкните по кнопке <Создать>;
– выберите Простой запрос и щелкните по кнопке <ОК>;
– выберите из таблицы Студенты поле "Номер группы";
– выберите из таблицы Дисциплины поле "Название дисциплины" ;
– выберите из таблицы Оценки поле "Оценки";
– щелкните по кнопке <Далее>, затем в появившемся окне снова щелкните по кнопке <Далее>;
– в появившемся окне введите имя запроса Дисциплины оценки группы;
– щелкните по кнопке <Готово>;
– сохраните запрос и закройте таблицу запроса. Теперь можно создавать перекрестный запрос. Для этого:
– на вкладке Запросы щелкните по кнопке <Создать>;
– выберите Перекрестный запрос и щелкните по кнопке <ОК>;
– щелкните по ячейке Запросы, выберите Дисциплины оценки группы и щелкните по кнопке <Далее>;
– выберите поле "Название дисциплины" и щелкните по кнопке <Далее>;
– выберите поле "Номер группы" и щелкните по кнопке <Далее>;
– выберите функцию AVG, т.е. среднее (она по умолчанию уже выделена), и щелкните по кнопке <Далее>;
– введите название запроса Средние оценки и щелкните по кнопке <Готово>. Откроется таблица перекрестного запроса. Обратите внимание на то, что Access создает еще итоговое значение средних оценок по дисциплинам;
– закройте таблицу запроса.

4. Для создания запроса на изменение заработной платы преподавателей:

– на вкладке Запросы щелкните по кнопке <Создать>;
– выберите Простой запрос;
– в таблице Преподаватели выберите поле <Зарплата>;
– щелкните по кнопке <Далее>, затем в появившемся окне снова щелкните по кнопке <Далее>;
– в появившемся окне введите имя запроса Изменение зарплаты;
– щелкните по ячейке Изменение структуры запроса;
– щелкните по кнопке <Готово>;
– в строке Условия отбора введите <1>5000;
– откройте пункт меню Запрос и выберите Обновление;
– в строке конструктора запроса Обновление в поле "Зарплата" введите:[Зарплата]* 1,1;
– выполните запрос, подтвердив готовность на обновление данных;
– закройте запрос, подтвердив его сохранение;
– откройте форму Преподаватели;

– просмотрите изменение заработной платы у преподавателей, получающих меньше 5000 р.;
– закройте форму.

5. Для создания запроса на отчисление студента гр. 152 Перлова Кирилла Николаевича:

– на вкладке Запросы щелкните по кнопке <Создать>;
– выберите Простой запрос;,
– в таблице Студенты выберите поля "Фамилия", "Имя", "Отчество", "Номер группы";
– щелкните по кнопке <Далее>, затем в появившемся окне снова щелкните по кнопке <Далее>;
– в появившемся окне введите имя запроса Отчисленные студенты;
– щелкните по ячейке Изменение структуры запроса;
– щелкните по кнопке <Готово>;
– в строке Условия отбора введите; в поле "Фамилия" - Перлов, в поле "Имя" - Кирилл, в поле "Отчество" - Николаевич, в поле "Номер группы" - 152;
– откройте пункт меню Запрос и выберите Удаление;
– просмотрите удаляемую запись, щелкнув по кнопке или выполнив команду Вид, Режим таблицы; если отчисляемый студент выбран правильно, то перейдите в режим конструктора и выполните запрос. Если условия отбора сделаны неправильно, измените их;
– закройте запрос;
– откройте форму Студенты и удостоверьтесь в удалении записи о студенте Перлове;
– закройте форму.

6. Для создания запроса на создание базы данных отличников:

– на вкладке Запросы щелкните по кнопке <Создать>;
– выберите Простой запрос;
– в таблице Студенты выберите поля "Фамилия", "Имя", "Отчество" и "Номер группы", а в таблице Оценки - поле "Оценки";
– щелкните по кнопке <Далее>, затем в появившемся окне вновь щелкните по кнопке <Далее>;
– в появившемся окне введите имя запроса Отличники;
– щелкните по ячейке Изменение структуры запроса;
– щелкните по кнопке <Готово>.

Примечание.Для создания этого запроса надо воспользоваться операцией группировки. Будем считать отличниками тех студентов, которые набрали за четыре экзамена 20 баллов. Операция группировки позволит просуммировать оценки студентов по всем экзаменационным дисциплинам.

- для выполнения групповых операции щелкните на панели инструментов по кнопке или выполните команду Вид, Групповые операции;
- в строке Групповые операции поля "Оценки" щелкните по ячейке Групповые операции. Откройте раскрывающийся список и выберите функцию SUM;
- в строке Условия отбора поля "Оценки" введите 20;
- просмотрите создаваемую базу, щелкнув по кнопке или выполнив команду Вид, Режим таблицы;
- перейдите в режим конструктора;
- выполните команду Запрос, Создание таблицы;
- введите имя таблицы Студенты-отличники и щелкните по кнопке <ОК>;
- подтвердите создание таблицы;
- закройте с сохранением запрос;
- откройте вкладку Таблицы;
- откройте таблицу Студенты-отличники. Удостоверьтесь в правильности создания таблицы. Закройте таблицу.

7. Для каждого из созданных запросов создайте форму (можно рекомендовать автоформу в столбец или ленточную автоформу) для удобного просмотра данных. При создании этих форм воспользуйтесь рекомендациями в работе 3.

ЛАБОРАТОРНАЯ РАБОТА 5. СОЗДАНИЕ СЛОЖНЫХ ФОРМ

ЗАДАНИЕ 1

1. Разработайте сложную форму, в которой с названиями дисциплин была бы связана подчиненная форма Студенты и подчиненная форма Оценки студентов.

2. Измените расположение элементов в форме в соответствии с рис. 12.

3. Вставьте в форму диаграмму, графически отражающую оценки студентов.

4. Отредактируйте вид осей диаграммы.

ТЕХНОЛОГИЯ РАБОТЫ

1. Для создания сложной формы:

- на вкладке Формы щелкните по кнопке <Создать>;
- выберите Мастер форм и, не выбирая таблицу или запрос, щелкните по кнопке <ОК>;
- в таблице Дисциплины выберите поле "Название дисциплины";
- в таблице Студенты выберите поля "Код студента", "Фамилия", "Имя", "Отчество", "Номер группы";

– в таблице Оценки выберите поле "Оценки" и щелкните по кнопке <Далее>;

– в появившемся окне вариант построения формы нас удовлетворяет, поэтому щелкните по кнопке <Далее>;

– оставьте табличный вариант подчиненной формы и щелкните по кнопке <Далее>;

– выберите нужный вам стиль оформления формы и щелкните по кнопке <Далее>;

– введите название формы Дисциплины и оценки,

– щелкните по кнопке <Готово> и просмотрите полученную форму.

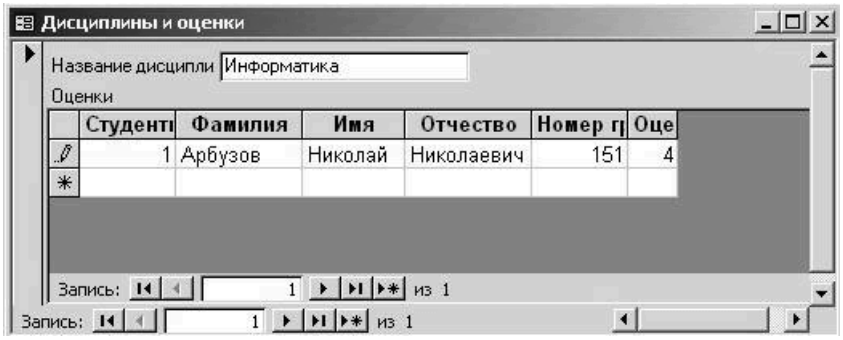

Рис.12. Форма Дисциплины и оценки

2. Нас не удовлетворяет расположение полей на экране. Измените их в соответствии с рис. 6, оставив место для диаграммы.

Для этого:

– перейдите в режим конструктора;

– стандартными средствами Windows (технология drag-and-drop) измените размеры подчиненной формы так, чтобы были видны все данные. Для этого надо (как правило, многократно) переключаться из режима конструктора в режим формы, смотреть на полученный результат и, если он не подходит, снова корректировать в режиме конструктора. Ширину столбцов в подчиненной форме можно изменить только в режиме формы.

3. Для того чтобы вставить в форму диаграмму оценок студентов по заданным дисциплинам, необходимо:

– переключиться в режим конструктора;

– выполнить команду Вид, Панель элементов;

– на этой панели щелкнуть по кнопке <Аа>;

– создать прямоугольник для надписи - заголовка диаграммы. Для этого переведите курсор в левый верхний угол будущего прямоугольника, нажмите левую кнопку мыши и, не отпуская ее, доведите до правого нижнего угла, затем отпустите кнопку;

– ввести надпись Диаграмма оценок;

– выполнить команду Вставка, Диаграмма;

– на свободном месте формы растянуть прямоугольник для диаграммы (нажмите левую кнопку мыши в левом верхнем углу и, не отпуская ее, растяните прямоугольник до правого нижнего угла, затем отпустите кнопку);
– выбрать таблицу Оценки и щелкнуть по кнопке <Далее>;
– выбрать поля "Код студента" и "Оценки";
– щелкнуть по кнопке <Далее>;
– выбрать вид диаграммы Гистограмма (по умолчанию он и стоит) и щелкнуть по кнопке <Далее>;
– дважды щелкнуть по надписи Сумма_оценки, выбрать Отсутствует и щелкнуть по кнопке <ОК>;
– щелкнуть по кнопке <Далее>;
– вновь щелкнуть по кнопке <Далее>, так как в строке Поля формы и в строке Поля диаграммы по умолчанию находится Код дисциплины (что нам и нужно);
– стереть название диаграммы Оценки (так как мы уже задали надпись для диаграммы) и щелкнуть по кнопке <Далее>.
4. Отредактируйте вид осей диаграммы. Для этого:
– дважды щелкните по диаграмме;
– дважды щелкните по значениям вертикальной оси;
– выберите вкладку Шкала;
– уберите "галочку" у надписи Минимальное значение, а в ячейке справа от этого названия введите 1
– уберите "галочку" у надписи Максимальное значение, а в ячейке справа от этого названия введите 5
– уберите "галочку" у надписи Цена основных делении, а в ячейке справа от этого названия введите 1 и щелкните по кнопке <ОК>;
– расширьте область диаграммы, перетащив правую границу окна диаграммы несколько правее (подведя курсор к правой границе до появления двойной стрелки и нажав левую кнопку мыши);
– закройте окно "Microsoft Graph", выбрав в меню Файл пункт Выход и возврат в дисциплины и оценки: форма.
– перейдите в режим формы (она представлена на рис. 13) и просмотрите форму для разных дисциплин (щелкая по кнопке перехода к следующей записи в нижней части формы). Вы увидите изменение названий дисциплин, а также оценок студентов по этим дисциплинам и изменение диаграмм, отображающих эти оценки;
– закройте форму.

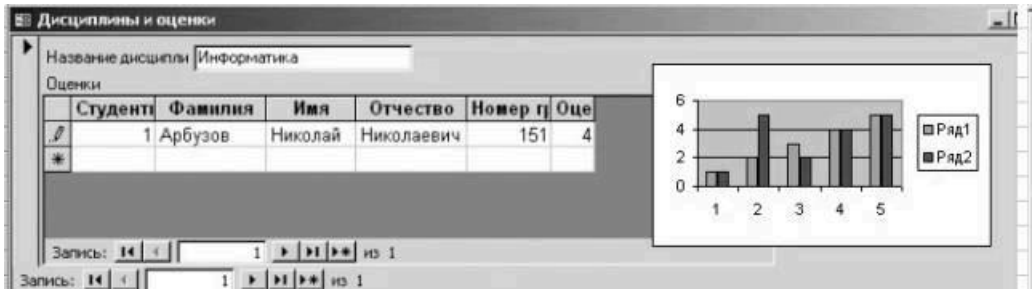

Рис. 13. Форма Дисциплины и оценки с включенной в нее диаграммой

ЗАДАНИЕ 2

Создание сложных отчетов.

1. Создайте запрос, на основе которого будет формироваться отчет. В запросе должны присутствовать: из таблицы Студенты - поля "Фамилия", "Имя", "Отчество" и "Номер группы", из таблицы Дисциплины - поле "Название дисциплины", из таблицы Оценки - поле "Оценки".

2. Создайте отчет по итогам сессии. В отчете оценки студентов должны быть сгруппированы по номерам групп и дисциплинам. Для каждого студента должна вычисляться средняя оценка в сессию, а для каждой группы - среднее значение оценок по всем предметам.

ТЕХНОЛОГИЯ РАБОТЫ

1. Для создания запроса:

– на вкладке Запросы щелкните по кнопке <Создать>;

– выберите Простой запрос и щелкните по кнопке <ОК>;

– из таблицы Студенты выберите поля "Фамилия", "Имя", "Отчество" и "Номер группы", из таблицы Дисциплины - поле "Название дисциплины", из таблицы Оценки - поле "Оценки" и щелкните по кнопке <Далее>;

– щелкните еще раз по кнопке <Далее>;

– введите название запроса Сессия и щелкните по кнопке <Готово>;

– закройте запрос.

2. Для создания итогового отчета выполните следующее:

на вкладке Отчеты щелкните по кнопке <Создать>;

– выберите Мастер отчетов, из раскрывающегося списка - запрос Сессия и щелкните по кнопке <ОК>;

– выберите все поля запроса и щелкните по кнопке <Далее>;

– тип представления данных нас удовлетворяет, поэтому щелкните по кнопке <Далее>;

– добавьте уровень группировки по номеру группы, выбрав в левом окне Номер группы и перенеся его в правое окно, щелкнув по кнопке ;

– щелкните по кнопке <Далее>;

– щелкните по кнопке <Итоги>, так как надо вычислять средний балл;

– поставьте "галочку" в ячейке поля "AVG" (эта функция вычисляет среднее) и щелкните по кнопке <ОК>;

– щелкните по кнопке <ОК>, так как сортировка не требуется, потому что данными являются название дисциплины и оценки, порядок которых не столь важен;

– выберите макет отчета. Рекомендуем ступенчатый, так как он занимает меньше места и в нем наглядно представлены данные (хотя это дело вкуса). Щелкните по кнопке <Далее>; выберите стиль отчета и щелкните по кнопке <Далее>;

– введите название отчета Итоги сессии и щелкните по кнопке <Готово>. На экране появится отчет. Его можно просмотреть, изменяя масштаб (щелкнув по листу) и перелистывая страницы (в нижней части экрана). Его можно также распечатать, выполнив команду Файл, Печать. После завершения необходимых вам операций закройте окно просмотра отчета.

ЗАДАНИЕ 3

Разработайте кнопочную форму-меню для работы с базами данных, в которой должны быть созданные вами формы и отчет.

ТЕХНОЛОГИЯ РАБОТЫ

Для создания кнопочного меню выполните следующие действия:

– выполните команду Сервис, Надстройки, Диспетчер кнопочных форм;

– подтвердите создание кнопочной формы, щелкнув по кнопке <Да>;

– Access предложит вам работать с главной кнопочной формой или создать дополнительно новую. Создайте свою форму, щелкнув по кнопке <Создать>;

– введите имя Меню и щелкните по кнопке <ОК>;

– в окне выберите Меню и щелкните по кнопке <Изменить>;

– создайте элементы данной кнопочной формы, щелкнув по кнопке <Создать>;

– в строке Текст введите поясняющую надпись к первой создаваемой кнопке Преподаватели,

– в строке Команда выберите из списка Открытие формы в режиме редактирования;

– Примечание. Диспетчер напрямую может связать кнопку с открытием формы или отчета. Чтобы открыть таблицу или запрос, надо создать соответствующий макрос и указать это в диспетчере.

– в строке Форма выберите из списка форму Преподаватели и щелкните по кнопке <ОК>; введите в меню все созданные формы и отчет, повторяя п. 6 - 9;

– закройте окно кнопочной формы, щелкнув по кнопке <Закрыть>;

– щелкните по кнопке <По умолчанию>;

– закройте диспетчер кнопочных форм, щелкнув по кнопке <Закрыть>;

– на вкладке Формы подведите курсор мыши к надписи Кнопочная форма, щелкните правой кнопкой мыши, выберите пункт Переименовать и введите новое имя Форма меню, затем нажмите клавишу <Enter>;

– откройте эту форму и просмотрите возможности открытия форм и отчета из меню.

Примечание. Для возврата из любой открытой формы или отчета в меню достаточно закрыть их.

ЛАБОРАТОРНАЯ РАБОТА 6. ИССЛЕДОВАНИЕ ОПЕРАТОРА SELECT ЯЗЫКА SQL

Лабораторная работа 6 выполняется для учебной базы данных, включающей шесть таблиц:

STUDENT (Студент)

STUDENT_ID	SURNAME	NAME	STIPEND	KURS	CITY	BIRTHDAY	UNIV_ID
1	Иванов	Иван	150	1	Орел	3/12/1982	10
3	Петров	Петр	200	3	Курск	1/12/1980	10
6	Сидоров	Вадим	150	4	Москва	7/06/1979	22
10	Кузнецов	Борис	0	2	Брянск	8/12/1981	10
12	Зайцева	Ольга	250	2	Липецк	1/05/1981	10
265	Павлов	Андрей	0	3	Воронеж	5/11/1979	10
32	Котов	Павел	150	5	Белгород	NULL	14
654	Лукин	Артем	200	3	Воронеж	1/12/1981	10
276	Петров	Антон	200	4	NULL	5/08/1981	22
55	Белкин	Вадим	250	5	Воронеж	7/01/1980	10
.....	.....	.....	.....	.....	.....	.....	.....

STUDENT_ID — числовой код, идентифицирующий студента,
SURNAME — фамилия студента,
NAME — имя студента,
STIPEND — стипендия, которую получает студент,
KURS — курс, на котором учится студент,
CITY — город, в котором живет студент,
BIRTHDAY — дата рождения студента,
UNIV_ID — числовой код, идентифицирующий университет, в котором учится студент.

SUBJ_LECT (Учебные дисциплины преподавателей)

LECTURER _ID	SUBJ_ID
24	24
46	46
74	74
108	108
276	276
328	328
. . . . .	. . . . .

LECTURER_ID — идентификатор преподавателя,
SUBJ_ID — идентификатор предмета обучения.

LECTURER (Преподаватель)

LECTURER_ID	SURNAME	NAME	CITY	UNIV_ID
24	Колесников	Борис	Воронеж	10
46	Никонов	Иван	Воронеж	10
74	Лагутин	Павел	Москва	22
108	Струков	Николай	Москва	22
276	Николаев	Виктор	Воронеж	10
328	Сорокин	Андрей	Орел	10
. . . . .	. . . . .	. . . . .	. . . . .	. . . . .

LECTURER_ID — числовой код, идентифицирующий преподавателя,

SURNAME — фамилия преподавателя,

NAME — имя преподавателя,

CITY — город, в котором живет преподаватель,

UNIV_ID — идентификатор университета, в котором работает преподаватель.

SUBJECT (Предмет обучения)

SUBJ_ID	SUBJ_NAME	HOUR	SEMESTER
10	Информатика	56	1
22	Физика	34	1
43	Математика	56	2
56	История	34	4
94	Английский	56	3
73	Физкультура	34	5
. . . . .	. . . . .	. . . . .	. . . . .

SUBJ_ID — идентификатор предмета обучения,

SUBJ_NAME — наименование предмета обучения,

HOUR — количество часов, отводимых на изучение предмета,

SEMESTER — семестр, в котором изучается данный предмет.

UNIVERSITY (Университеты)

UNIV_ID	UNIV_NAME	RATING	CITY
22	МГУ	606	Москва
10	ВГУ	296	Воронеж
11	НГУ	345	Новосибирск
32	РГУ	416	Ростов
14	БГУ	326	Белгород
15	ТГУ	368	Томск
18	ВГМА	327	Воронеж
.....	.....	.....	.....

UNIV_ID — идентификатор университета,
UNIV_NAME — название университета,
RATING — рейтинг университета,
CITY — город, в котором расположен университет.

EXAM_MARKS (Экзаменационные оценки)

EXAM_ID	STUDENT_ID	SUBJ_ID	MARK	EXAM_DATE
145	12	10	5	12/01/2000
34	32	10	4	23/01/2000
75	55	10	5	05/01/2000
238	12	22	3	17/06/1999
639	55	22	**NULL**	22/06/1999
43	6	22	4	18/01/2000
.....	.....	.....	.....	.....

EXAM_ID — идентификатор экзамена,
STUDENT_ID — идентификатор студента,
SUBJ_ID — идентификатор предмета обучения,
MARK — экзаменационная оценка,
EXAM_DATE — дата экзамена.

Простейшие SELECT-запросы

Оператор **SELECT** (выбрать) языка SQL является самым важным и самым часто используемым оператором. Он предназначен для *выборки* информации из таблиц базы данных. Упрощенный синтаксис оператора **SELECT** выглядит следующим образом.

SELECT [DISTINCT] <список атрибутов>
FROM <список таблиц> [WHERE <условие выборки>]
[ORDER BY <список атрибутов>]
[GROUP BY <список атрибутов>]
[HAVING <условие>]
[UNION <выражение с оператором SELECT>] ;

В квадратных скобках указаны элементы, которые могут отсутствовать в запросе.

Ключевое слово SELECT сообщает базе данных, что данное предложение является запросом *на извлечение* информации. После слова SELECT через запятую перечисляются *наименования полей* (список атрибутов), содержимое которых запрашивается.

Обязательным ключевым словом в предложении-запросе **SELECT** является слово **FROM** (из). За ключевым словом **FROM** указывается список разделенных запятыми имен таблиц, из которых извлекается информация.

Например,

SELECT NAME, SURNAME
FROM STUDENT;

Любой SQL-запрос должен заканчиваться символом «;»

Приведенный запрос осуществляет выборку всех значений полей NAME И SURNAME из таблицы STUDENT.
Его результатом является таблица следующего вида:

NAME	SURNAME
Иван	Иванов
Петр	Петров
Вадим	Сидоров
Борис	Кузнецов
Ольга	Зайцева
Андрей	Павлов
Павел	Котов
Артем	Лукин
Антон	Петров
Вадим	Белкин

Порядок следования столбцов в этой таблице соответствует порядку полей name и surname, указанному в запросе, а не их порядку во входной таблице STUDENT.

Если необходимо вывести значения *всех* столбцов таблицы, то можно вместо перечисления их имен использовать символ «*» (звездочка).

SELECT *
FROM STUDENT;

В данном случае результатом выполнения запроса будет вся таблица STUDENT.

Еще раз обратим внимание на то, что получаемые в результате SQL-запроса таблицы не в полной мере отвечают определению реляционного отношения. В частности, в них могут оказаться кортежи (строки) с одинаковыми значениями атрибутов.

Например, запрос «Получить список названий городов, где проживают студенты, сведения о которых находятся в таблице student», можно записать в следующем виде.

SELECT CITY **FROM** STUDENT;

Его результатом будет таблица:

CITY
Орел
Курск
Москва
Брянск
Липецк
Воронеж
Белгород
Воронеж
NULL
Воронеж
…

Видно, что в таблице встречаются одинаковые строки (выделены жирным шрифтом).

Для исключения из результата SELECT-запроса повторяющихся записей используется ключевое слово **DISTINCT** (отличный). Если запрос **SELECT** извлекает множество полей, то **DISTINCT** *исключает* дубликаты строк, в которых значения *всех* выбранных полей идентичны.

Предыдущий запрос можно записать в следующем виде.

SELECT DISTINCT CITY
FROM STUDENT;

В результате получим таблицу, в которой дубликаты строк исключены.

CITY
Орел
Курск
Москва
Брянск
Липецк
Воронеж
Белгород
NULL
…

Ключевое слово **ALL** (все), в отличие от DISTINCT**,** оказывает противоположное действие, то есть при его использовании повторяющиеся строки *включаются* в состав выходных данных. Режим, задаваемый ключевым словом **ALL,** действует по умолчанию, поэтому в реальных запросах для этих целей оно практически не используется.

Использование в операторе **SELECT** предложения, определяемого ключевым словом **where** (где), позволяет задавать выражение условия (предикат), принимающее значение *истина* или *ложь для* значений полей строк таблиц, к которым обращается оператор **select.** Предложение **where** определяет, ***какие*** *строки указанных таблиц должны быть выбраны*. В таблицу, являющуюся результатом запроса, включаются только те строки, для которых условие (предикат), указанное в предложении **WHERE,** принимает значение *истина.*

Пример

Написать запрос, выполняющий выборку имен (NAME) всех студентов с фамилией (SURNAME) Петров, сведения о которых находятся в таблице student.

```
SELECT SURNAME, NAME
 FROM STUDENT
WHERE SURNAME = 'Петров';
```

Результатом этого запроса будет таблица:

SURNAME	NAME
Петров	Петр
Петров	Антон

В задаваемых в предложении **WHERE** условиях могут использоваться операции сравнения, определяемые операторами = (равно), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), <> (не равно), а также логические операторы

AND, OR И NOT.

Например, запрос для получения *имен* и *фамилий* студентов, обучающихся на *третьем* курсе и получающих стипендию (размер стипендии *больше нуля),* будет выглядеть таким образом:

```
SELECT NAME, SURNAME
FROM STUDENT
WHERE KURS = 3 AND STIPEND > 0;
```

Результат выполнения этого запроса имеет вид:

SURNAME	NAME
Петров	Петр
Лукин	Артем

Упражнения

1. Напишите запрос для вывода идентификатора (номера) предмета обучения, его наименования, семестра, в котором он читается, и количества отводимых на этот предмет часов для всех строк таблицы SUBJECT.
2. Напишите запрос, позволяющий вывести все строки таблицы EXAM_MARKS, в которых предмет обучения имеет номер (SUBJ_ID), равный 12.
3. Напишите запрос, выбирающий все данные из таблицы STUDENT, расположив столбцы таблицы в следующем порядке: KURS, SURNAME, NAME, STIPEND.
4. Напишите запрос SELECT, который выводит наименование предмета обучения (SUB J_NAME) и количество часов (HOUR) для каждого предмета (SUBJECT) в 4-м семестре (SEMESTER).
5. Напишите запрос, позволяющий получить из таблицы exam_marks значения столбца mark (экзаменационная оценка) для всех студентов, исключив из списка повторение одинаковых строк.
6. Напишите запрос, который выводит список фамилий студентов, обучающихся на третьем и последующих курсах.
7. Напишите запрос, выбирающий данные о фамилии, имени и номере курса для студентов, получающих стипендию больше 140.
8. Напишите запрос, выполняющий выборку из таблицы SUBJECT названий всех предметов обучения, на которые отводится более 30часов.
9. Напишите запрос, который выполняет вывод списка университетов, рейтинг которых превышает 300 баллов.
10. Напишите запрос к таблице student для вывода списка фамилий (SURNAME), имен (NAME) и номера курса (KURS) всех студентов со стипендией, большей или равной 100, и живущих в Воронеже.
11. Какие данные будут получены в результате выполнения запроса?

```
SELECT *
FROM STUDENT
WHERE  (STIPEND <  100 OR
```

NOT (BIRTHDAY >= '10/03/1980' **AND** STUDENT_ID > 1003**));**

12. Какие данные будут получены в результате выполнения запроса?

SELECT *
FROM STUDENT
WHERE NOT ((BIRTHDAY = '10/03/1980' **OR** STIPEND > 100**)**
AND STUDENT_ID >= **1003);**

Операторы IN, BETWEEN, LIKE, IS NULL

При задании логического условия в предложении WHERE могут быть использованы операторы IN, BETWEEN, LIKE, IS NULL.

Операторы IN (равен любому из списка) и NOT IN (не равен ни одному из списка) используются для сравнения проверяемого значения поля с заданным списком. Этот список значений указывается в скобках справа от оператора IN.

Построенный с использованием IN предикат (условие) считается истинным, если значение поля, имя которого указано слева от IN, *совпадает* (подразумевается точное совпадение) с одним из значений, перечисленных в списке, указанном в скобках справа от IN.

Предикат, построенный с использованием NOT IN, считается истинным, если значение поля, имя которого указано слева от not IN, *не совпадает* ни с одним из значений, перечисленных в списке, указанном в скобках справа от NOT IN.

Примеры

Получить из таблицы exam_marks сведения о студентах, *имеющих* экзаменационные оценки только 4 и 5.

SELECT *
FROM EXAM_MARKS
WHERE MARK IN (4, 5);

Получить сведения о студентах, *не имеющих* ни одной экзаменационной оценки, равной 4 и 5.

SELECT *
FROM EXAM_MARKS
WHERE MARK **NOT IN (4, 5**);

Оператор BETWEEN используется для проверки условия вхождения значения поля в заданный интервал, то есть вместо списка значений атрибута этот оператор задает границы его изменения.

Например, запрос на вывод записей о предметах, на изучение которых отводится количество часов, находящееся в пределах между 30 и 40, имеет вид:

SELECT *
FROM SUBJECT
WHERE HOUR **BETWEEN** 30 **AND** 40**;**

Граничные значения, в данном случае значения 30 и 40, *входят* во множество значений, с которыми производится сравнение. Оператор BETWEEN может использоваться как для числовых, так и для символьных типов полей.

Оператор LIKE применим только к символьным полям типа CHAR или VARCHAR.

Этот оператор просматривает строковые значения полей с целью определения, входит ли заданная в операторе LIKE подстрока (образец поиска) в символьную строку-значение проверяемого поля.

Для выборки строковых значений по заданному образцу подстроки можно применять шаблон искомого образца строки, использующий следующие символы:

- символ подчеркивания «_», указанный в шаблоне, определяет возможность наличия в указанном месте *одного любого* символа;
- символ «%» допускает присутствие в указанном месте проверяемой строки последовательности любых символов произвольной длины.

Пример

Написать запрос, выбирающий из таблицы student сведения о студентах, фамилии которых начинаются на букву «Р».

```
SELECT *
FROM STUDENT
WHERE SURNAME LIKE 'P%';
```

В случае необходимости включения в образец самих символов «_» и «%» применяют так называемые *escape-символы*. Если escape-символ предшествует знаку «_» и «%», то эти знаки будут восприниматься буквально. Например, можно задать образец поиска с помощью следующего выражения:

```
LIKE '_\_P' ESCAPE '\'
```

В этом выражении символ '\' с помощью ключевого слова **escape** объявляется escape-символом. Первый символ «_» в заданном шаблоне поиска '__P' будет соответствовать, как и ранее, любому символу в проверяемой строке. Однако второй символ «_», следующий после символа '\', объявленного escape-символом, уже будет интерпретироваться буквально как обычный символ, так же как и символ 'P' в заданном шаблоне.

Обращаем внимание на то, что рассмотренные выше операторы сравнения «=, <, >, <=, >=, <>» и операторы **IN, BETWEEN** и **LIKE** ни в коем случае нельзя использовать для проверки содержимого поля на наличие в нем пустого значения NULL. Для этих целей предназначены специальные операторы IS NULL (является пустым) и IS NOT NULL (является не пустым).

Упражнения

1.Напишите запрос на вывод находящихся в таблице EXAM_MARKS номеров предметов обучения, экзамены по которым сдавались между 10 и 20 января 1999 года.

2. Напишите запрос, выбирающий данные обо всех предметах обучения, экзамены по которым сданы студентами, имеющими идентификаторы 12 и 32.

3.Напишите запрос на вывод названий предметов обучения, начинающихся на букву «И».

4. Напишите запрос, выбирающий сведения о студентах, у которых имена начинаются на буквы «И» или «С».

5.Напишите запрос для выбора из таблицы EXAM_MARKS записей, в которых отсутствуют значения оценок (поле mark).

6. Напишите запрос на вывод из таблицы EXAM_MARKS записей, имеющих в поле MARK значения оценок.

Агрегирование и групповые функции

Агрегирующие функции позволяют получать из таблицы сводную (агрегированную) информацию, выполняя операции над группой строк таблицы. Для задания в SELECT-запросе агрегирующих операций используются следующие ключевые слова:

COUNT определяет количество строк или значений поля, выбранных посредством запроса и не являющихся NULL-значениями;

SUM вычисляет арифметическую сумму всех выбранных значений данного поля;

AVG вычисляет среднее значение для всех выбранных значений данного поля;

MAX вычисляет наибольшее из всех выбранных значений данного поля;

MIN вычисляет наименьшее из всех выбранных значений данного поля.

В SELECT-запросе агрегирующие функции используются аналогично именам полей, при этом последние (имена полей) используются в качестве аргументов этих функций.

Функция AVG предназначена для подсчета среднего значе- ния поля на множестве записей таблицы.

Например, для определения среднего значения поля MARK (оценки) по всем записям таблицы EXAM_MARKS можно использовать запрос с функцией AVG следующего вида:

SELECT AVERAGE (MARK)
FROM EXAM_MARKS;

Для подсчета общего количества строк в таблице следует использовать функцию COUNT со звездочкой.

SELECT COUNT**(*)**
FROM EXAM MARKS;

Аргументы DISTINCT и ALL позволяют, соответственно, исключать и включать дубликаты обрабатываемых функцией COUNT значений, при этом необходимо учитывать, что при использовании опции ALL значения NULL все равно не войдут в число подсчитываемых значений.

SELECT COUNT(DISTINCT SUBJ_ID)
FROM SUBJECT;

Предложение GROUP BY (группировать по) позволяет группировать записи в подмножества, определяемые значениями какого-либо поля, и

применять агрегирующие функции уже не ко всем записям таблицы, а раздельно к каждой сформированной группе.

Предположим, требуется найти максимальное значение оценки, полученной каждым студентом. Запрос будет выглядеть следующим образом:

```
SELECT STUDENT_ID, MAX(MARK)
FROM EXAM_MARKS
GROUP BY STUDENT_ID;
```

Выбираемые из таблицы EXAM_MARKS записи группируются по значениям поля STUDENT_ID, указанного в предложении GROUP BY, и для каждой группы находится максимальное значение поля mark. Предложение GROUP BY позволяет применять агрегирующие функции к каждой группе, определяемой общим значением поля (или полей), указанных в этом предложении. В приведенном запросе рассматриваются группы записей, сгруппированные по идентификаторам студентов.

В конструкции GROUP BY для группирования может быть использовано более одного столбца. Например:

```
SELECT STUDENT_ID, SUBJ_ID, MAX(MARK)
 FROM EXAM_MARKS
 GROUP BY STUDENT_ID, SUBJ_ID;
```

В этом случае строки вначале группируются по значениям первого столбца, а внутри этих групп − в подгруппы по значениям второго столбца. Таким образом, GROUP BY не только устанавливает столбцы, по которым осуществляется группирование, но и указывает порядок разбиения столбцов на группы. Следует иметь в виду, что в предложении GROUP BY должны быть указаны все выбираемые столбцы, приведенные после ключевого слова SELECT, кроме столбцов, указанных в качестве аргумента в агрегирующей функции.

При необходимости часть сформированных с помощью GROUP BY групп может быть исключена с помощью предложения HAVING.

Предложение HAVING определяет критерий, по которому ты следует включать в выходные данные, по аналогии с предложением WHERE, которое осуществляет это для отдельных строк.

```
SELECT SUBJ_NAME, MAX(HOUR)
 FROM SUBJECT
 GROUP BY SUBJ_NAME
 HAVING MAX (HOUR) >= 72;
```

В условии, задаваемом предложением HAVING, указывают только поля или выражения, которые на выходе имеют единенное значение для каждой выводимой группы.

Упражнения

1. Напишите запрос для подсчета количества студентов, сдававших экзамен по предмету обучения с идентификатором, равным 20.
2. Напишите запрос, который позволяет подсчитать в таблице EXAM_MARKS количество различных предметов обучения.
3. Напишите запрос, который выполняет выборку для каждого студента значения его идентификатора и минимальной из полученных им оценок.
4. Напишите запрос, осуществляющий выборку для каждого студентам значения его идентификатора и максимальной из полученных им оценок.
5. Напишите запрос, выполняющий вывод фамилии первого в алфавитном порядке (по фамилии) студента, фамилия которого начинается на букву «И».
6. Напишите запрос, который выполняет вывод (для каждого предмета обучения) наименования предмета и максимального значения номера семестра, в котором этот предмет преподается.
7. Напишите запрос, который выполняет вывод данных для каждого конкретного дня сдачи экзамена о количестве студентов, сдававших экзамен в этот день.
8. Напишите запрос для получения среднего балла для каждого курса по каждому предмету.
9. Напишите запрос для получения среднего балла для каждого студента.
10. Напишите запрос для получения среднего балла для каждого экзамена.
11. Напишите запрос для определения количества студентов, сдававших каждый экзамен.
12. Напишите запрос для определения количества изучаемых предметов на каждом курсе.

Упорядочение выходных полей (ORDER BY)

Как уже отмечалось, записи в таблицах реляционной базы не упорядочены. Однако данные, выводимые в результате выполнения запроса, могут быть упорядочены. Для этого используется оператор ORDER BY, который позволяет упорядочить выводимые записи в соответствии со значениями одного нескольких выбранных столбцов. При этом можно задать возрастающую (ASC) или убывающую (DESC) последовательность сортировки для каждого из столбцов. По умолчанию принята возрастающая последовательность сортировки. Запрос, позволяющий выбрать все данные из таблицы предметов обучения SUBJECT с упорядочением по наименованиям предметов, выглядит следующим образом:

SELECT*

FROM SUBJECT
ORDER BY SUBJ_NAME;

Тот же список, но упорядоченный в обратном порядке, получить запросом:
SELECT*
FROM SUBJECT
ORDER BY SUBJ NAME **DESC;**

Можно упорядочить выводимый список предметов обучения по значениям семестров, а внутри семестров – по наименованиям предметов.

SELECT *
FROM SUBJECT
ORDER BY SEMESTER, SUBJ_NAME;

Предложение **ORDER BY** может использоваться с GROUP BY для упорядочения групп записей. При этом оператор ORDER BY в запросе *всегда должен быть последним.*

SELECT SUBJ_NAME, SEMESTER, **MAX**(HOUR)
FROM SUBJECT
GROUP BY SEMESTER, SUBJ_NAME
ORDER BY SEMESTER;

При упорядочении вместо наименований столбцов можно указывать их номера, имея, однако, ввиду, что в данном случае это номера столбцов, указанные при определении выходных данных в запросе, а не номера столбцов в таблице. Полем с номером 1 является первое поле, указанное в предложении **ORDER BY** – независимо от его расположения в таблице.

SELECT SUBJ_ID, SEMESTER
FROM SUBJECT
ORDER BY 2 DESC;

В этом запросе выводимые записи будут упорядочены по полю SEMESTR**.**

Если в поле, которое используется для упорядочения, существуют NULL-значения, то все они размещаются в конце или предшествуют всем остальным значениям этого поля.

Упражнения

1. Предположим, что стипендия всем студентам увеличена на *20%*. Напишите запрос к таблице STUDENT, выполняющий вывод номера студента, фамилию студента и величину увеличенной стипендии. Выходные данные упорядочить: а) по значению последнее столбца (величине стипендии); б) в алфавитном порядке фамилий студентов.

Вложенные подзапросы

SQL позволяет использовать одни запросы внутри других запросов, то есть вкладывать запросы друг в друга. Предположим, известна фамилия студента («Петров»), но неизвестно значение STUDENT_ID для него. Чтобы извлечь данные обо всех оценках этого студента, можно записать следующий запрос:

```
SELECT *
FROM EXAM_MARKS
WHERE STUDENT_ID =
(SELECT STUDENT_ID
 FROM STUDENT SURNAME = 'Петров');
```

Как работает запрос SQL со связанным подзапросом?

- выбирается строка из таблицы, имя которой указано во внешнем запросе.
- выполняется подзапрос и полученное значение применяется для анализа этой строки в условии предложения WHERE внешнего запроса.
- по результату оценки этого условия принимается решение о включении или не включении строки в состав выходных данных.
- Процедура повторяется для следующей строки таблицы внешнего запроса.

Следует обратить внимание, что приведенный выше запрос корректен только в том случае, если в результате выполнения указанного в скобках подзапроса возвращается *единственное значение.* Если в результате выполнения подзапроса будет возвращено несколько значений, то этот подзапрос будет ошибочным. В данном примере это произойдет, если в таблице STUDENT будет несколько записей со значениями пол SURNAME = 'Петров'.

В некоторых случаях для гарантии получения единственного значения в результате выполнения подзапроса используете DISTINCT. Одним из видов функций, которые автоматически *всегда* выдают в результате единственное значение для любого количества строк, являются агрегирующие функции.

Оператор IN также широко применяется в подзапросах. О задает список значений, с которыми сравниваются другие значения для определения истинности задаваемого этим оператором предиката.

Данные обо всех оценках (таблица EXAM_MARKS) студентов из Воронежа можно выбрать с помощью следующего запроса:

```
SELECT *
FROM EXAM_MARKS WHERE STUDENT_ID IN
(SELECT STUDENT_ID
FROM STUDENT
WHERE CITY = 'Воронеж');
```

Подзапросы можно применять внутри предложения HAVING. Пусть требуется определить количество предметов обучение с оценкой, превышающей среднее значение оценки студенте с идентификатором 301:

```
SELECT COUNT(DISTINCT SUBJ_ID), MARK
FROM EXAM_MARKS
GROUP BY MARK
HAVING MARK >
(SELECT AVG(MARK)
FROM EXAM_MARKS
WHERE STUDENT_ID = 301);
```

Формирование связанных подзапросов

При использовании подзапросов во внутреннем запросе можно ссылаться на таблицу, имя которой указано в предложении FROM внешнего запроса. В этом случае такой *связанный* подзапрос выполняется по одному разу для *каждой* строки таблицы основного запроса.

Пример: выбрать сведения обо всех предметах обучения, по которым проводился экзамен 20 января 1999 г.

```
SELECT *
  FROM SUBJECT  SU
  WHERE '20/01/1999' IN
    (SELECT EXAM_DATE
     FROM EXAM_MARKS  EX
     WHERE  SU.SUBJ_ID =  EX.SUBJ_ID);
```

В некоторых СУБД для выполнения этого запроса может потребоваться преобразование значения даты в символьный тип. В приведенном запросе SU и EX являются псевдонимами (алиасами), то есть специально вводимыми именами, которые могут быть использованы в данном запросе вместо настоящих имён. В приведенном примере они используются вместо имён таблиц SUBJECT и EXAM_MARKS **.**

Эту же задачу можно решить с помощью операции соединения таблиц:

```
SELECT DISTINCT SU.SUBJ_ID, SUBJ_NAME, HOUR, SEMESTER
  FROM SUBJECT FIRST,EXAM_MARKS SECOND
  WHERE FIRST.SUBJ_ID = SECOND.SUBJ_ID
  AND SECOND. EXAM_DATE = '20/01/1999';
```

В этом выражении алиасами таблиц являются имена FIRST и SECOND.

Можно использовать подзапросы, связывающие таблицу со своей собственной копией. Например, надо найти идентификаторы, фамилии и

стипендии студентов, получающих стипендию выше средней на курсе, на котором они учатся.

```
SELECT DISTINCT STUDENT_ID, SURNAME, STIPEND
FROM STUDENT E1
WHERE STIPEND >
(SELECT AVG(STIPEND)
FROM STUDENT E2
WHERE El.KURS = E2.KURS);
```

Тот же результат можно получить с помощью следующего запроса:

```
SELECT DISTINCT STUDENT_ID, SURNAME, STIPEND
FROM STUDENT E1,
(SELECT KURS, AVG (STIPEND) AS AVG_STIPEND
FROM STUDENT E2
GROUP BY E2.KURS) E3
WHERE E1.STIPEND > AVG_STIPEND AND E1.KURS=E3.KURS;
```

Обратите внимание – второй запрос будет выполнен гораздо быстрее. Дело в том, что в первом варианте запроса агрегирующая функция AVG выполняется над таблицей, указанной в подзапросе, для *каждой* строки внешнего запроса. В другом варианте вторая таблица (алиас E2) обрабатывается агрегирующей функцией один раз, в результате чего формируется вспомогательная таблица (в запросе она имеет алиас E3), со строками которой затем соединяются строки первой таблицы (алиас E1). Следует иметь в виду, что реальное время выполнения запроса в большой степени зависит от оптимизатора запросов конкретной СУБД.

Связанные подзапросы в HAVING

В разделе 2.4 указывалось, что предложение GROUP BY позволяет группировать выводимые SELECT-запросом записи по значению некоторого поля. Использование предложения HAVING позволяет при выводе осуществлять фильтрацию таких групп.

Предикат предложения HAVING оценивается не для каждой строки результата, а для каждой группы выходных записей, сформированной предложением GROUP BY внешнего запроса. Пусть, например, необходимо по данным из таблицы EXAM_MARKS определить сумму полученных студентами оценок (значений поля MARK), сгруппировав значения оценок по датам экзаменов и исключив те дни, когда число студентов, сдававших в течение дня экзамены, было меньше 10.

```
SELECT EXAM_DATE, SUM(MARK)
 FROM EXAM_MARKS A
 GROUP BY EXAM_DATE
```

```
HAVING 10 <
(SELECT COUNT(MARK) FROM EXAM_MARKS B
  WHERE A.EXAM_DATE = B .EXAM_DATE);
```

Подзапрос вычисляет количество строк с одной и той же датой, совпадающей с датой, для которой сформирована очередная группа основного запроса.

Упражнения

1. Напишите запрос с подзапросом для получения данных обо всех оценках студента с фамилией «Иванов». Предположим, что его персональный номер неизвестен. Всегда ли такой запрос будет корректным?

2. Напишите запрос, выбирающий данные об именах всех студентов, имеющих по предмету с идентификатором 101 балл выше общего среднего балла.

3. Напишите запрос, который выполняет выборку имен всех студентов, имеющих по предмету с идентификатором 102 балл ниже общего среднего балла.

4. Напишите запрос, выполняющий вывод количества предметов, по которым экзаменовался каждый студент, сдававший более 20 предметов.

5. Напишите команду SELECT, использующую связанные подзапросы и выполняющую вывод имен и идентификаторов студентов, у которых стипендия совпадает с максимальным значением стипендии для города, в котором живет студент.

6. Напишите запрос, который позволяет вывести имена и идентификаторы всех студентов, для которых точно известно, что они проживают в городе, где нет ни одного университета.

7. Напишите два запроса, которые позволяют вывести имена и идентификаторы всех студентов, для которых точно известно, что они проживают не в том городе, где расположен их университет. Один запрос с использованием соединения, а другой − с использованием связанного подзапроса.

Использование оператора EXISTS

Используемый в SQL оператор **EXISTS** (существует) генерирует значение **истина** или **ложь,** подобно булеву выражению. Используя подзапросы в качестве аргумента, этот оператор оценивает результат выполнения подзапроса как истинный, если этот подзапрос генерирует выходные данные, то в случае *существования (возврата) хотя бы одного найденного значения.* В противном случае результат подзапроса ложный. Оператор EXISTS не может принимать значение UNKNOWN (неизвестно).

Пусть, например, нужно извлечь из таблицы EXAM_MARKS данные о студентах, получивших хотя бы одну неудовлетворительную оценку.

```
SELECT DISTINCT STUDENT_ID
```

```
FROM EXAM_MARKS A
WHERE EXISTS
 (SELECT *
FROM EXAM_MARKS B
WHERE MARK < 3
AND B.STUDENT_ID = A. STUDENT__ID);
```

При использовании связанных подзапросов предложение EXISTS анализирует каждую строку таблицы, на которую имеется ссылка во внешнем запросе. Главный запрос получает строки-кандидаты на проверку условия. Для каждой строки-кандидата выполняется подзапрос. Как только подзапрос находит строку, где в столбце MARK значение удовлетворяет условию, он прекращает выполнение и возвращает значение истина внешнему запросу, который затем анализирует свою строку-кандидата.

Например, требуется получить идентификаторы предметов обучения, экзамены по которым сдавались не одним, а несколькими студентами:

```
SELECT DISTINCT SUBJ_ID
FROM EXAM_MARKS A
WHERE EXISTS
 (SELECT *
FROM EXAM_MARKS B
WHERE A.SUBJ_ID = B.SUBJ_ID
AND A.STUDENT_ID <  > B. STUDENT_ID);
```

Часто EXISTS применяется с оператором NOT (по-русски NOT EXISTS интерпретируется, как *«не существует»).* Если предыдущий запрос сформулировать следующим образом – найти идентификаторы предметов обучения, которые сдавались одним и только одним студентом (другими словами, для которых не существует другого сдававшего студента), то достаточно просто поставить NOT перед EXISTS.

Следует иметь в виду, что в подзапросе, указываемом в операторе EXISTS, *нельзя использовать агрегирующие функции.*

Возможности применения вложенных запросов весьма разнообразны. Например, пусть из таблицы STUDENT требуется извлечь строки для каждого студента, сдавшего более одного предмета.

```
SELECT *
FROM STUDENT FIRST
WHERE EXISTS
(SELECT SUBJ_ID
FROM EXAM_MARKS SECOND
GROUP BY SUBJ_ID
HAVING COUNT (SUBJ_ID) > 1
WHERE FIRST.STUDENT_ID = SECOND.STUDENT_ID);
```

Упражнения

1. Напишите запрос с **EXISTS,** позволяющий вывести данные обо всех студентах, обучающихся в вузах, которые имеют рейтинг выше 300.
2. Напишите предыдущий запрос, используя соединения.
3. Напишите запрос с **EXISTS,** выбирающий сведения обо всех студентах, для которых в том же городе, где живет студент, существуют университеты, в которых он не учится.
4. Напишите запрос, выбирающий из таблицы SUBJECT данные о названиях предметов обучения, экзамены по которым сданы более чем одним студентом.

Учебное издание

Голубенко Евгений Владимирович
Кулькин Александр Георгиевич

УПРАВЛЕНИЕ ДАННЫМИ

Печатается в авторской редакции

Технический редактор А.В. Артамонов

Подписано в печать 14.08.17. Формат 60×84/16.
Бумага газетная. Ризография. Усл. печ. л. 3,25.
Тираж экз. Изд. № 903. Заказ .

Редакционно-издательский центр ФГБОУ ВО РГУПС.

Адрес университета: 344038, г. Ростов н/Д, пл. Ростовского Стрелкового Полка Народного Ополчения, д. 2.