Белорусский национальный технический университет
Республиканский институт инновационных технологий
Кафедра «Инженерная педагогика и психология»

СОГЛАСОВАНО
Заведующий кафедрой
_____И.Л. Прокопчик-Гайко

СОГЛАСОВАНО
Директор института
_____В.Л.Соломахо

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

«Модели данных и системы управления базами данных»

для специальности 1-31 03 74 «Прикладная информатика»

Составители: Молчина Л.И., старший преподаватель

Рассмотрено и утверждено
на заседании Совета РИИТ ____. _____. 20___ г.,
протокол №

СОДЕРЖАНИЕ

1 Теоретический раздел

В теоретическом разделе представлены:
- материалы к лекционному курсу;
- мультимедийные презентации;
- электронные образовательные ресурсы по дисциплине.

Разделы "Теория множеств", "Математическая логика", "Реляционная алгебра" разработаны кандидатом технических наук, доцентом Чичко О.И.

1.1 Материалы к лекционному курсу

Материалы к лекционному курсу

1.1.1 Элементы теории множеств

Человеческое мышление устроено так, что мир представляется состоящим из отдельных “объектов”, хотя философам давно ясно, что мир – единое неразрывное целое, и выделение в нем объектов – это не более чем произвольный акт нашего мышления, позволяющий сформировать доступную для рационального анализа картину мира. Но как бы там ни было, выделение объектов и их совокупностей – естественный (или даже единственно возможный) способ организации нашего мышления, поэтому неудивительно, что он лежит в основе главного инструмента описания точного знания – математики.

1.1.1.1 Теория множеств.

План

1. Понятие множества.
2. Задание множеств.
3. Операции над множествами
4. Свойства операций над множествами.
5. Упорядоченные пары.
6. Прямое произведение множеств.
7. Отношения и их свойства.
8. Функции и их свойства .

1. Множества.

Понятия “множества”, “отношения”, “функции” и близкие к ним составляют основной словарь дискретной (равно как и классической “непрерывной”) математики. Именно эти базовые понятия будем рассматривать, закладывая необходимую основу для дальнейшего изучения предмета. Причем, будем рассматриваются только ***конечные*** множества

Можно сказать, что ***множество*** – это любая определенная совокупность

объектов. Объекты, из которых составлено множество, называются его ***элементами***. Элементы множества различны и отличимы друг от друга.

Примеры.

Множество *S* страниц в книге.

Множество N *натуральных* чисел 1, 2, 3, ...

Множество P *простых* чисел 2, 3, 5, 7, 11, ...

Множество Z *целых* чисел: ... , -2, -1, O, 1, 2, ...

Множество R *вещественных* чисел.

Если объект *x* является элементом множества M, то говорят, что *x принадлежит M.* Обозначение: *x* ° *M.* В противном случае говорят, что *x не принадлежит M.* Обозначение: *x* □ *M.*

ЗАМЕЧАНИЕ ——————

Обычно множества обозначают прописными буквами латинского алфавита, а элементы множеств – строчными буквами.

ОТСТУПЛЕНИЕ ——————

Понятия множества, элемента и принадлежности, которые на первый взгляд представляются интуитивно ясными, при ближайшем рассмотрении такую ясность утрачивают. Во-первых, проблематична отличимость элементов. Например, символы *a и α,* которые встречаются– это один элемент множества *A* или два разных элемента? Во-вторых, проблематична возможность (без дополнительных усилий) указать, принадлежит ли данный элемент данному множеству. Например, является ли число 869584769215374850678574 67 простым?

Множества, как объекты, могут быть элементами других множеств. Множество, элементами которого являются множества, обычно называется *классом* или *семейством.*

ЗАМЕЧАНИЕ ——————

Семейства множеств обычно обозначают прописными “рукописными” буквами латинского алфавита, чтобы отличить их от множеств, не содержащих множеств в качестве элементов.

Множество, не содержащее элементов, называется ***пустым.***

Обозначение □ **.**

Обычно в конкретных рассуждениях элементы всех множеств берутся из некоторого одного, достаточно широкого множества (своего для каждого случая), которое называется ***универсальным*** множеством или ***универсумом*** и обозначается **U**.

2. Задание множеств.

Чтобы задать множество, нужно указать, какие элементы ему принадлежат. Это можно сделать различными способами:

перечислением элементов: $M: =\{a_1, a_2,..., a_k\}$;

характеристическим предикатом'. M :={x\ P(x)};

порождающей процедурой: $M : = [x \setminus x: = f\}$.

ЗАМЕЧАНИЕ

При задании множеств перечислением обозначения элементов обычно

заключают в фигурные скобки и разделяют запятыми. Характеристический предикат – это некоторое условие, выраженное в форме логического утверждения или процедуры, возвращающей логическое значение. Если для данного элемента условие выполнено, то он принадлежит определяемому множеству, в противном случае – не принадлежит. Порождающая процедура – это процедура, которая, будучи запущенной, порождает некоторые объекты, являющиеся элементами определяемого множества

На латыни слово "предикат" (praedicatum) означает "сказуемое". Традиционная логика выделяет в элементарном высказывании (суждении) субъект и предикат. Субъект (логическое подлежащее) - то, о чем говориться в высказывании; предикат (логическое сказуемое) - то, что говорится (утверждается или отрицается) о субъекте. Например, в высказывании "Кошка имеет четыре ноги" "кошка" - субъект, "имеет четыре ноги" - предикат. Если на место слова "кошка" поставить слово "собака", то снова получится истинное высказывание "Собака имеет четыре ноги". Если же в качестве субъекта взять слово "курица", то получится ложное высказывание "Курица имеет четыре ноги". Заменив субъект переменной, получаем высказывательную форму *"x* имеет четыре ноги" и предикат как функцию, задаваемую этой формой. Естественным обобщением понятия "одноместный предикат" является понятие "n-местный предикат".

Одноместные предикаты иногда называют *предикатами-свойствами,* а n-местные при $n > 1$ - предикатами-*отношениями.* Так, например, предикат $x < 0$ выражает свойство чисел, а предикат $x < y$ - отношение между числами. Позже смысл слов "свойство" и "отношение" будет уточнен.

Примеры.

1. M_1:={1,2,3,4,5,6,7,8,9};

2. M_2:={n|n° N&n< 10};

3. Предикат P задан таблицей

1	2	3	4	5	6	7	8	9
л	*и*	*л*	*и*	*л*	*и*	*л*	*и*	*л*

Выпишите область определения и множество значений предиката *P.* Укажите, чему равны P(1), P(6). Подберите высказывательную форму, с помощью которой можно задать предикат *P.*

3. Предикат *Q* задан на множестве однозначных натуральных чисел вы сказы нательной формой *"x* - простое число". Задайте предикат *Q* с помощью таблицы. Чему равны Q(l), Q(2), Q(6)?

Множество **целых чисел в диапазоне от *m* до *n* обозначают так: m...n. То есть** *m..n: = {k ° Z | 0 k & k □ n}* .

Перечислением можно задавать только *конечные* множества.

3. Операции над множествами.

Самого по себе понятия множества еще недостаточно – нужно определить

способы конструирования новых множеств из уже имеющихся, то есть операции над множествами.

Множество *A содержится* в множестве *B* (множество *B включает* множество *A),* если каждый элемент *A* есть элемент *B:*

A□B:=x ° A. x ° B.

В этом случае А называется *подмножеством B, B – надмножеством A.* Если *A □ B и A □B, то A* называется *собственным* подмножеством *B.* Заметим, что ∀M *M □ M.* По определению ∀M □ □ *M.*

Два множества *равны,* если они являются подмножествами друг друга:

A=B:=A □B &B □A.

Совокупность всех подмножеств множества А называется его ***булеаном*** и обозначается 2^A.

Мощность множества *M* обозначается как]M|. Для конечных множеств мощность – это число элементов. Например, |□ | = 0, но |{□ }| = 1. Если |A| = |B|, то множества *A n B* называются *равномощными.*

Обычно рассматриваются следующие операции над множествами:

объединение: **A□ B:={x\ x° A ∗ x° B};**

пересечение*:* **A□ B:={x\ x□ A & x□ B};**

разность: **A\B:={x | x □*A&x*□ B};**

симметрическая разность:

A! B*: =(A U* B) \ (A □*B)* = {x \ {x □ A *&* x □ B) *V (x*□ A *&* □ B)};

дополнение:

A:={x\x□ A}.

Пример

Пусть *A:* ={1,2,3}, *B:* ={3,4,5}.

Тогда

A U* B *={1,2,3,4,5}, A* □ B *{3}, A\B={1,2}, A ! B =-{1,2,4,5}.

На рис. 1.1 приведены *диаграммы Эйлера,* иллюстрирующие операции над множествами. Сами исходные множества изображаются фигурами (в данном случае овалами), а результат графически выделяется (в данном случае для выделения использована штриховка).

Операции пересечения и объединения допускают следующее обобщение. Пусть *I* – некоторое множество, элементы которого используются как индексы, и пусть для любого *i* ° *I* множество *Ai* известно.

Тогда

$\bigcup_{i\,°\,I} A_i := \{x \mid \exists\, i\,°\ I\, x\,°\, A_i\}$, $\bigcap_{i\,°\,I} A_i := \{x \mid \forall\, i\,°\ I\, x\,°\, A_i\}$.

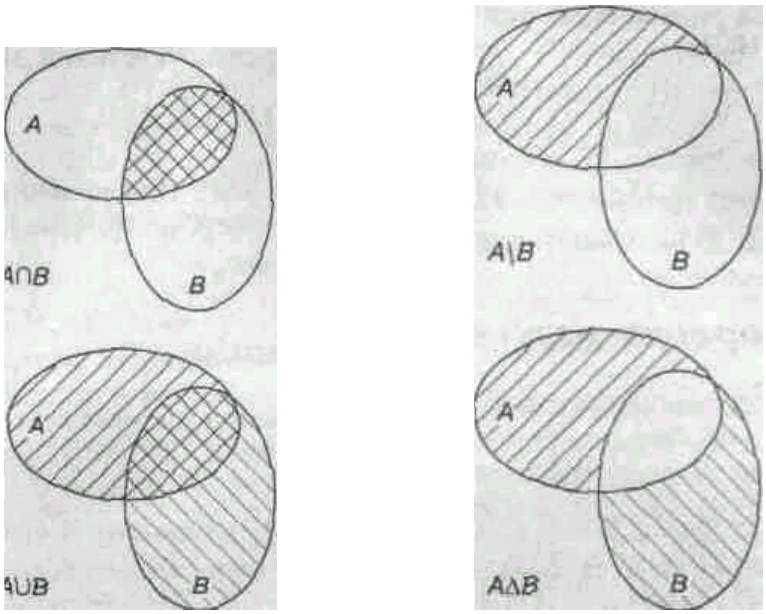


Рис. 1.1. Операции над множествами

4. Свойства операций над множествами.

Пусть задан универсум. Тогда для ∀ A,B,C □ U выполняются следующие свойства:

1. идемпотентность A □ A = A	A □ A = A
1. коммутативность A □ B = B □ A	A □ B = B □ A
1. ассоциативность A □ (B □ C) = (A □ B) □ C	A □ (B □ C) = (A □B) □ C
1. дистрибутивность A □ (B □ C) = (A □ B) □ (A □ C)	A □ (B □ C) = (A □ B) □ (A □ C)
1. поглощение (A □ B) A = A	(A □ B) □ A = A
1. свойство нуля A □ □ = A	A □ □ = □
1. свойство единицы A □ U = U	A □ U = A
1. инволютивность $\overline{\overline{A}} = A$	
1. законы де Моргана	

$\overline{A \cap B} = \overline{A} \cup \overline{B}$	$\overline{A \cup B} = \overline{A} \cap \overline{B}$
1. свойства дополнения $A \cup \overline{A} = U$	$A \cap \overline{A} =$ □
1. выражение для разности A \ B = $A \cap \overline{B}$	

В справедливости перечисленных свойств можно убедиться различными способами. Например, нарисовать диаграммы Эйлера для левой и правой частей равенства и убедиться, что они совпадают, или же провести формальное рассуждение для каждого равенства.

Рассмотрим для примера первое равенство: *A* □ *A* = A. Возьмем произвольный элемент *x,* принадлежащий левой части равенства, *x* ° A□A. По определению операции объединения □ имеем *x* ° A∗ *x* ° A . В любом случае *x* ° A. Взяв произвольный элемент из множества в левой части равенства, обнаружили, что он принадлежит множеству в правой части. Отсюда по определению включения множеств получаем, что *A* ° A. Пусть теперь *x* ° A. Тогда, очевидно, верно *x* ° A∗ *x* ° A. Отсюда по определению операции объединения имеем *x* ° A □ *A* . Таким образом,

A□ *A* □*A*. Следовательно, по определению равенства множеств,

A□*A*=A.

Аналогичные рассуждения нетрудно провести и для остальных равенств.

5. Упорядоченные пары.

Если *a* и *b* – объекты, то через (*a,b)* обозначим ***упорядоченную пару. Равенство*** упорядоченных пар определяется следующим образом:

(a,b) ==(c,d):=a=c&b=d.

Вообще говоря, *(a,b)* □ *(b,a).*

ЗАМЕЧАНИЕ Упорядоченные пары можно рассматривать как множества, если определить их так:

(a,b):={a,{a,b}}.

Упорядоченную **последовательность из n элементов $x_1, x_2, \ldots, x_n$ будем обозначать через $(x_1, x_2, \ldots, x_n)$. Круглые скобки используются для того, чтобы указать на порядок, в котором записаны элементы. Такая последовательность называется** *кортежем длины n* **или просто** *n-кой***. Элемент x_i называется** *i- той координатой кортежа.*

5.2. Прямое произведение множеств.

Пусть *A* и *B* – два множества. ***Прямым (декартовым)*** *произведением* двух множеств *A и B* называется множество упорядоченных пар, в котором первый элемент каждой пары принадлежит *A,* а второй принадлежит *B:*

A ∗ B: = {(a, b) | *a* □ *A & b* □ *B}.*

ЗАМЕЧАНИЕ ————————————

Точка на плоскости может быть задана упорядоченной парой координат, то есть двумя точками на координатных осях. Таким образом, $R^2 = R$ x R. Метод координат ввел в употребление Рене Декарт (1596-1650), отсюда и название «декартово произведение».

Степенью множества *A* называется его прямое произведение самого на себя. Обозначение: $A^n = A * A \ldots * A$.

Пример. Пусть A={1,2}, B={3,4}. Тогда, A*B={(1,3), (1,4), (2,3), (2,4)},
B*A={(3,1), (3,2), (4,1), (4,2)},
A*A={(1,1), (1,2), (2,1), (2,2)}.

6. Отношения.

Равенство, неравенство, параллельность, перпендикулярность, подобие, а также дружба, родство, соседство – все это примеры отношений.

Примеры.

1. Форма $x \downarrow\downarrow y$ выделяет из множества прямых на плоскости такие пары, компоненты которых находятся в отношении параллельности.

2. Если A={2,3,4,5,6,7,8}, то бинарное отношение P{(x,y) ¦x,y ° A, x делит y и x d3} можно записать в виде P={(2,2), (2,4), (2,6), (2,8), (3,3), (3,6)}.

3. На множестве (1,2,3,4) задано отношение x>y . Выпишите пары, компоненты которых находятся в отношении R.

Бинарные отношения *R* A*B иногда удобно изображать графически.

Пример 4. Показать отношение P_1 = {(a,2), (b,1),(c,2)} между множествами A= {a, b,c} и B={1,2,3}, а также P_2 = {(a,b), (b,b), (c,a)} на множестве A.

Пусть *A* и *B* – два множества. ***(Бинарным) отношением R*** из множества A в множество *B* называется подмножество прямого произведения *A* и *B:*

R *A* x *B*.

Для бинарных отношений обычно используется *инфиксная* форма записи:

aRb:==(a,b) ° R *A*x B.

Если A = B, то говорят, что R есть отношение *на множестве A*.

Пример.

Пусть задан универсум *U*. Тогда ° (принадлежность) – отношение из множества *U* в множество 2^U, а (включение) и = (равенство) – отношения на 2^U . Хорошо известны отношения =, <, , >, ς, , определенные на множестве чисел.

Введем обобщенное понятие отношения: ***n-местное (n-арное)*** отношение R – это множество упорядоченных наборов *(кортежей):*

$R \subseteq A_1 \times \bullet\bullet\bullet \times An = \{(a_1,a_2,\ldots,a_n) \mid a_1 \circ A_1 \& \ldots \& a_n \circ A_n\}$.

Множества A_i не обязательно различны.

Произведением бинарных отношений P_1 □ A*B и P_2 □ B*C или ***композицией*** P_1 и P_2 называется множество $\mathbf{P_1}$ ° $\mathbf{P_2}$ {(x,y) | x° A,y° C, и найдется элемент z° B такой, что (x,z) ° P_1 и (z,y)° P_2}.

7. Свойства отношений.

Отношения характеризуются наличием у них следующих свойств:

1) отношение называется **симметричным**, если ∀a,b° X из следует bRa***;***

2) отношение называется **антисимметричным**, если ∀a,b° X из aRb следует, что *Ь* не находится в отношении R к a;

3) отношение (X,R) называется **рефлексивным**, если ∀a° X и справедливо aRa;

4) отношение называется **антирефлексивным**, если ∀a° X не выполняется aRa

5) отношение называется **транзитивным**, если из того, что aRb и bR*c,* следует *aRc.*

Примеры.

1. **Симметричности и антисимметричности**:

1. Отношение равенства симметрично на любом множестве
1. Отношение «быть больше» и «быть не меньше» на любом числовом множестве
1. Отношение родства симметрично на любом множестве людей
1. Отношение «быть сестрой» симметрично на любом множестве женщин.
1. Отношение дружбы, как правило, симметрично, а отношение любви, к сожалению, часто бывает несимметричным.

2. **Рефлексивности и антирефлексивности**:

2. Отношение равенства рефлексивно на любом множестве – каждый предмет равен самому себе.
2. Отношение неравенства антирефлексивно на любом множестве

3. **Транзитивности:**

3.1. На множестве прямых плоскости отношение параллельности транзитивно, а отношение перпендикулярности нетранзитивно.

Наиболее известными отношениями являются отношение эквивалентности и отношение порядка.

Отношение называется ***отношением эквивалентности*** и обозначается символом «Т», если оно рефлексивно (xTx), симметрично *x*T*y–>y*T*x) и* транзитивно *(x*T *y* и *y*T*z–>x*T*z).* Примерами отношений эквивалентности являются отношение «жить в одном городе» для множества людей, отношение параллельности прямых и т.п.

Отношение эквивалентности на любом множестве *M* задаёт разбиение его на подмножества, которые в этом случае называются ***классами эквивалентности.*** С другой стороны, любое разбиение $M' = \{A_1, A_2, ..., A_m\}$ множества *M* задаёт на этом множестве отношение, которое можно назвать «входить в одно и то же подмножество разбиения». Нетрудно убедиться, что это отношение симметрично, рефлексивно и транзитивно, т.е. является отношением эквивалентности.

Отношение называется ***отношением нестрогого порядка*** и обозначается символом «□», если оно рефлексивно, антисимметрично и транзитивно. Отношение называется ***отношением строгого порядка*** и обозначается символом «<», если оно антирефлексивно, антисимметрично и транзитивно. Для числовых множеств отношениями нестрогого порядка являются известные отношения «меньше или равно», «больше или равно», отношениями строго порядка - «меньше» или «больше».

Отношения порядка определяют некоторый порядок расположения элементов множества, на котором они заданы, другими словами позволяют сравнивать элементы множества по некоторому признаку. Множество называется ***упорядоченным,*** если любые два его элемента сравнимы, и ***частично упорядоченным*** в противном случае.

8. Функции и их свойства.

Понятие «функции» является одним из основополагающих в математике. В данном случае подразумеваются прежде всего функции, отображающие одно конечное множество объектов в другое конечное множество.

Определение функции.

Пусть f – отношение из A в B, такое что

$$\forall a(a,b) \in f \;\&\; (a,c) \in f \Rightarrow b = c.$$

Такое свойство отношения называется однозначностью, или функциональностью, а само отношение называется функцией из A в B и обозначается следующим образом:

$f : A \to B$ или $A \xrightarrow{f} B$

Если $f : A \to B$ то обычно используется префиксная форма записи:

$$b = f(a) := (a,b) \in f.$$

Если $b = f(a)$ то a называют аргументом, а b – значением функции.

ЗАМЕЧАНИЕ ———————————

Вообще, всякому отношению *R* из *A* в $B(R \subset A \times B)$ можно сопоставить (тотальную) функцию $R : A \times B \to \mathbf{0..1}$ (эта функция называется характеристической функцией отношения полагая:

$$R(a,b) := \begin{cases} \mathbf{1}, если a R b \\ \mathbf{0}, если a \overline{R} b \end{cases}$$

Пусть $f : A \to B$ тогда

область определения функции: $f_{\mathcal{A}} = \{a \in A \mid \exists b \in B\, b = f(a)\}$

область значений функции: $f_B := \{b \in B | \exists a \in A b = f(a)\}$

Если $f(a) = A,$ то функция называется тотальной, а если $f(a \neq A)$ – частичной.

Сужением функции $f : A \to B$ на множество M с A называется функция $f|M$ определяемая следующим образом:

$f|M := \{(a,b) | (a,b) \in f \ \& \ a \in M\}$

Для тотальной функции $f = f|A$.

Функция $f : A_1 \times \ldots \times A_n \to B$ называется функцией n аргументов, или n - местной функцией.

.

1.1.2 Понятия математической логики

Презентация: Лекция - Понятия математической логики

1.1.3 Основные понятия реляционной алгебры

Основная идея реляционной алгебры состоит в том, что коль скоро отношения являются множествами, то средства манипулирования отношениями могут базироваться на традиционных теоретико-множественных операциях, дополненных некоторыми специальными операциями, специфичными для баз данных.

1.1.3.1 Основные понятия реляционной алгебры

Реляционная алгебра (http://citforum.ru/database/osbd/glava_20.shtml)

Основная идея реляционной алгебры состоит в том, что коль скоро отношения являются множествами, то средства манипулирования отношениями могут базироваться на традиционных теоретико-множественных операциях, дополненных некоторыми специальными операциями, специфичными для баз данных.

Существует много подходов к определению реляционной алгебры, которые различаются набором операций и способами их интерпретации, но в принципе, более или менее равносильны. Мы опишем немного расширенный начальный вариант алгебры, который был предложен Коддом. В этом варианте набор основных алгебраических операций состоит из восьми операций, которые делятся на два класса - теоретико-множественные операции и специальные реляционные операции. В состав теоретико-множественных операций входят операции:

- объединения отношений;

- пересечения отношений;
- взятия разности отношений;
- прямого произведения отношений.

Специальные реляционные операции включают:

- ограничение отношения;
- проекцию отношения;
- соединение отношений;
- деление отношений.

Кроме того, в состав алгебры включается операция присваивания, позволяющая сохранить в базе данных результаты вычисления алгебраических выражений, и операция переименования атрибутов, дающая возможность корректно сформировать заголовок (схему) результирующего отношения.

1.1. Общая интерпретация реляционных операций

Если не вдаваться в некоторые тонкости, которые мы рассмотрим в следующих подразделах, то почти все операции предложенного выше набора обладают очевидной и простой интерпретацией.

- При выполнении операции объединения двух отношений производится отношение, включающее все кортежи, входящие хотя бы в одно из отношений-операндов.
- Операция пересечения двух отношений производит отношение, включающее все кортежи, входящие в оба отношения-операнда.
- Отношение, являющееся разностью двух отношений включает все кортежи, входящие в отношение - первый операнд, такие, что ни один из них не входит в отношение, являющееся вторым операндом.
- При выполнении прямого произведения двух отношений производится отношение, кортежи которого являются конкатенацией (сцеплением) кортежей первого и второго операндов.
- Результатом ограничения отношения по некоторому условию является отношение, включающее кортежи отношения-операнда, удовлетворяющее этому условию.

- При выполнении проекции отношения на заданный набор его атрибутов производится отношение, кортежи которого производятся путем взятия соответствующих значений из кортежей отношения-операнда.

- При соединении двух отношений по некоторому условию образуется результирующее отношение, кортежи которого являются конкатенацией кортежей первого и второго отношений и удовлетворяют этому условию.

- У операции реляционного деления два операнда - бинарное и унарное отношения. Результирующее отношение состоит из одноатрибутных кортежей, включающих значения первого атрибута кортежей первого операнда таких, что множество значений второго атрибута (при фиксированном значении первого атрибута) совпадает со множеством значений второго операнда.

- Операция переименования производит отношение, тело которого совпадает с телом операнда, но имена атрибутов изменены.

- Операция присваивания позволяет сохранить результат вычисления реляционного выражения в существующем отношении БД.

Поскольку результатом любой реляционной операции (кроме операции присваивания) является некоторое отношение, можно образовывать реляционные выражения, в которых вместо отношения-операнда некоторой реляционной операции находится вложенное реляционное выражение.

1.2. Замкнутость реляционной алгебры и операция переименования

Как мы говорили в предыдущей лекции, каждое отношение характеризуется схемой (или заголовком) и набором кортежей (или телом). Поэтому, если действительно желать иметь алгебру, операции которой замкнуты относительно понятия отношения, то каждая операция должна производить отношение в полном смысле, т.е. оно должно обладать и телом, и заголовком. Только в этом случае будет действительно возможно строить вложенные выражения.

Заголовок отношения представляет собой множество пар <имя-атрибута, имя-домена>. Если посмотреть на общий обзор реляционных операций, приведенный в предыдущем подразделе, то видно, что домены атрибутов результирующего отношения однозначно определяются доменами отношений-операндов. Однако с именами атрибутов результата не всегда все так просто.

Например, представим себе, что у отношений-операндов операции прямого произведения имеются одноименные атрибуты с одинаковыми доменами. Каким был бы заголовок результирующего отношения? Поскольку это множество, в нем не должны содержаться одинаковые элементы. Но и потерять атрибут в результате недопустимо. А это значит, что в этом случае вообще невозможно корректно выполнить операцию

прямого произведения.

Аналогичные проблемы могут возникать и в случаях других двуместных операций. Для их разрешения в состав операций реляционной алгебры вводится операция переименования. Ее следует применять в любом случае, когда возникает конфликт именования атрибутов в отношениях - операндах одной реляционной операции. Тогда к одному из операндов сначала применяется операция переименования, а затем основная операция выполняется уже безо всяких проблем.

В дальнейшем изложении мы будем предполагать применение операции переименования во всех конфликтных случаях.

1.3. Особенности теоретико-множественных операций реляционной алгебры

Хотя в основе теоретико-множественной части реляционной алгебры лежит классическая теория множеств, соответствующие операции реляционной алгебры обладают некоторыми особенностями.

Начнем с операции объединения (все, что будет говориться по поводу объединения, переносится на операции пересечения и взятия разности). Смысл операции объединения в реляционной алгебре в целом остается теоретико-множественным. Но если в теории множеств операция объединения осмысленна для любых двух множеств-операндов, то в случае реляционной алгебры результатом операции объединения должно являться отношение. Если допустить в реляционной алгебре возможность теоретико-множественного объединения произвольных двух отношений (с разными схемами), то, конечно, результатом операции будет множество, но множество разнотипных кортежей, т.е. не отношение. Если исходить из требования замкнутости реляционной алгебры относительно понятия отношения, то такая операция объединения является бессмысленной.

Все эти соображения приводят к появлению понятия *совместимости отношений по объединению*: два отношения совместимы по объединению в том и только в том случае, когда обладают одинаковыми заголовками. Более точно, это означает, что в заголовках обоих отношений содержится один и тот же набор имен атрибутов, и одноименные атрибуты определены на одном и том же домене.

Если два отношения совместимы по объединению, то при обычном выполнении над ними операций объединения, пересечения и взятия разности результатом операции является отношение с корректно определенным заголовком, совпадающим с заголовком каждого из отношений-операндов. Напомним, что если два отношения "почти" совместимы по объединению, т.е. совместимы во всем, кроме имен атрибутов, то до выполнения операции типа соединения эти отношения можно сделать полностью совместимыми по объединению путем применения операции переименования.

Заметим, что включение в состав операций реляционной алгебры трех операций

объединения, пересечения и взятия разности является очевидно избыточным, поскольку известно, что любая из этих операций выражается через две других. Тем не менее, Кодд в свое время решил включить все три операции, исходя из интуитивных потребностей потенциального пользователя системы реляционных БД, далекого от математики.

Другие проблемы связаны с операцией взятия прямого произведения двух отношений. В теории множеств прямое произведение может быть получено для любых двух множеств, и элементами результирующего множества являются пары, составленные из элементов первого и второго множеств. Поскольку отношения являются множествами, то и для любых двух отношений возможно получение прямого произведения. Но результат не будет отношением! Элементами результата будут являться не кортежи, а пары кортежей.

Поэтому в реляционной алгебре используется специализированная форма операции взятия прямого произведения - расширенное прямое произведение отношений. При взятии расширенного прямого произведения двух отношений элементом результирующего отношения является кортеж, являющийся конкатенацией (или слиянием) одного кортежа первого отношения и одного кортежа второго отношения.

Но теперь возникает второй вопрос - как получить корректно сформированный заголовок отношения-результата? Очевидно, что проблемой может быть именование атрибутов результирующего отношения, если отношения-операнды обладают одноименными атрибутами.

Эти соображения приводят к появлению понятия *совместимости по взятию расширенного прямого произведения*. Два отношения совместимы по взятию прямого произведения в том и только в том случае, если множества имен атрибутов этих отношений не пересекаются. Любые два отношения могут быть сделаны совместимыми по взятию прямого произведения путем применения операции переименования к одному из этих отношений.

Следует заметить, что операция взятия прямого произведения не является слишком осмысленной на практике. Во-первых, мощность ее результата очень велика даже при допустимых мощностях операндов, а во-вторых, результат операции не более информативен, чем взятые в совокупности операнды. Как мы увидим немного ниже, основной смысл включения операции расширенного прямого произведения в состав реляционной алгебры состоит в том, что на ее основе определяется действительно полезная операция соединения.

По поводу теоретико-множественных операций реляционной алгебры следует еще заметить, что все четыре операции являются ассоциативными. Т. е., если обозначить через OP любую из четырех операций, то (A OP B) OP C = A (B OP C), и следовательно, без введения двусмысленности можно писать A OP B OP C (A, B и C - отношения, обладающие свойствами, требуемыми для корректного выполнения соответствующей

операции). Все операции, кроме взятия разности, являются коммутативными, т.е. A OP B = B OP A.

1.4. Специальные реляционные операции

В этом подразделе мы несколько подробнее рассмотрим специальные реляционные операции реляционной алгебры: ограничение, проекция, соединение и деление.

Операция ограничения

Операция ограничения требует наличия двух операндов: ограничиваемого отношения и простого условия ограничения. Простое условие ограничения может иметь либо вид (a comp-op b), где a и b - имена атрибутов ограничиваемого отношения, для которых осмысленна операция сравнения comp-op, либо вид (a comp-op const), где a - имя атрибута ограничиваемого отношения, a const - литерально заданная константа.

В результате выполнения операции ограничения производится отношение, заголовок которого совпадает с заголовком отношения-операнда, а в тело входят те кортежи отношения-операнда, для которых значением условия ограничения является true.

Пусть UNION обозначает операцию объединения, INTERSECT - операцию пересечения, а MINUS - операцию взятия разности. Для обозначения операции ограничения будем использовать конструкцию A WHERE comp, где A - ограничиваемое отношение, а comp - простое условие сравнения. Пусть comp1 и comp2 - два простых условия ограничения. Тогда по определению:

- A WHERE comp1 AND comp2 обозначает то же самое, что и (A WHERE comp1) INTERSECT (A WHERE comp2)
- A WHERE comp1 OR comp2 обозначает то же самое, что и (A WHERE comp1) UNION (A WHERE comp2)
- A WHERE NOT comp1 обозначает то же самое, что и A MINUS (A WHERE comp1)

С использованием этих определений можно использовать операции ограничения, в которых условием ограничения является произвольное булевское выражение, составленное из простых условий с использованием логических связок AND, OR, NOT и скобок.

На интуитивном уровне операцию ограничения лучше всего представлять как взятие некоторой "горизонтальной" вырезки из отношения-операнда.

Операция взятия проекции

Операция взятия проекции также требует наличия двух операндов - проецируемого

отношения A и списка имен атрибутов, входящих в заголовок отношения A.

Результатом проекции отношения A по списку атрибутов a_1, a_2, ..., a_n является отношение, с заголовком, определяемым множеством атрибутов a_1, a_2, ..., a_n, и с телом, состоящим из кортежей вида $<a_1:v_1, a_2:v_2, ..., a_n:v_n>$ таких, что в отношении A имеется кортеж, атрибут a_1 которого имеет значение v_1, атрибут a_2 имеет значение v_2, ..., атрибут a_n имеет значение v_n. Тем самым, при выполнении операции проекции выделяется "вертикальная" вырезка отношения-операнда с естественным уничтожением потенциально возникающих кортежей-дубликатов.

Операция соединения отношений

Общая операция соединения (называемая также соединением по условию) требует наличия двух операндов - соединяемых отношений и третьего операнда - простого условия. Пусть соединяются отношения A и B. Как и в случае операции ограничения, условие соединения comp имеет вид либо (a comp-op b), либо (a comp-op const), где a и b - имена атрибутов отношений A и B, const - литерально заданная константа, a comp-op - допустимая в данном контексте операция сравнения.

Тогда по определению результатом операции сравнения является отношение, получаемое путем выполнения операции ограничения по условию comp прямого произведения отношений A и B.

Если внимательно осмыслить это определение, то станет ясно, что в общем случае применение условия соединения существенно уменьшит мощность результата промежуточного прямого произведения отношений-операндов только в том случае, когда условие соединения имеет вид (a comp-op b), где a и b - имена атрибутов разных отношений-операндов. Поэтому на практике обычно считают реальными операциями соединения именно те операции, которые основываются на условии соединения приведенного вида.

Хотя операция соединение в нашей интерпретации не является примитивной (поскольку она определяется с использованием прямого произведения и проекции), в силу особой практической важности она включается в базовый набор операций реляционной алгебры. Заметим также, что в практических реализациях соединение обычно не выполняется именно как ограничение прямого произведения. Имеются более эффективные алгоритмы, гарантирующие получение такого же результата.

Имеется важный частный случай соединения - эквисоединение и простое, но важное расширение операции эквисоединения - естественное соединение. Операция соединения называется *операцией эквисоединения*, если условие соединения имеет вид (a = b), где a и b - атрибуты разных операндов соединения. Этот случай важен потому, что (a) он часто встречается на практике, и (b) для него существуют эффективные

алгоритмы реализации.

Операция естественного соединения применяется к паре отношений A и B, обладающих (возможно составным) общим атрибутом c (т.е. атрибутом с одним и тем же именем и определенным на одном и том же домене). Пусть ab обозначает объединение заголовков отношений A и B. Тогда естественное соединение A и B - это спроектированный на ab результат эквисоединения A и B по A/c и BBC. Если вспомнить введенное нами в конце предыдущей главы определение внешнего ключа отношения, то должно стать понятно, что основной смысл операции естественного соединения - возможность восстановления сложной сущности, декомпозированной по причине требования первой нормальной формы. Операция естественного соединения не включается прямо в состав набора операций реляционной алгебры, но она имеет очень важное практическое значение.

Операция деления отношений

Эта операция наименее очевидна из всех операций реляционной алгебры и поэтому нуждается в более подробном объяснении. Пусть заданы два отношения - A с заголовком $\{a_1, a_2, ..., a_n, b_1, b_2, ..., b_m\}$ и B с заголовком $\{b_1, b_2, ..., b_m\}$. Будем считать, что атрибут b_i отношения A и атрибут b_i отношения B не только обладают одним и тем же именем, но и определены на одном и том же домене. Назовем множество атрибутов $\{a_j\}$ составным атрибутом a, а множество атрибутов $\{b_j\}$ - составным атрибутом b. После этого будем говорить о реляционном делении бинарного отношения A(a,b) на унарное отношение B(b).

Результатом деления A на B является унарное отношение C(a), состоящее из кортежей v таких, что в отношении A имеются кортежи <v, w> такие, что множество значений {w} включает множество значений атрибута b в отношении B.

Предположим, что в базе данных сотрудников поддерживаются два отношения: СОТРУДНИКИ (ИМЯ, ОТД_НОМЕР) и ИМЕНА (ИМЯ), причем унарное отношение ИМЕНА содержит все фамилии, которыми обладают сотрудники организации. Тогда после выполнения операции реляционного деления отношения СОТРУДНИКИ на отношение ИМЕНА будет получено унарное отношение, содержащее номера отделов, сотрудники которых обладают всеми возможными в этой организации именами.

2. Реляционное исчисление

Предположим, что мы работаем с базой данных, обладающей схемой СОТРУДНИКИ (СОТР_НОМ, СОТР_ИМЯ, СОТР_ЗАРП, ОТД_НОМ) и ОТДЕЛЫ (ОТД_НОМ, ОТД_КОЛ, ОТД_НАЧ), и хотим узнать имена и номера сотрудников, являющихся начальниками отделов с количеством сотрудников больше 50.

Если бы для формулировки такого запроса использовалась реляционная алгебра, то мы

получили бы алгебраическое выражение, которое читалось бы, например, следующим образом:

- выполнить соединение отношений СОТРУДНИКИ и ОТДЕЛЫ по условию СОТР_НОМ = ОТД_НАЧ;
- ограничить полученное отношение по условию ОТД_КОЛ > 50;
- спроецировать результат предыдущей операции на атрибут СОТР_ИМЯ, СОТР_НОМ.

Мы четко сформулировали последовательность шагов выполнения запроса, каждый из которых соответствует одной реляционной операции. Если же сформулировать тот же запрос с использованием реляционного исчисления, которому посвящается этот раздел, то мы получили бы формулу, которую можно было бы прочитать, например, следующим образом: Выдать СОТР_ИМЯ и СОТР_НОМ для сотрудников таких, что существует отдел с таким же значением ОТД_НАЧ и значением ОТД_КОЛ большим 50.

Во второй формулировке мы указали лишь характеристики результирующего отношения, но ничего не сказали о способе его формирования. В этом случае система должна сама решить, какие операции и в каком порядке нужно выполнить над отношениями СОТРУДНИКИ и ОТДЕЛЫ. Обычно говорят, что алгебраическая формулировка является процедурной, т.е. задающей правила выполнения запроса, а логическая - описательной (или декларативной), поскольку она всего лишь описывает свойства желаемого результата. Как мы указывали в начале лекции, на самом деле эти два механизма эквивалентны и существуют не очень сложные правила преобразования одного формализма в другой.

2.1. Кортежные переменные и правильно построенные формулы

Реляционное исчисление является прикладной ветвью формального механизма исчисления предикатов первого порядка. Базисными понятиями исчисления являются понятие переменной с определенной для нее областью допустимых значений и понятие правильно построенной формулы, опирающейся на переменные, предикаты и кванторы.

В зависимости от того, что является областью определения переменной, различаются исчисление кортежей и исчисление доменов. В исчислении кортежей областями определения переменных являются отношения базы данных, т.е. допустимым значением каждой переменной является кортеж некоторого отношения. В исчислении доменов областями определения переменных являются домены, на которых определены атрибуты отношений базы данных, т.е. допустимым значением каждой переменной является значение некоторого домена. Мы рассмотрим более подробно исчисление кортежей, а в конце лекции коротко опишем особенности исчисления

доменов.

В отличие от раздела, посвященного реляционной алгебре, в этом разделе нам не удастся избежать использования некоторого конкретного синтаксиса, который мы, тем не менее, формально определять не будем. Необходимые синтаксические конструкции будут вводиться по мере необходимости. В совокупности, используемый синтаксис близок, но не полностью совпадает с синтаксисом языка баз данных QUEL, который долгое время являлся основным языком СУБД Ingres.

Для определения кортежной переменной используется оператор RANGE. Например, для того, чтобы определить переменную СОТРУДНИК, областью определения которой является отношение СОТРУДНИКИ, нужно употребить конструкцию

```
RANGE СОТРУДНИК IS СОТРУДНИКИ
```

Как мы уже говорили, из этого определения следует, что в любой момент времени переменная СОТРУДНИК представляет некоторый кортеж отношения СОТРУДНИКИ. При использовании кортежных переменных в формулах можно ссылаться на значение атрибута переменной (это аналогично тому, как, например, при программировании на языке Си можно сослаться на значение поля структурной переменной). Например, для того, чтобы сослаться на значение атрибута СОТР_ИМЯ переменной СОТРУДНИК, нужно употребить конструкцию СОТРУДНИК.СОТР_ИМЯ.

Правильно построенные формулы (WFF - Well-Formed Formula) служат для выражения условий, накладываемых на кортежные переменные. Основой WFF являются простые сравнения (comparison), представляющие собой операции сравнения скалярных значений (значений атрибутов переменных или литерально заданных констант). Например, конструкция "СОТРУДНИК.СОТР_НОМ = 140" является простым сравнением. По определению, простое сравнение является WFF, а WFF, заключенная в круглые скобки, является простым сравнением.

Более сложные варианты WFF строятся с помощью логических связок NOT, AND, OR и IF ... THEN. Так, если form - WFF, а comp - простое сравнение, то NOT form, comp AND form, comp OR form и IF comp THEN form являются WFF.

Наконец, допускается построение WFF с помощью кванторов. Если form - это WFF, в которой участвует переменная var, то конструкции EXISTS var (form) и FORALL var (form) представляют wff.

Переменные, входящие в WFF, могут быть свободными или связанными. Все переменные, входящие в WFF, при построении которой не использовались кванторы, являются свободными. Фактически, это означает, что если для какого-то набора значений свободных кортежных переменных при вычислении WFF получено значение true, то эти значения кортежных переменных могут входить в результирующее отношение. Если же имя переменной использовано сразу после квантора при построении WFF вида EXISTS var (form) или FORALL var (form), то в этой WFF и во

всех WFF, построенных с ее участием, var - это связанная переменная. Это означает, что такая переменная не видна за пределами минимальной WFF, связавшей эту переменную. При вычислении значения такой WFF используется не одно значение связанной переменной, а вся ее область определения.

Пусть СОТР1 и СОТР2 - две кортежные переменные, определенные на отношении СОТРУДНИКИ. Тогда, WFF EXISTS СОТР2 (СОТР1.СОТР_ЗАРП > СОТР2. СОТР_ЗАРП) для текущего кортежа переменной СОТР1 принимает значение true в том и только в том случае, если во всем отношении СОТРУДНИКИ найдется кортеж (связанный с переменной СОТР2) такой, что значение его атрибута СОТР_ЗАРП удовлетворяет внутреннему условию сравнения. WFF FORALL СОТР2 (СОТР1. СОТР_ЗАРП > СОТР2.СОТР_ЗАРП) для текущего кортежа переменной СОТР1 принимает значение true в том и только в том случае, если для всех кортежей отношения СОТРУДНИКИ (связанных с переменной СОТР2) значения атрибута СОТР_ЗАРП удовлетворяют условию сравнения.

На самом деле, правильнее говорить не о свободных и связанных переменных, а о свободных и связанных вхождениях переменных. Легко видеть, что если переменная var является связанной в WFF form, то во всех WFF, включающих данную, может использоваться имя переменной var, которая может быть свободной или связанной, но в любом случае не имеет никакого отношения к вхождению переменной var в WFF form. Вот пример:

```
EXISTS СОТР2 (СОТР1.СОТР_ОТД_НОМ = СОТР2.СОТР_ОТД_НОМ) AND

  FORALL СОТР2 (СОТР1.СОТР_ЗАРП > СОТР2.СОТР_ЗАРП)
```

Здесь мы имеем два связанных вхождения переменной СОТР2 с совершенно разным смыслом.

2.2. Целевые списки и выражения реляционного исчисления

Итак, WFF обеспечивают средства формулировки условия выборки из отношений БД. Чтобы можно было использовать исчисление для реальной работы с БД, требуется еще один компонент, который определяет набор и имена столбцов результирующего отношения. Этот компонент называется целевым списком (target_list).

Целевой список строится из целевых элементов, каждый из которых может иметь следующий вид:

- var.attr, где var - имя свободной переменной соответствующей WFF, а attr - имя атрибута отношения, на котором определена переменная var;
- var, что эквивалентно наличию подсписка var.attr1, var.attr2, ..., var.attrn, где attr1, attr2, ..., attrn включает имена всех атрибутов определяющего отношения;
- new_name = var.attr; new_name - новое имя соответствующего атрибута

результирующего отношения.

Последний вариант требуется в тех случаях, когда в WFF используются несколько свободных переменных с одинаковой областью определения.

Выражением реляционного исчисления кортежей называется конструкция вида target_list WHERE wff. Значением выражения является отношение, тело которого определяется WFF, а набор атрибутов и их имена - целевым списком.

2.3. Реляционное исчисление доменов

В исчислении доменов областью определения переменных являются не отношения, а домены. Применительно к базе данных СОТРУДНИКИ-ОТДЕЛЫ можно говорить, например, о доменных переменных ИМЯ (значения - допустимые имена) или НОСОТР (значения - допустимые номера сотрудников).

Основным формальным отличием исчисления доменов от исчисления кортежей является наличие дополнительного набора предикатов, позволяющих выражать так называемые условия членства. Если R - это n-арное отношение с атрибутами a_1, a_2, ..., a_n, то условие членства имеет вид

```
R (ai1:vi1, ai2:vi2, ..., aim:vim) (m <= n),
```

где v_{ij} - это либо литерально задаваемая константа, либо имя кортежной переменной. Условие членства принимает значение true в том и только в том случае, если в отношении R существует кортеж, содержащий указанные значения указанных атрибутов. Если v_{ij} - константа, то на атрибут a_{ij} задается жесткое условие, не зависящее от текущих значений доменных переменных; если же v_{ij} - имя доменной переменной, то условие членства может принимать разные значения при разных значениях этой переменной.

Во всех остальных отношениях формулы и выражения исчисления доменов выглядят похожими на формулы и выражения исчисления кортежей. В частности, конечно, различаются свободные и связанные вхождения доменных переменных.

Для примера сформулируем с использованием исчисления доменов запрос "Выдать номера и имена сотрудников, не получающих минимальную заработную плату" (будем считать для простоты, что мы определили доменные переменные, имена которых совпадают с именами атрибутов отношения СОТРУДНИКИ, а в случае, когда требуется несколько доменных переменных, определенных на одном домене, мы будем добавлять в конце имени цифры):

```
СОТР_НОМ, СОТР_ИМЯ WHERE EXISTS СОТР_ЗАРП1

  (СОТРУДНИКИ (СОТР_ЗАРП1) AND
```

```
СОТРУДНИКИ (СОТР_НОМ, СОТР_ИМЯ, СОТР_ЗАРП) AND

СОТР_ЗАРП > СОТР_ЗАРП1)
```

Реляционное исчисление доменов является основой большинства языков запросов, основанных на использовании форм. В частности, на этом исчислении базировался известный язык Query-by-Example, который был первым (и наиболее интересным) языком в семействе языков, основанных на табличных формах.

Литература и источники

1. Кузнецов С.Н., Центр Информационных Технологий. "Основы современных баз данных"
2. Дейт К. Введение в системы баз данных. М., 1998.
3. Пушников А.Ю. Введение в системы управления базами данных. Часть 1. Реляционная модель данных: Учебное пособие/Изд-е Башкирского ун-та. - Уфа, 1999. - 108 с. , а также CITFORUM.RU
4. Пушников А.Ю. Введение в системы управления базами данных. Часть 2. Нормальные формы отношений и транзакции: Учебное пособие/Изд-е Башкирского ун-та. - Уфа, 1999. - 138 с. а также CITFORUM.RU

.

1.1.4 Автоматизация обработки информации

Вопросы

1. Понятие экономической и автоматизированной информационной системы.
2. Понятие БД и СУБД. функциональные возможности и классификация СУБД.

1.1.4.1 Понятие экономической и автоматизированной информационной системы.

Информационная система *(ИС) – коммуникационная система по сбору, передаче, переработке информации об объекте, снабжающая работников различного ранга информацией для реализации функций управления.*

В зависимости от степени (уровня) автоматизации выделяют *ручные, автоматизированные* и *автоматические информационные системы.*

Ручные ИС *характеризуются тем, что все операции по переработке информации выполняются человеком.*

Автоматизированные ИС *– часть функций (подсистем) управления или обработки данных осуществляется автоматически, а часть –*

человеком.

***Автоматические ИС** – все функций управления и обработки данных осуществляется техническими средствами без участия человека.*

***Экономическая ИС (ЭИС) –** система для хранения, поиска и выдачи экономической информации по запросам пользователей.*

Классификация ЭИС

По уровню применения и административному делению различают ЭИС:

1. государства;
2. области;
3. района;
4. города;
5. предприятия;
6. подразделения.

По сфере применения – ЭИС:

1. банковские ИС;
2. ИС фондового рынка;
3. налоговые ИС;
4. страховые ИС;
5. статистические ИС;
6. ИС промышленных предприятий и организаций (особое место по значимости и распространенности среди них занимают *бухгалтерские ИС*).

ЭИС [систему] можно разделить на 2 части: *обеспечивающую* и *функциональную*.

Обеспечивающая часть ЭИС (включает в себя следующие виды обеспечения: *информационное, техническое, программное, правовое* и другие).

1.1.4.2 Понятие БД и СУБД. функциональные возможности и классификация СУБД.

***База данных** (БД) – это организованная структура, предназначенная для*

хранения информации.

Система управления базами данных *(СУБД) – это программа (или комплекс программ), предназначенная для создания структуры новой базы данных, наполнения ее содержимым, редактирования содержимого и визуализации информации.*

Под *визуализацией информации* базы понимается отбор отображаемых данных по заданному критерию, их упорядочение, оформление и выдача на устройства вывода или передача по каналам связи.

Функциональные возможности СУБД

СУБД позволяют осуществлять следующие *функции*:

- Создавать БД в виде файла (или файлов) на внешнем носителе;
- Загружать, редактировать БД;
- Осуществлять поиск информации по запросу;
- Упорядочивать информацию в БД;
- Выводить информацию из БД;
- Формировать отчёты и другие объекты на основании БД;
- Обеспечивать защиту данных от разрушения и несанкционированного доступа;
- Сохранять имеющиеся данные при изменении структуры БД;
- Сохранять имеющиеся программы при изменении структуры БД.

Классификация СУБД

1. **По уровню использования** СУБД делят на:

- Профессиональные (промышленные);
- Настольные (персональные).

Профессиональные СУБД представляют собой программную основу для разработки автоматизированных систем управления (АСУ) крупными экономическими объектами. На основе этих СУБД создаются АСУ банков, крупных предприятий, отраслей. К этой группе относятся Oracle, DB2, Informix, Progress, Ingress.

Персональные СУБД ориентированы на решение задач отдельного пользователя или небольшой группы пользователей, и предназначены для использования на ПК.

Основными характеристиками настольных СУБД являются:

- простота эксплуатации;
- невысокие требования к аппаратным ресурсам компьютера.

К ним относятся: Dbase, FoxBase, FoxPro, Clipper, MS Access, Approach.

2. **По степени универсальности** различают СУБД:

- Общего назначения;
- Специального назначения;

СУБД общего назначения не имеют ориентации на какую-то конкретную область. Они обладают средствами настройки на конкретную БД.

Однако в некоторых предметных областях СУБД общего назначения не позволяют эффективно организовать работу с конкретной БД. В таких случаях создаются ***специализированные СУБД*** для конкретного применения, например, СУБД Imbase, используемая для автоматизации проектных и конструкторских работ.

3. **По типу моделей данных:**

- иерархическая (IMS фирмы IBM);
- сетевая (IDS фирмы General Electric)
- реляционные (MS Access).

4. **По способу распределения данных:**

- централизованные;
- децентрализованные;
- смешанные.

Централизованная организация БД является самой простой. На одном компьютере – сервере находится единственная БД. Все операции с БД обеспечиваются этим сервером.

Децентрализованная организация БД предполагает разбиение БД на несколько

физически распределенных БД. Каждый клиент пользуется своей БД, которая может быть либо копией общей БД, либо ее частью (используется дублирование).

Возможна смешанная организация, которая объединяет два этих способа.

Концепции сетевой обработки данных *файл-сервер* и *клиент-сервер*

Концепция ***файл-сервер*** предполагает наличие компьютера, выделенного под файловый сервер, на котором находится сетевая ОС и централизованно хранимые файлы. Для этой архитектуры характерен коллективный доступ к общей базе данных на файловом сервере. При обновлении файла одним из пользователей, он блокируется для доступа другим пользователям. Запрошенные данные передаются по сети с файлового сервера на рабочие станции, где их обработка выполняется средствами СУБД.

В концепции файл-сервер вся тяжесть выполнения запросов к БД и управления целостностью БД ложится на СУБД пользователя.

Концепция ***клиент-сервер*** подразумевает разделение функций обработки данных между клиентом – рабочей станцией и машиной-сервером баз данных, где обработку осуществляет установленная там СУБД. Запрос на обработку данных выдается клиентом и передается по сети на сервер баз данных, где осуществляется поиск. Обработанные данные передаются по сети от сервера к клиенту. (Спецификой архитектуры клиент-сервер является использование языка SQL для запросов к БД, что обеспечивает работу с общими данными из разнотипных приложений клиентов сети.)

1.1.5 Проектирование БД. Организация связей между объектами БД

Вопросы

1. Требования, предъявляемые к БД
2. Модели данных. Компьютерная реляционная БД
3. Этапы проектирования баз данных

1.1.5.1 Требования, предъявляемые к БД

Первые автоматизированные информационные системы (АИС) появились в конце 50-х начале 60-х годов. Данные в первых АИС хранились в файлах последовательного доступа. Последовательному доступу присущи следующие недостатки:

- низкая скорость обработки;
- недостаточная возможность управления данными;
- большой объем дублирования данных;
- ограниченные возможности по контролю данных;
- жесткая связь данных и прикладных программ.

БД представляет собой поименованную совокупность данных, отражающую состояние объекта или множества объектов, их свойств и взаимоотношений.

Требования, предъявляемые к БД:

1. Обеспечение *независимости программ от данных* (это позволяет сохранять существующие программы при модификации БД).
2. Возможность *модификации структуры БД без повторного ввода имеющихся данных*.
3. *Отсутствие избыточности* (минимальная избыточность), что позволяет однократно вводить и корректировать информацию.
4. Обеспечение *непротиворечивости данных и целостности* БД. Это значит, что обработка данных должна быть построена таким образом, чтобы в случае возникновения физических сбоев или попыток неверного изменения данных или структуры данных система смогла восстановить и согласовать данные без потерь.

   Важнейшим механизмом поддержания целостности БД является *транзакция.*

   Транзакция – это совокупность операций с БД, которая должна быть выполнена до конца, чтобы БД не оказалась в противоречивом состоянии (например, изменение в БД при переводе студента на следующий курс, удалении из БД сведений о сотруднике при увольнении).
5. Возможность *многоаспектного доступа*, т.е. различные пользователи могут

представлять себе данные, хранимые в БД, по-разному, в соответствии с собственными потребностями.

6. Возможность *осуществлять всевозможные выборки данных* и *использовать их различными приложениями* пользователя.
7. Разграничение *доступа* к данным *и защита от несанкционированного доступа*.
8. *Наличие языка запроса высокого уровня*, ориентируемого на конечного пользователя-непрограммиста.

1.1.5.2 Модели данных. Компьютерная реляционная БД

Модель данных – это совокупность взаимосвязанных структур данных и операций над этими структурами.

По способу установления связей между данными наиболее распространёнными являются *иерархическая, сетевая* и *реляционная* модели данных.

Иерархическая модель представляется в виде дерево-графа, где возможны только односторонние связи от старших вершин к младшим, т.е. любой объект может подчиняться только одному объекту старшего уровня. *Достоинство* этой модели – простота доступа к данным. *Недостаток* – жесткая фиксированность взаимосвязей между элементами данных.

Сетевая модель обеспечивает прямой доступ к любому объекту (каждый элемент связан со всеми остальными).

Сетевая БД состоит из набора записей и связей между ними. Запись в сетевой модели в отличие от иерархической может иметь множество как подчиненных ей записей, так и записей, которым она подчинена.

По сравнению с иерархичной моделью сетевая обладает большей гибкостью, но ее *недостаток* – трудности реализации этой модели в полном объёме.

Реляционная модель (relation – отношение) считается простейшей и наиболее привычной формой представления данных в виде таблиц.

Под *реляционным отношением* (таблицей) будем понимать двумерный массив типа *объекты-признаки*, обладающий следующими свойствами:

- все столбцы таблицы однородны;
- все столбцы таблицы имеют уникальные имена:

- в таблице нет одинаковых строк;
- все строки таблицы имеют одну и ту же структуру, т.е. одно и то же количество атрибутов с соответственно одинаковыми именами;
- в операциях с таблицами строки и столбцы могут просматриваться в любом порядке без относительно к их информационному содержанию и смыслу.

Столбец таблицы со значениями соответствующего атрибута называется ***полем***, а строка со значением разных атрибутов называется ***записью***.

Одно или несколько полей, значения которых однозначно идентифицируют строку таблицы, является ***ключом*** таблицы (или ***первичным ключом***). *Первичный ключ* называется ***простым***, если он состоит из одного поля, или ***составным***, если он состоит из нескольких полей.

Кроме первичных ключей в таблицах могут присутствовать *вторичные ключи*. ***Вторичный ключ*** (еще его называют ***индекс***) – это такой ключ, значения которого могут повторяться в разных строках (записях). В отличие от первичного ключа, по которому всегда отыскивается только одна строка, по вторичному ключу может отыскиваться группа строк с одинаковым значениям вторичного ключа.

Типы связи между таблицами

В реляционных БД поддерживается четыре типа отношений (связей) между таблицами:

1. Отношение ***один-к-одному*** означает, что каждая запись в одной таблице соответствует одной записи в другой таблице. (Связь уникальна как в одном направлении, так и в обратном направлении.)
2. Отношение ***один-ко-многим*** означает, что каждой записи в одной таблице соответствует несколько записей в другой таблице. (В одном направлении связь не является уникальной, а в обратном направлении – уникальна.)
3. Отношение ***многие-к-одному*** означает, что нескольким записям в одной таблице соответствует одна запись в другой таблице. (В одном направлении связь уникальна, а в обратном направлении – нет.)
4. Отношение ***многие-ко-многим*** означает, что нескольким записям в одной

таблице соответствует несколько записей в другой таблице и наоборот. (Связь не является уникальной в обоих направлениях.)

Нормализация баз данных

При проектировании реляционной БД необходимо решить вопрос о наиболее эффективной структуре данных. Основные цели, которые при этом преследуются:

- обеспечить быстрый доступ к данным в таблицах;
- исключить ненужное повторение данных, которое может являться причиной ошибок при вводе и нерационального использования дискового пространства компьютера;
- обеспечить целостность данных таким образом, чтобы при изменении одних объектов автоматически происходило соответствующее изменение связанных с ними объектов.

Процесс уменьшения избыточности информации в БД называется ***нормализацией***.

Таблица в ***первой нормальной форме*** должна удовлетворять следующим требованиям:

1. Таблица не должна иметь повторяющихся записей.
2. В таблице не должно быть повторяющихся полей или групп полей.

Таблица во ***второй нормальной форме*** должна удовлетворять следующим требованиям:

1. Она должна находиться в первой нормальной форме.
2. Любое ключевое поле однозначно идентифицируется полным набором ключевых полей.

Таблица в ***третей нормальной форме*** должна удовлетворять следующим требованиям:

1. Она должна находиться во второй нормальной форме.
2. Ни одно из неключевых полей таблицы не идентифицируется с помощью другого неключевого поля

1.1.5.3 Этапы проектирования баз данных

Проектирование базы данных можно разбить на три этапа – ***концептуальное, логическое*** и ***физическое***.

1. На этапе ***концептуального проектирования*** осуществляется сбор, анализ и упорядочивание требований к данным, построение модели данных. Концептуальная модель создаётся как обобщение пользовательских представлений и включает в себя совокупность всех данных и требований к ним.
2. Этап ***логического проектирования*** включает в себя выбор конкретной модели в СУБД и отображение концептуального представления в логическую модель, основанную на структурах, характерных для выбранной СУБД.

Для реляционной БД этот этап включает разработку структуры записей данных, организацию их в таблицы, определение связей между таблицами и полей для реализации этих связей. На этом же этапе разрабатывается словарь данных, в который включается информация о базе в целом, обо всех ее таблицах, полях, ключах, а также используемых кодах.

3. На этапе ***физического проектирования*** логическая модель реализуется средствами выбранной СУБД: создаются объекты БД и связи между ними. На этом же этапе выбираются физические устройства для размещения БД и ее копий.

1.1.6 Общая характеристика СУБД MS Access

Вопросы

1. Общая характеристика СУБД MS Access
2. Описание полей базы данных

1.1.6.1 1. Общая характеристика СУБД MS Access

СУБД MS Access – это программа, позволяющая легко и быстро проектировать базы данных, состоящие из 7 типов объектов: *таблиц, запросов, форм, отчетов, страниц, макросов, модулей*.

Таблица – это основная структура, предназначенная для хранения информации. Ее строки называются записями, а столбцы - полями базы данных. Каждое поле имеет определенный тип данных (текст, число, дата и т.д.), длину и уникальное имя, которое идентифицирует (однозначно определяет) хранящуюся в этом поле информацию.

Записи идентифицируются по одному или нескольким полям, которые однозначно определяют хранящуюся в этой записи информацию. Такие поля называются ключевыми полями или ключом.

Запрос – это требование на отбор данных, хранящихся в таблицах, на выполнение определенных действий с данными. Запрос позволяет создать набор из записей, находящихся в разных таблицах, и использовать его как источник данных для формы или отчета. Кроме того, запрос дает возможность вносить изменения в саму БД. Запросы служат для анализа данных.

Форма – это созданный на экране шаблон для ввода, просмотра и редактирования записей БД (как бумажный бланк).

Отчет – это средство для отображения на экране или принтере информации из БД в виде, удобном для ее восприятия и анализа пользователем.

Макрос – это последовательность действий (макрокоманд) для автоматизации выполнения операций (в среде без программирования).

Модуль – это программа для обработки данных, написанная на языке Visual Basic для приложений (VBA). Access позволяет создавать эффективные модули для работы с БД, содержащие меню, диалоговые окна и командные кнопки.

Страница – это специальные объекты баз данных, позволяющие просматривать информацию из базы данных, через Web-страницы на удаленных компьютерах (*страницы доступа к данным*).

Таблицы, запросы, формы, отчеты, страницы, макросы, модули – называются *объектами БД*. Объекты БД хранятся в едином файле - файле БД, имеющем расширение *.mdb*. Это упрощает их перенос с компьютера на компьютер, облегчает создание связанных объектов, проверку целостности данных.

В Access имеется несколько средств создания каждого из основных объектов базы. Их можно классифицировать как:

- *ручные* (создание и редактирование объектов в режиме *Конструктора*);
- *автоматизированные* (создание объектов с помощью *Мастеров);*
- *автоматические* – средства ускоренного создания простейших объектов.

Ручные средства являются наиболее трудоемкими, но обеспечивают

максимальную гибкость; *автоматизированные* и *автоматические* средства являются наиболее производительными, но и наименее гибкими.

Конструктор предоставляет пользователю ряд инструментальных средств, с помощью которых можно быстро и просто создавать и модифицировать объекты БД.

Мастер делает это по-другому: задает пользователю ряд вопросов и на основе его ответов строит вполне законченный объект БД.

В учебных целях для создания разных объектов целесообразно пользоваться разными средствами.

При разработке учебных *таблиц* и *запросов* рекомендуется использовать ручные средства – работать в режиме *Конструктора*. Использование *Мастеров* ускоряет работу, но не способствует освоению понятий и методов.

При разработке учебных *форм, отчетов* и *страниц* доступа к данным наоборот лучше пользоваться автоматизированными средствами, предоставляемыми *Мастерами*, так как для данных объектов большую роль играет внешний вид. Дизайн этих объектов весьма трудоемок, поэтому, его лучше поручить программе, и сосредоточиться на содержательной части работы.

MS Access располагает разнообразными графическими средствами для оформления таблиц, форм, отчетов, страниц.

Кроме проектирования объектов БД, MS Access осуществляет управление БД:

- защиту;
- резервирование;
- репликацию (создание специальных копий БД, с которыми пользователи могут одновременно работать на разных компьютерах);
- восстановление;
- сжатие;
- повышение быстродействия БД;
- просмотр сведений о БД;
- поиск файла БД по свойствам БД;
- экспорт и импорт данных.

1.1.6.2 2. Описание полей базы данных

Чтобы пользователь мог ввести свои данные в компьютерную БД, он должен описать эти поля согласно требованиям Access:

- присвоить имена полям;
- указать, какого типа данные допускается вводить в каждое поле;
- каждому полю дать определенные свойства, которые позволят управлять сохранением, обработкой и отображением данных поля.

Полям присваиваются имена с учетом следующих требований:

- имя должно содержать не более 64 символов;
- оно может включать любую комбинацию букв, цифр, пробелов и специальных символов, за исключением точки (.), восклицательного знака (!), апострофа (‘) и квадратных скобок ([ ]);
- имя не должно начинаться с символа пробела;
- оно не может включать управляющие символы (с кодами ASCII от 0 до 31).

Типы данных

В поля БД можно вводить данные следующих типов.

Текстовый. В поля такого типа помещают текст или комбинацию текстовых и числовых значений (например, адреса) длиной до 255 символов. Кроме того, в такие поля записывают числовые значения, для которых не предполагается выполнение расчетов, такие как телефонные или инвентарные номера или почтовые индексы.

Поле МЕМО. Длинный текст (до 65 535 символов), например, примечания или описания.

Числовой. Числовые данные, используемые в математических вычислениях, за исключением денежных расчетов (для последних определен тип "Денежный").

Денежный. Денежные значения. Тип "Денежный" позволяет проводить вычисления без округления значений. Максимальная точность составляет 15 знаков слева от десятичной запятой и 4 знака справа от запятой.

Дата/время. Значения даты или времени.

Счетчик. Уникальные последовательные (с шагом 1) или случайные номера,

автоматически вставляемые при вставке записи в БД.

Логический. Поля, которые могут иметь только одно значение из пары значений, таких как Да/Нет, Истина/Ложь или Вкл/Выкл.

Поле объекта OLE. Объекты, созданные в других приложениях, которые могут быть связаны или внедрены в таблицу Microsoft Access (например документы Microsoft Word, электронные таблицы Microsoft Excel, рисунки, звукозапись или другие данные в двоичном формате).

Гиперссылка. Специальное поле для хранения адресов (URL и UNC) Web-объектов Интернета.

Мастер подстановок. Создает поле, позволяющее выбрать с помощью раскрывающегося списка значение из другой таблицы или из списка значений.

Набор допустимых свойств для поля зависит от того, какого типа данные будут храниться в поле. Наборы свойств для полей с часто используемыми типами данных приведены в табл. 4.1.

Таблица 4.1.

Свойства часто используемых полей

Числовой тип данных	**Текстовый тип данных**	**Тип данных Дата/время**
Размер поля	Размер поля	–
Формат поля	Формат поля	Формат поля
Число десятичных знаков	–	–
Маска ввода	Маска ввода	Маска ввода
Подпись	Подпись	Подпись
Значение по умолчанию	Значение по умолчанию	Значение по умолчанию
Условие на значение	Условие на значение	Условие на значение
Сообщение об ошибке	Сообщение об ошибке	Сообщение об ошибке
Обязательное поле	Обязательное поле	Обязательное поле
–	Пустые строки	–

Индексированное поле	Индексированное поле	Индексированное поле

Размер поля. Для числового поля допустимыми являются следующие значения:

- целые числа от 0 до 255. Данный размер поля обозначается в Access как байт;
- целые числа от -32 768 до 32 767 (обозначение размера – целое);
- целые числа от -2 147 483 648 до 2 147 483 647 (длинное целое);
- числа с плавающей точкой от -3.402823E38 до 3.402823E38, в дробной части до 7 знаков (с плавающей точкой (4 байт));
- числа с плавающей точкой от -1.79769313486232E308 до 1.79769313486232E308, в дробной части - до 15 знаков (с плавающей точкой (8 байт)).

Формат поля. Это свойство определяет способ отображения текста, чисел, дат и значений времени на экране и на печати.

Число десятичных знаков. Дает возможность указывать для чисел количество дробных знаков.

Маска ввода. Задает маску ввода, облегчающую ввод данных в поле.

Подпись. Определяет текст, который выводится в подписях полей в таблицах, запросах, формах, отчетах.

Значение по умолчанию. Позволяет указать значение, автоматически вводящееся в поле при создании новой записи.

Условие на значение. Определяет требования к данным, вводящимся в поле.

Сообщение об ошибке. Позволяет указать текст сообщения, выводящегося на экран, если введенные данные нарушают условие, определенное в свойстве *Условие на значение.*

Обязательное поле. Указывает, требует ли поле обязательного ввода значения.

Пустые строки. Определяет, допускается ли ввод в текстовое поле пустых строк (не содержащих символов).

Индексированное поле. Задает индекс для поля, ускоряющий поиск и сортировку в таблице.

1.1.7 Среда СУБД MS Access. Проектирование таблиц и работа с ними

Вопросы

1. Создание новой базы данных
2. Создание связей между таблицами

1.1.7.1 1. Создание новой базы данных

Способы создания новой базы данных

В *Access* поддерживается 2 способа создания базы данных:

1-й способ: *Создание пустой базы данных*, а затем добавление в нее таблиц, запросов, форм, отчетов, страниц, макросов, модулей (объектов).

2-й способ: *Создание с помощью мастера* базы данных выбранного из списка типа со всеми необходимыми таблицами, отчетами и др. объектами. *Access* содержит набор различных типовых баз данных.

Создание пустой базы данных

После запуска *Access* открывается диалоговое окно, с помощью которого можно приступить к созданию новой базы данных (*Новая база данных*), вызвать мастер создания (*Мастера, страницы и проекты баз данных*) или открыть существующую базу данных (*Открыть базу данных*). Эти режимы можно вызвать также выполнением команд *Файл-Создать базу данных, Файл-Открыть базу данных.*

Способы создания таблиц

Процесс создания новой БД начинается с создания структуры таблицы.

Таблица это основной (обязательный) объект СУБД *Access*, предназначенный для хранения информации. Ее строки называются записями, а столбцы – полями БД. Каждая таблица имеет уникальное имя в БД и содержит информацию о каких-либо объектах или свойствах объектов одного типа.

Каждое поле содержит однородные данные и имеет уникальное имя, определенный тип, длину и другие свойства.

Таблицы создаются в окне уже созданной (пустой или содержащей объекты) БД – *Имя_файла:База данных.*

В *Access* имеется *5 способов создания таблицы*:

1. *Режим таблицы* – упрощенный режим создания таблицы. Однако в этом режиме отсутствует доступ ко многим управляющим элементам, которые позволяют указывать свойства полей таблицы. Этот режим подходит для создания "черновика" таблицы, наброска таблицы.
2. *Конструктор* – предоставляет наиболее широкие возможности по заданию свойств создаваемой таблицы и ее полей.
3. *Мастер таблиц* – позволяет создавать новые таблицы на основе готовых шаблонов, имеющихся в *Access*.
4. *Импорт таблицы* – позволяет выполнить создание новой таблицы вставкой существующей таблицы из внешнего файла.
5. *Связь с таблицами* – позволяет выполнить создание новой таблицы методом связывания с таблицей из внешнего файла.

Создание в таблицы в режиме Конструктора

Режим *Конструктора* предоставляет наиболее широкие возможности по определению параметров создаваемой таблицы.

После запуска *Access* открывается диалоговое окно, с помощью которого можно приступить к созданию новой базы данных (*Новая база данных*), вызвать мастер создания (*Мастера, страницы и проекты баз данных*) или открыть существующую базу данных (*Открыть базу данных*). Эти режимы можно вызвать также выполнением команд *Файл-Создать базу данных, Файл-Открыть базу данных.*

Для создания таблицы вручную, следует использовать режим *Создание таблицы в режиме конструктора* или на *Панели инструментов* кнопку *Создать,* затем режим *Конструктор*. В результате откроется окно конструктора, предназначенное для ввода структуры создаваемой таблицы. В верхней части окна в каждой строке задается имя поля, его тип и описание (если необходимо). В нижней части задаются свойства текущего поля.

Для создания таблицы в режиме *Конструктора* необходимо:

1. Выбрать объект *Таблицы* в диалоговом окне *База данных.* (Это окно

появляется после создания нового или открытия уже имеющегося на диске файла базы данных.)

2. Выбрать режим *Создание таблицы в режиме конструктора.*

   (или кнопку *Создать* на *Панели инструментов* и, в появившемся диалоговом окне *Новая таблица,* выбрать способ создания таблицы *Конструктор*).

3. В появившемся окне *Таблица* ввести в колонку *Имя поля* имена полей создаваемой таблицы.
4. В колонке *Тип данных* из раскрывающегося списка выбрать нужный тип данных для каждого имени поля.
5. В нижней части окна задать нужные свойства поля для каждого имени поля.
6. Установить признаки ключевых полей, выделив нужные поля и щелкнув мышью по кнопке *Ключевое поле* на *Панели инструментов* или командой *Правка- Ключевое поле,* (или щелчком правой кнопки мыши по нужному полю (или выделенным полям) и выбрав из контекстного меню команду *Ключевое поле*)

1.1.7.2 2.Создание связей между таблицами

Связь между таблицами устанавливает откос между совпадающими значениями полей разных таблиц. Поля, с помощью которых устанавливается связь, могут иметьразличные имена, но один тип и одинаковый размер данных.

Исключение из этого правила:

Можно установить между полем, имеющим тип данных счетчик и полем, имеющим тип данных числовой, размер длинное целое.

Связь 1:1 – устанавливается обычно по полю, являющимся ключевым.

Связь 1:M – обычно устанавливается по полю, которое является ключевым в основе таблицы и индексом, для которой установлено значение “да”; значения повторяются связями таблицы.

Связь M:M – в Access реализуется только через промежуточную таблицу.

Для создания связи используется диалог по схеме данных.

В окне связи присутствуют следующие флажки:

1) Обеспечение целостности данных

2) Каскадное обновление связанных полей

3) Каскадное удаление связанных полей

Если включается (1) флажок , то невозможно ввести в поле связанной таблицы значение , не содержащиеся в ключевом поле главной таблицы.

Если включается (2) флажок , то при изменении ключевого поля главной таблицы , в связанной оно изменяется автоматически.

Если включается (3) флажок – удаление автоматически из связанной таблицы.

1.2 Мультимедийные презентации

В разделе представлены ссылки на мультимедийные презентации по темам:

1. Понятия математической логики

- **Лекция**
- **Практическое занятие - часть 1**
- **Практическое занятие - часть 2**
- **Практическое занятие - часть 3**

2. Организация и хранение данных в базах данных

1.3 Электронные образовательные ресурсы по дисциплине

Для более полного и подробного изучения дисциплины рекомендуем обратиться к электронным образовательным ресурсам:

- материалы университетских репозиториев открытого доступа (http://rep.bntu.by/; http://elib.bsu.by; http://www.lib.grsu.by/cgi-bin/lib.cgi; http://ir.kneu.edu.ua:8080/ и др.)

- электронные архивы научной периодики (http://csl.bas-net.by/Web/Pages/magNAS.asp и др.)

2 Практический раздел

В практическом разделе представлены:
- задания для практических занятий;
- задания для самостоятельной работы.

Разделы "Теория множеств", "Математическая логика", "Реляционная алгебра" разработаны кандидатом технических наук, доцентом Чичко О. И.

2.1 Задания для практических занятий

В разделе представлены задания для практической работы с методическими указаниями для их выполнения.

2.1.1 Теория множеств

1) Доказать принцип двойственности: $C(A \cup B) = CA \cap CB$, $C(A \cap B) = CA \cup CB$.

2) Доказать равенства $A \cup (A \cap B) = A \cap (A \cup B) = A$.

3) Доказать равенства: а) $CCA = A$; б) $C\mathcal{I} = \varnothing$; в) $C\varnothing = \mathcal{I}$.

4) Доказать справедливость включения $(A \setminus B) \subset (A \setminus D) \cup (D \setminus B)$.

5) Определить множества $A \cup B$, $A \cap B$, $A \setminus B$, $B \setminus A$, $A \oplus B$, если:
а) $A = \{x: 0 < x < 2\}$, $B = \{x: 1 \le x \le 3\}$;
б) $A = \{x: x^2 - 3x < 0\}$, $B = \{x: x^2 - 4x + 3 \le 0\}$;
в) $A = \{x: |x - 1| < 2\}$, $B = \{x: |x - 1| + |x - 2| < 3\}$.

6) Имеем $A = \{(x, y): |x| + |y| < \delta\}$, $B = \{(x, y): \sqrt{x^2 + y^2} < \delta\}$, $D = \{(x, y): \max(|x|, |y|) < \delta\}$. Показать, что $A \subset B \subset D$.

7) Пусть $A = \{x: 2 \le x \le 4\}$, $B = \{y: 1 \le y \le 3\}$. Изобразить на плоскости xOy множество точек $A \times B$.

8) Показать, что семейство R, замкнутое относительно объединения и разности, является кольцом.

9) Показать, что семейство $R = \{\varnothing, O\}$, состоящее из непустого множества $\varnothing$ и пустого множества O, образует кольцо. Является ли это кольцо алгеброй?

10) Пусть множество $\mathcal{J} = \{\varnothing, I\}$ состоит из трех элементов, а $P(\mathcal{J})$ - семейство всех подмножеств множества $\mathcal{J}$.
а) Записать все алгебры, которые можно построить из элементов множества $P(\mathcal{J})$, и указать их единицы.
б) Описать все кольца, которые можно построить из элементов множества $P(\mathcal{J})$.
в) Описать все полукольца, которые можно построить из элементов множества $P(\mathcal{J})$ и которые не являются кольцами.

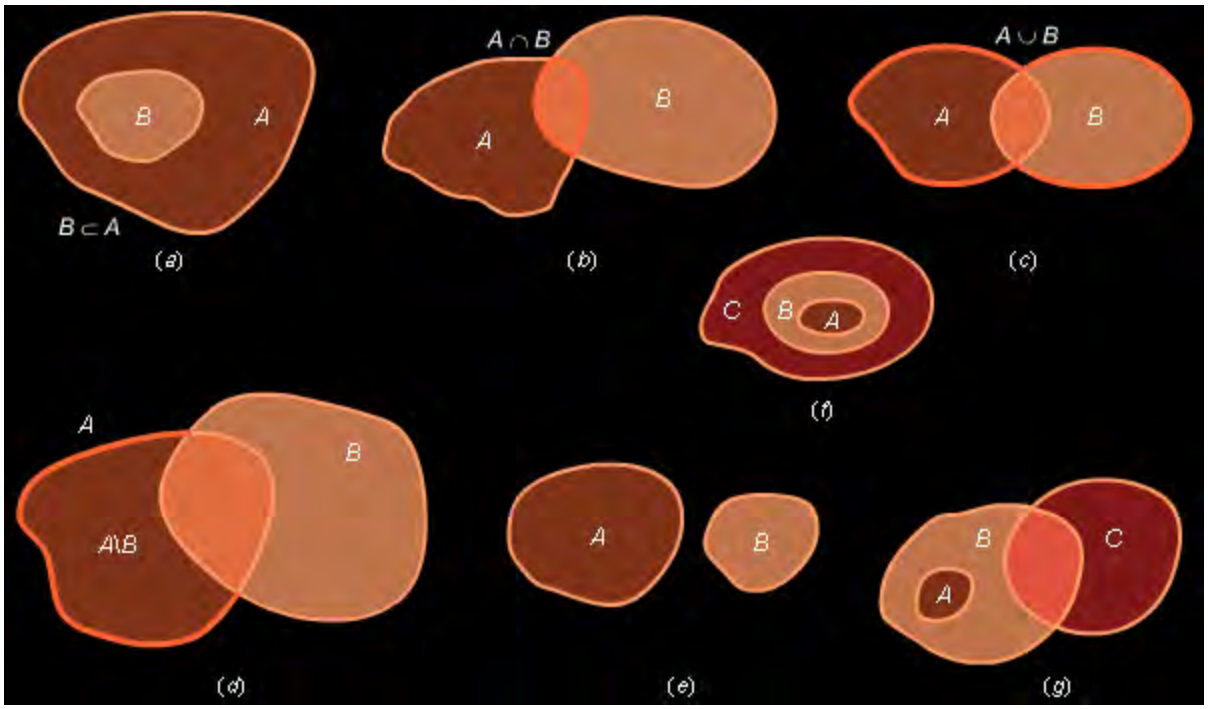
11) Доказать, что $(A \setminus B) \times (D \setminus E) = (A \times D) \setminus (B \times E)$.

(Информация с сайта: http://www.pm298.ru/reshenie/elem5.php)

12) Определить множества (http://cadmium.ru/content/view/754/45/)

1) $\{1, 2, 3\} \cup \{2, 3, 4\} = \{1, 2, 3, 4\}$;
2) $\{1, 2, 3\} \cap \{2, 3, 4\} = \{2, 3\}$;
3) $\{1, 2, 3\} \setminus \{2, 3, 4\} = \{1\}$;
4) $\{2, 3, 4\} \subset \{1, 2, 3, 4\} \Rightarrow \overline{\{1\}} = \{2, 3, 4\}$

13) Записать отношения, представленные на диаграмме Эйлера–Венна (http://webmath.exponenta.ru/s/c/planimetry/content/chapter16/section/paragraph1/theory.html):



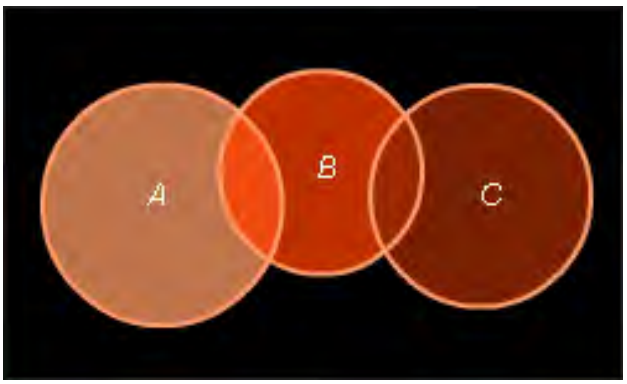
14) Найти объединение и пересечение множеств, если
$A = \{x \mid x \in \mathbb{R},\ 0 < x < 10\}$, $B = \{x \mid x \in \mathbb{R},\ -7 \le x < 2\}$.

Ответ. По определению, объединение множеств A и B состоит из элементов как множества A, так и B. Таким образом, $A \cup B$ состоит из всех действительных чисел, входящих либо в промежуток (0; 10), либо в промежуток [-7; 2), то есть является объединением этих промежутков. На числовой прямой отметим концы данных промежутков.

Объединением, очевидно, будет промежуток [-7; 10). Пересечением данных числовых промежутков будет промежуток с двойной штриховкой. Таким образом,

$$A \cup B = \{x \mid x \in \mathbb{R}, -7 \le x < 10\},\ A \cap B = \{x \mid x \in \mathbb{R}, 0 < x < 2\}$$

15) Изобразите с помощью диаграмм Эйлера-Венна следующую форму логического рассуждения: "Если некоторые a являются b, а некоторые b являются c, то некоторые a являются c".



Ответ. На языке теории множеств это означает, что если A ∩ B ≠ ∅ и B ∩ C ≠ ∅, то A ∩ C ≠ ∅. Но это не всегда верно, что и демонстрирует приведенная диаграмма.

Выполните упражнения(http://festival.1september.ru/articles/513260/):

1. Назовите известные вам названия множеств людей (например, команда).
2. Назовите известные вам названия множеств живых существ (например, табун).
3. Запишите множества, элементами которого являются: *а*) планеты солнечной системы; *б*) столицы государств; *в*) все двузначные числа; *г*) числа, делящиеся на 7.
4. Пусть *A* – множество чисел, на которые делится число 100 без остатка. Верна ли запись: .
5. Запишите множество *B*, элементами которого являются ученики 5 *А* класса. Выпишите два элемента принадлежащих множеству *B* и два элемента не принадлежащих множеству *B*.

6. Пусть даны множества $A = \{x | x - имя\ девочки\}$ и $B = \{x | x - имя\ мальчика\}$. Выпишите: *а*) два элемента принадлежащих множеству *A*, но не принадлежащих множеству *B*; *б*) два элемента принадлежащих множеству *B*, но не принадлежащих множеству *A*; *в)* два элемента принадлежащих и множеству *A* и множеству *B*; *г*) два элемента не принадлежащих ни множеству *A* ни множеству *B*.

7. Придумайте и запишите какое-нибудь множество. Выпишите два элемента принадлежащих этому множеству и два элемента не принадлежащих этому множеству.

8. Придумайте и запишите множество, не содержащее ни одного элемента (пустое множество).

Выполните упражнения (http://festival.1september.ru/articles/513260/):

1. Верна ли запись:
   $а)\ \{8, 12, 16, 20\} = \{12, 20, 16, 8\};\quad б)\ \{m, n, p, g\} = \{p, m, q, n\};$
   $в)\ \{3, 4, 3, 5\} = \{3, 4, 5\};\quad г)\ \{a, s, f, d, a, f\} = \{a, d, f, s\}?$

2. Запишите все множества, равные множеству $\{x, y, z, w\}$.

3. Запишите множества, равные данным:
   $а)\ \{2, 3, 2, 4, 2, 5\};\quad б)\ \{f, f, f, m, m, m\};\quad в)\ \{1, 1, 1, 1, 1, 1, 1\}.$

4. Придумайте и запишите множество, состоящее из пяти элементов. Выпишите все множества равные придуманному множеству. Сколько множеств получилось?

5. **Выполните упражнения (** http://festival.1september.ru/articles/513260/):

1. Даны множества:

   a. множество *A* учеников 5 класса нашей школы;

   b. множество *B* всех учеников нашей школы;

c. множество *C* учеников 5 класса нашей школы, посещающих бассейн;

d. множество *E* всех учащихся школ города Новокузнецка;

e. множество *K* учеников 5 математического класса нашей школы.

Верно ли что:

a. множество *A* есть подмножество множества *B*;

b. множество *A* есть подмножество множества *K*;

c. множество *B* есть подмножество множества *E*;

d. множество *K* есть подмножество множества *C*;

Запишите с помощью знака ⊂ названия множеств в таком порядке, чтобы каждое следующее множество было подмножеством предыдущего множества.

2. Для множества $P = \{8, 10, 12, 14\}$ выпишите все его подмножества.

Выполните упражнения (http://festival.1september.ru/articles/513260/):

1. Даны множества $A = \{3, 4, 5\}, \quad B = \{5, 6, 7, 8\},$ $C = \{2, 4, 8\}, \quad K = \{1, 3, 5, 7\}$. Найдите: а) $A \cap K$; б) $A \cap C$; в) $A \cap B$; г) $A \cap K \cap B$.

2. Найдите $A \cap B$, если а) $A = \{0, 1, 2, 3, 4\}$ и $B = \{1, 2, 3, 4, 5\}$; б) $A = \{x | x - двузначное число\}$ и $B = \{x | x - число\ меньшее\ 75\}$

Выполните упражнения (http://festival.1september.ru/articles/513260/):

1. Даны множества

. Найдите: *а*) ; *б*) ; *в*) ; *г*)

.

2. Найдите $A \cup B$ если $A = \{x | x - число\ меньшее\ 32\}$ и $B = \{x | x - число\ большее\ 7, но\ меньшее\ 45\}$.

3. Даны множества $A = \{3, 4, 5\}, \quad B = \{5, 6, 7, 8\},$ $C = \{2, 4, 8\}, \quad K = \{1, 3, 5, 7\}$. Найдите: *а*) $A \cup B \cap K$; *б*) $A \cap C \cup K$; *в*) $A \cup C \cap B$;
г) $A \cap K \cup B \cap C$.

Выполните упражнения (http://festival.1september.ru/articles/513260/):

Даны множества $A = \{3, 4, 5\}, \quad B = \{5, 6, 7, 8\},$ $C = \{2, 4, 8\}, \quad K = \{1, 3, 5, 7\}$. Найдите: а) $A \setminus K$; б) $C \setminus A$; в) $K \setminus B$; г) $A \setminus K \setminus B$.

Найдите $A \setminus B$ и $B \setminus A$ если $A = \{x | x - число\ меньшее\ 77\}$ и $B = \{x | x - число\ большее\ 12, но\ меньшее\ 93\}$.

Даны множества $A = \{3, 4, 5\}, \quad B = \{5, 6, 7, 8\},$ $C = \{2, 4, 8\}, \quad K = \{1, 3, 5, 7\}$. Найдите: а) $A \cup K \setminus B$; б) $A \cap C \cup B \setminus K$;
в) $A \cup B \setminus C \setminus K$;
г) $A \cap K \cup B \setminus C$

Пример решения задачи (http://festival.1september.ru/articles/513260/):

Задача. В классе 35 учеников. Из них 20 занимаются в математическом кружке, 11 – в биологическом, 10 ребят не посещают эти кружки. Сколько биологов увлекаются математикой?

Решение. Изобразим эти кружки (рис. 5). Большой круг будет изображать учащихся класса. В этот круг поместим два поменьше. Один обозначим буквой *М* и он будет изображать математиков класса. Другой круг

обозначим *Б* – биологи класса. Очевидно, в общей части кругов, обозначенной *МБ*, окажутся те самые биологи – математики, которые нас интересуют. Теперь посчитаем: Всего внутри большого круга 35 ребят, внутри двух меньших $35-10=25$ ребят. Внутри "математического" круга *М* находятся 20 ребят, значит, в той части "биологического" круга, которая расположена вне круга *М*, находятся $25-20=5$ биологов, не посещающих математический кружок. Остальные биологи, их $11-5=6$ человек, находятся в общей части кругов *МБ*. Таким образом, 6 биологов увлекаются математикой.

Ответ. 6 биологов увлекаются математикой.

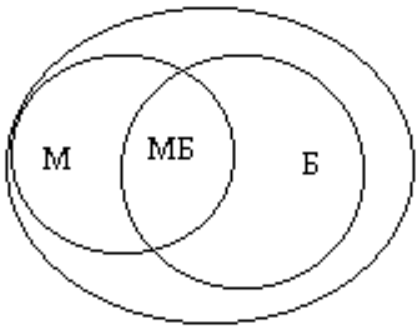


Рис. 5

Выполните упражнения (http://festival.1september.ru/articles/513260/):

1. В классе 29 учащихся. Каждый из них изучает хотя бы один язык – английский или немецкий. Английский язык изучают 18 человек, немецкий язык изучают 15 человек. Сколько человек изучают два языка и немецкий, и английский?

2. В классе 29 учащихся. Из них 16 занимаются музыкой, 21 посещают математический кружок; 4 не занимаются музыкой и не посещают математический кружок. Сколько учащихся посещают только математический кружок? Сколько математиков занимаются и музыкой?

3. В пионерском лагере 70 ребят. Из них 27 занимаются в драмкружке, 32 поют в хоре, 22 увлекаются спортом. В

драмкружке 10 ребят из хора, в хоре 6 спортсменов, в драмкружке 8 спортсменов; 3 спортсмена посещают и драмкружок и хор. Сколько ребят не поют, не увлекаются спортом, не занимаются в драмкружке? Сколько ребят заняты только спортом?

4. В классе 38 человек. Из них 16 человек играют в баскетбол, 17 человек – в хоккей, 18 человек – в волейбол. Увлекаются двумя видами спорта – баскетболом и хоккеем 4 человека, баскетболом и волейболом 3 человека, волейболом и хоккеем 5 человек. Трое не увлекаются ни баскетболом, ни волейболом, ни хоккеем. Сколько ребят увлекаются одновременно тремя видами спорта?

Вопросы к тематическому зачету по теме "Элементы теории множеств"

1. Кто является основоположником теории множеств?
2. Какие два множества являются равными?
3. Как называется множество, в котором нет ни одного элемента?
4. Составьте множество и запишите два элемента принадлежащие этому множеству и два элемента не принадлежащие ему.
5. Какое множество является подмножеством данного множества?
6. Для множества $A = \{8, 11, 20\}$ составьте все его подмножества.
7. Какое множество является пересечением двух множеств? Приведите пример.
8. Какое множество является объединением двух множеств? Приведите пример.
9. Как вычитаются множества? Приведите пример.

2.1.2 Математическая логика

Презентация: Практическое занятие - часть 1
Презентация: Практическое занятие - часть 2
Презентация: Практическое занятие - часть 3

2.1.3 Реляционная алгебра

Основы реляционной алгебры (http://habrahabr.ru/post/145381/)

Реляционная алгебра базируется на теории множеств и является основой логики работы баз данных.

Реляционной базой данных называется совокупность отношений, содержащих всю информацию, которая должна хранится в базе. В данном определении нам интересен термин отношение, но пока оставим его без строго определения.
Лучше представим себе таблицу продуктов.

таблица PRODUCTS

ID	NAME	COMPANY	PRICE
123	Печеньки	ООО ”Темная сторона”	190
156	Чай	ООО ”Темная сторона”	60
235	Ананасы	ОАО ”Фрукты”	100
623	Томаты	ООО ”Овощи”	130

Таблица состоит из 4х строк, строка в таблице является кортежем в реляционной теории. Множество упорядоченных кортежей называется отношением.
Перед тем как дать определение отношения, введем еще один термин — домен. Домены применительно к таблице это столбцы.

Для ясности, теперь введем строгое определение отношения.

Пусть даны N множеств D1,D2, …. Dn (домены), отношением R над этими множествами называется множество упорядоченных N-кортежей вида <d1,d1,...dn>, где d1 принадлежит D1 и тд. Множества D1,D2,..Dn называются доменами отношения R.
Каждый элемент кортежа представляет собой значение одного из атрибутов, соответствующего одному из доменов.

Ключи в отношениях
В отношении требованием является то, что все кортежи должны различаться. Для однозначной идентификации кортежа существует первичный ключ. Первичный ключ это атрибут или набор из

минимального числа атрибутов, который однозначно идентифицирует конкретный кортеж и не содержит дополнительных атрибутов. Подразумевается, что все атрибуты в первичном ключе должны быть необходимыми и достаточными для идентификации конкретного кортежа, и исключение любого из атрибутов в ключе сделает его недостаточным для идентификации.
Например, в такой таблице ключом будет сочетание атрибутов из первого и второго столбца.

таблица DRIVERS

COMPANY	DRIVER
ООО ”Темная сторона”	Владимир
ООО ”Темная сторона”	Михаил
ОАО ”Фрукты”	Руслан
ООО ”Овощи”	Владимир

Видно, что в организации может быть несколько водителей, и чтобы однозначно идентифицировать водителя необходимо и значение из столбца “Название организации” и из “Имя водителя”. Такой ключ называется составным.

В реляционной БД таблицы взаимосвязаны и соотносятся друг с другом как главные и подчиненные. Связь главной и подчиненнной таблицы осуществляется через первичный ключ (primary key) главной таблицы и внешний ключ (foreign key) подчиненной таблицы.
Внешний ключ это атрибут или набор атрибутов, который в главной таблице является первичным ключем.

Этой подготовительной теории будет достаточно для знакомства с основными операциями реляционной алгебры.

Операции реляционной алгебры

Основные восемь операций реляционной алгебры были предложены Э. Коддом.

- Объединение
- Пересечение

- Вычитание
- Декартово произведение
- Выборка
- Проекция
- Соединение
- Деление
  Первая половина операций аналогична таким же операциям над множествами. Часть операций можно выразить через другие операции. Рассмотрим большую часть операций с примерами.

  Для понимания важно запомнить, что результатом любой операции алгебры над отношениями является еще одно отношение, которое можно потом так же использовать в других операциях.
  Создадим еще одну таблицу, которая нам пригодится в примерах.

  таблица SELLERS

ID	SELLER
123	ООО “Дарт”
156	ОАО ”Ведро”
235	ЗАО “Овоще База”
623	ОАО ”Фирма”

Условимся, что в этой таблице ID это внешний ключ, связанный с первичным ключом таблицы PRODUCTS.

Для начала рассмотрим самую простую операцию — имя отношения. Её результатом будет такое же отношение, то есть выполнив операцию PRODUCTS, мы получим копию отношения PRODUCTS.

Проекция
Проекция является операцией, при которой из отношения выделяются атрибуты только из указанных доменов, то есть из таблицы выбираются только нужные столбцы, при этом, если получится несколько одинаковых кортежей, то в результирующем отношении остается только

по одному экземпляру подобного кортежа.
Для примера сделаем проекцию на таблице PRODUCTS выбрав из нее ID и PRICE.

Синтаксис операции:
$A_{(ID, PRICE)}$ PRODUCTS

В результате этой операции получим отношение:

ID	PRICE
123	190
156	60
235	100
623	130

Выборка
Выборка — это операция, которая выделяет множество строк в таблице, удовлетворяющих заданным условиям. Условием может быть любое логическое выражение.
Для примера сделаем выборку из таблицы с ценой больше 90.

Синтаксис операции:
$\Gamma_{(PRICE>90)}$ PRODUCTS

ID	NAME	COMPANY	PRICE
123	Печеньки	ООО ”Темная сторона”	190
235	Ананасы	ОАО ”Фрукты”	100
623	Томаты	ООО ”Овощи”	130

В условии выборки мы можем использовать любое логическое выражение. Сделаем еще одну выборку с ценой больше 90 и ID товара меньше 300:

$\Gamma_{(PRICE>90 \wedge ID<300)}$ PRODUCTS

ID	NAME	COMPANY	PRICE
123	Печеньки	ООО ”Темная сторона”	190

235 Ананасы ОАО ”Фрукты” 100

Совместим операторы проекции и выборки. Мы можем это сделать, потому что любой из операторов в результате возвращает отношение и в качестве аргументов использует также отношение.
Из таблицы с продуктами выберем все компании, продающие продуты дешевле 110.

$\Pi_{COMPANY}(PRICE<100)$ PRODUCTS

COMPANY
ООО ”Темная сторона”
ОАО ”Фрукты”

Умножение
Умножение или декартово произведение является операцией, производимой над двумя отношениями, в результате которой мы получаем отношение со всеми доменами из двух начальных отношений. Кортежи в этих доменах будут представлять из себя все возможные сочетания кортежей из начальных отношений. На примере будет понятнее.

Получим декартово произведения таблиц PRODUCTS и SELLERS. Синтаксис операции:

PRODUCTS Ч SELLERS
Можно заметить, что у двух этих таблиц есть одинаковый домен ID. В подобной ситуации домены с одинаковыми названиями получают префикс в виде названия соответствующего отношения, как показано ниже.
Для краткости перемножим не полные отношения, а выборки с условием ID<235

(цветом выделены одни и те же кортежи)

PRODUCTS.ID	NAME	COMPANY	PRICE	SELLERS.ID	SELLER
123	Печеньки	ООО ”Темная сторона”	190	123	ООО “Дарт”

156	Чай	ООО ”Темная сторона”	60	156	ОАО ”Ведро”
123	Печеньки	ООО ”Темная сторона”	190	156	ОАО ”Ведро”
156	Чай	ООО ”Темная сторона”	60	123	ООО “Дарт”

Для примера использования этой операции представим себе необходимость выбрать продавцов с ценами меньше 90. Без произведения необходимо было бы сначала получить ID продуктов из первой таблицы, потом по этим ID из второй таблицы получить нужные имена SELLER, а с использованием произведения будет такой запрос:

A(SELLER) Г(RODUCTS.ID=SELLERS.ID^PRICE<90) PRODUCTS Ч SELLERS

В результате этой операции получим отношение:

SELLER

ОАО ”Ведро”

Соединение и естественное соединение

Операция соединения обратна операции проекции и создает новое отношение из двух уже существующих. Новое отношение получается конкатенацией кортежей первого и второго отношений, при этом конкатенации подвергаются отношения, в которых совпадают значения заданных атрибутов. В частности, если соединить отношения PRODUCTS и SELLERS, этими атрибутами будут атрибуты доменов ID.

Также для понятности можно представить соеднинение как результат двух операций. Сначала берется произведение исходных таблиц, а потом из полученного отношения мы делаем выборку с условием равенства атрибутов из одинаковых доменов. В данном случае условием является равенство PRODUCTS.ID и SELLERS.ID.

Попробуем соединить отношения PRODUCTS и SELLERS и получим отношение.

PRODUCTS NAME COMPANY PRIC SELLERS. SELLER

.ID			E	ID	
123	Печенья	ООО ”Темная сторона”	190	123	ООО “Дарт”
156	Чай	ООО ”Темная сторона”	60	156	ОАО ”Ведро”
235	Ананасы	ОАО ”Фрукты”	100	235	ЗАО “Овоще База”
623	Томаты	ООО ”Овощи”	130	623	ОАО ”Фирма”

Натуральное соединение получает схожее отношение, но в случае, если у нас корректно настроена схема в базе (в данном случае первичный ключ таблицы PRODUCTS ID связан с внешним ключем таблицы SELLERS ID), то в результирующем отношении остается один домен ID.

Синтаксис операции:
PRODUCTS$_{И}$SELLERS;

Получится такое отношение:

PRODUCTS. ID	NAME	COMPANY	PRICE	SELLER
123	Печеньки	ООО ”Темная сторона”	190	ООО “Дарт”
156	Чай	ООО ”Темная сторона”	60	ОАО ”Ведро”
235	Ананасы	ОАО ”Фрукты”	100	ЗАО “Овоще База”
623	Томаты	ООО ”Овощи”	130	ОАО ”Фирма”

Пересечение и вычитание.
Результатом операции пересечения будет отношение, состоящее из кортежей, полностью входящих в состав обоих отношений.
Результатом вычитания будет отношение, состоящее из кортежей, которые являются кортежами первого отношения и не являются кортежами второго отношения.
Данные операции аналогичны таким же операциям над множествам, так что, я думаю, нет необходимости подробно их расписывать.

Источники информации

- Основы использования и проектирования баз данных — В. М. Илюшечкин
- курс лекций Introduction to Databases — Jennifer Widom, Stanford University

Основные операции реляционной алгебры (http://www.e-uni.ee/e-kursused/eucip/arendus_vk/252____.html)

Для того чтобы действительно понять поиск данных, полезно начать с понятий реляционной алгебры, исследуя ее потенциальные операции.

- **Объединение** двух таблиц (union). Это набор строк (совокупность, комплект), который принадлежит, по крайней мере, одному из двух табличных комплектов. Объединение можно представлять, как таблицу, которая собрана из различных столбцов двух в таблице.
- **Разница** между двумя таблицами (difference). Предполагая, что в двух таблицах имеются одинаковые столбцы, определим разницу как набор строк (совокупность, комплект), которые имеются в первой и отсутствуют во второй таблице.
- **Прямым произведением** (декартово (Descartes) произведение, Cartesian product) является набором строк (совокупность, комплект), который получается путем расположения каждой строки из первой таблицы по одной рядом со строками второй таблицы. Пример прямого произведения двух приведенных ниже таблиц приведен в следующей ниже второй таблице. Таким образом, имеем здесь дело со всевозможными комбинациями между строками двух таблиц.

Пользователи			Учебный предмет	
Код студента	**Имя**		**Предмет**	**Студент**
A01	Марек		История	A01
A02	Мария		История	A03
A03	Элла		География	A01
			География	A02

Прямое произведение: Пользователи и Учебные предметы			
Код студента	**Имя**	**Предмет**	**Студент**
A01	Марек	История	A01
A01	Марек	История	A03
A01	Марек	География	A01
A01	Марек	География	A02
A02	Мария	История	A01

A02	Мария	История	A03
A02	Мария	География	A01
A02	Мария	География	A02
A03	Элла	История	A01
A03	Элла	История	A03
A03	Элла	География	A01
A03	Элла	География	A02

- **Выборка из таблицы**. Это набор строк (совокупность, комплект) таблицы, удовлетворяющий сериям условий, которые видны из самого выбора. В приведенной выше таблице серым цветом выделены строки, которые удовлетворяют условию «Код студента» таблицы «Пользователей» соответствует «Студенту» таблицы «Учебные предметы». Эти строки скопированы в следующую таблицу.

Выборка из прямого произведения			
Код студента	**Имя**	**Предмет**	**Студент**
A01	Марек	История	A01
A01	Марек	География	A01
A02	Мария	География	A02
A03	Элла	История	A03

- **Проекция.** Проекция представляет собой подмножество

столбцов, получаемое из прямого произведения путем удаления многократного вхождения столбцов (или неопределенных атрибутов), таким образом, исключают многократно представленные столбцы и удаляют столбцы с нежелательной информацией. В приведенном выше примере столбец «Студент» представляет ту же информацию, что столбец «Код студента» и, в итоге, не присутствует в результате операции проекция.

Проекция		
Код студента	**Имя**	**Предмет**
A01	Марек	История
A01	Марек	География
A02	Мария	География
A03	Элла	История

- **Конъюнкция** или **соединение** (слияние, объединение - *join*). Операция соединения, по сути, прямое произведение с последующей выборкой. Поскольку выборка содержит только равные условия, поэтому название операции - «эквивалентная связь, объединение по эквивалентности - equi-join ». Результат эквивалентной связи - особое промежуточное соединение двух таблиц. Если этому следует проекция, то такая операция именуется «естественное соединение - natural join». Две таблицы, сопровождающие последний вышеприведенный пример есть примеры эквивалентной связи и естественного соединения.

- **Переименование.** Используют для переименования столбцов таблицы. Например, столбец «Код студента» можно переименовать в столбец «Код».

Переименование		
Код	**Имя**	**Предмет**
A01	Марек	История
A01	Марек	География
A02	Мария	География
A03	Элла	История

Операции над данными (http://library.fentu.ru/book/iu/31/_2__.html)

Операции обработки кортежей

Эти операции связаны с изменением состава кортежей в каком-либо отношении.

- ДОБАВИТЬ - необходимо задать имя отношения и ключ кортежа.
- УДАЛИТЬ - необходимо указать имя отношения, а также идентифицировать кортеж или группу кортежей, подлежащих удалению.
- ИЗМЕНИТЬ - выполняется для названного отношения и может корректировать как один, так и несколько кортежей.

Операции обработки отношений.

На входе каждой такой операции используется одно или несколько отношений, результатом выполнения операции всегда является новое отношение.

В реляционной алгебре определены следующие операций обработки отношений:

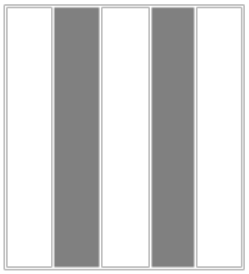
- ПРОЕКЦИЯ (ВЕРТИКАЛЬНОЕ ПОДМНОЖЕСТВО). Операция

проекции представляет из себя выборку из каждого кортежа отношения значений атрибутов, входящих в список A, и удаление из полученного отношения повторяющихся строк.

Проекция / PROJECT /

Обозначение	Определение	LEAP
$R[A]$	$\{r[A] : r \in R\}$	$r = \text{project}\ (R)\ (A_1, A_2, \ldots, A_m)$

Пример:



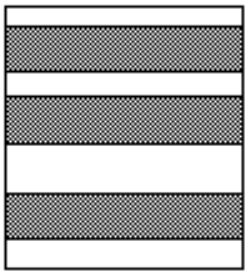
$$R[M,T] = \begin{bmatrix} x & a \\ y & a \\ z & a \\ w & b \\ w & b \\ w & b \end{bmatrix} = \begin{bmatrix} x & a \\ y & a \\ z & a \\ w & b \end{bmatrix}$$

- ВЫБОРКА (ОГРАНИЧЕНИЕ, ГОРИЗОНТАЛЬНОЕ ПОДМНОЖЕСТВО). На входе используется одно отношение, результат - новое отношение, построенное по той же схеме, содержащее подмножество кортежей исходного отношения, удовлетворяющих условию выборки.

Выборка / SELECT /

Обозначение	Определение	LEAP
$R[A\ \theta v]$ $R[A_1\ \theta A_2]$	$\{r : r \in R \wedge (r[A]\ \theta v)\}$ $\{r : r \in R \wedge (r[A_1]\ \theta r[A_2])\}$	r = select (R) ((cond) bool (cond))

Пример:



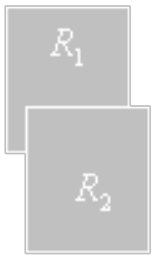
$$P[D_2 = 11] = \begin{bmatrix} 1 & 11 & x \\ 2 & 11 & y \\ 3 & 11 & z \end{bmatrix}$$

- ОБЪЕДИНЕНИЕ. Отношения-операнды в этом случае должны быть определены по одной схеме. Результирующее отношение содержит все строки операндов за исключением повторяющихся.

Объединение / UNION /

Обозначение	Определение	LEAP
$R_1 \cup R_2$	$\{r : r \in R_1 \vee r \in R_2\}$	r = (R1) union (R2)

Пример:



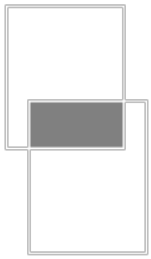
$$R[Q,T] \cup S = \begin{bmatrix} 5 & a \\ 3 & a \\ 9 & a \\ 1 & b \\ 2 & b \\ 4 & b \end{bmatrix} \cup \begin{bmatrix} 5 & a \\ 10 & b \\ 15 & c \\ 2 & d \\ 6 & a \\ 1 & b \end{bmatrix} = \begin{bmatrix} 5 & a \\ 3 & a \\ 9 & a \\ 1 & b \\ 2 & b \\ 4 & b \\ 10 & b \\ 15 & c \\ 2 & d \\ 6 & a \end{bmatrix}$$

- ПЕРЕСЕЧЕНИЕ. На входе операции два отношения, определенные по одной схеме. На выходе - отношение, содержащие кортежи, которые присутствуют в обоих исходных отношениях.

Пересечение / INTERSECT /

Обозначение	Определение	LEAP
$R_1 \cap R_2$	$\{r : r \in R_1 \wedge r \in R_2\}$	r = (R1) intersect (R2)

Пример:



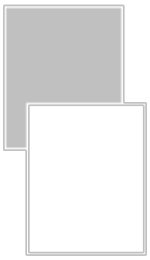
$$R[Q,T] \cap S = \begin{bmatrix} 5 & a \\ 3 & a \\ 9 & a \\ 1 & b \\ 2 & b \\ 4 & b \end{bmatrix} \cap \begin{bmatrix} 5 & a \\ 10 & b \\ 15 & c \\ 2 & d \\ 6 & a \\ 1 & b \end{bmatrix} = \begin{bmatrix} 5 & a \\ 1 & b \end{bmatrix}$$

- РАЗНОСТЬ. Операция во многом похожая на ПЕРЕСЕЧЕНИЕ, за исключением того, что в результирующем отношении содержатся кортежи, присутствующие в первом и отсутствующие во втором исходных отношениях.

Разность / SET DIFFERENCE /

Обозначение	Определение	LEAP
$R_1 - R_2$	$\{r : r \in R_1 \wedge r \notin R_2\}$	r = (R1) difference (R2) r = (R1) minus (R2)

Пример:



$$R[Q,T] - S = \begin{bmatrix} 5 & a \\ 3 & a \\ 9 & a \\ 1 & b \\ 2 & b \\ 4 & b \end{bmatrix} \cap \begin{bmatrix} 5 & a \\ 10 & b \\ 15 & c \\ 2 & d \\ 6 & a \\ 1 & b \end{bmatrix} = \begin{bmatrix} 3 & a \\ 9 & a \\ 2 & b \\ 4 & b \end{bmatrix}$$

- ДЕКАРТОВО ПРОИЗВЕДЕНИЕ Входные отношения могут быть

определены по разным схемам. Схема результирующего отношения включает все атрибуты исходных. Кроме того:

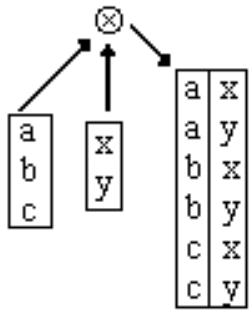
о степень результирующего отношения равна сумме степеней исходных отношений

о мощность результирующего отношения равна произведению мощностей исходных отношений.

Декартово произведение / CARTESIAN PRODUCT /

Обозначение	Определение	LEAP
$R_1 \otimes R_2$	$\{(r_1 \| r_2) : r_1 \in R_1 \wedge r_2 \in R_2\}$	r = (R1) product (R2)

Пример:



$$R[M,T] \otimes \left(R[Q,T] \cap S\right) = \begin{bmatrix} x & a \\ y & a \\ z & a \\ w & b \end{bmatrix} \otimes \begin{bmatrix} 5 & a \\ 1 & b \end{bmatrix} = \begin{bmatrix} x & a & 5 & a \\ x & a & 1 & b \\ y & a & 5 & a \\ y & a & 1 & b \\ z & a & 5 & a \\ z & a & 1 & b \\ w & b & 5 & a \\ w & b & 1 & b \end{bmatrix}$$

- СОЕДИНЕНИЕ Данная операция имеет сходство с ДЕКАРТОВЫМ ПРОИЗВЕДЕНИЕМ. Однако, здесь добавлено условие, согласно которому вместо полного произведения всех строк в результирующее отношение включаются только строки, удовлетворяющие опредленному соотношению между атрибутами соединения (A1,A2) соответствующих отношений.

Соединение / JOIN /

Обозначение	Определение
$R_1 \left[A_1 \; \theta A_2 \right] R_2$	$\left\{ \left(r_1 \| r_2 \right) : r_1 \in R_1 \wedge r_2 \in R_2 \wedge \left(r_1 \left[A_1 \right] \theta r_2 \left[A_2 \right] \right) \right\}$

LEAP : r = join (R1) (R2) ((cond) bool (cond))

Пример:

$$P\left[D_3 = D_4 \right] Q = \begin{bmatrix} 1 & 11 & x & x & 1 \\ 1 & 11 & x & x & 2 \\ 2 & 11 & y & y & 1 \\ 4 & 12 & x & x & 1 \\ 4 & 12 & x & x & 2 \end{bmatrix} = \begin{bmatrix} 1 & 11 & x & 1 \\ 1 & 11 & x & 2 \\ 2 & 11 & y & 1 \\ 4 & 12 & x & 1 \\ 4 & 12 & x & 2 \end{bmatrix}$$

- ДЕЛЕНИЕ Пусть отношение R , называемое делимым, содержит атрибуты (A1,A2,...,An). Отношение S - делитель содержит подмножество атрибутов A: (A1,A2,...,Ak) (k<n). Результирующее отношение C определено на атрибутах отношения R, которых нет в S, т.е. Ak+1,Ak+2,...,An. Кортежи включаются в результирующее отношение C только в том случае, если его декартово произведение с отношением S содержится в делимом R.

Деление / DIVISION /

Обозначение	Определение	LEAP :
$R_1\left[A_1 \div A_2\right]R_2$	$\left\{r\left[A_1\right] : r \in R_1 \wedge R_2\left[A_2\right] \subseteq g_R\left(r\left[A_1\right]\right)\right\}$	не поддерживается

Пример:

пусть

$$R_1\left(A_1, A_2\right) = \begin{bmatrix} a & x \\ a & y \\ a & z \\ b & x \\ c & y \end{bmatrix} \quad R_2\left(A_3, A_4\right) = \begin{bmatrix} 1 & x \\ 2 & x \\ 1 & y \end{bmatrix}$$

тогда

$$R_1\left[A_2 \div A_4\right] = \begin{bmatrix} a & x \\ a & y \\ a & z \\ b & x \\ c & y \end{bmatrix} \div \begin{bmatrix} x \\ y \end{bmatrix} = \left[a\right]$$

Реляционное исчисление

В реляционной модели определяются два базовых механизма манипулирования данными [5]:

- основанная на теории множеств реляционная алгебра
- основанное на математической логике реляционное исчисление.

Также как и выражения реляционной алгебры формулы реляционного исчисления определяются над отношениями реляционных баз данных, и результатом вычисления также является отношение. Эти механизмы манипулирования данными различаются уровнем процедурности:

- запрос, представленный на языке реляционной алгебры, может быть вычислен на основе вычисления элементарных алгебраических операций с учетом их старшинства и возможных скобок
- формула реляционного исчисления только устанавливает условия, которым должны удовлетворять кортежи результирующего отношения. Поэтому языки реляционного исчисления являются

более непроцедурными или декларативными.

Пример: Пусть даны два отношения:

СОТРУДНИКИ (СОТР_НОМЕР, СОТР_ИМЯ, СОТР_ЗАРПЛ, ОТД_НОМЕР) ОТДЕЛЫ(ОТД_НОМЕР, ОТД_КОЛ, ОТД_НАЧ)

Мы хотим узнать имена и номера сотрудников, являющихся начальниками отделов с количеством работников более 10. Выполнение этого запроса средствами реляционной алгебры распадается на четко определенную последовательность шагов:

(1).выполнить соединение отношений СОТРУДНИКИ и ОТДЕЛЫ по условию СОТР_НОМ = ОТДЕЛ_НАЧ.

С1 = СОТРУДНИКИ [СОТР_НОМ = ОТД_НАЧ] ОТДЕЛЫ

(2).из полученного отношения произвести выборку по условию ОТД_КОЛ > 10

С2 = С1 [ОТД_КОЛ > 10].

(3).спроецировать результаты предыдущей операции на атрибуты СОТР_ИМЯ, СОТР_НОМЕР

С3 = С2 [СОТР_ИМЯ, СОТР_НОМЕР]

Заметим, что порядок выполнения шагов может повлиять на эффективность выполнения запроса. Так, время выполнения приведенного выше запроса можно сократить, если поменять местами этапы (1) и (2). В этом случае сначала из отношения СОТРУДНИКИ будет сделана выборка всех кортежей со значением атрибута ОТДЕЛ_КОЛ > 10, а затем выполнено соединение результирующего отношения с отношением ОТДЕЛЫ. Машинное время экономится за счет того, что в операции соединения участвуют меньшие отношения. На языке реляционного исчисления данный запрос может быть записан как:

Выдать СОТР_ИМЯ и СОТР_НОМ для СОТРУДНИКИ таких, что

существует ОТДЕЛ с таким же, что и СОТР_НОМ значением ОТД_НАЧ

и значением ОТД_КОЛ большим 50.

Здесь мы указываем лишь характеристики результирующего отношения, но не говорим о способе его формирования. СУБД сама должна решить какие операции и в каком порядке надо выполнить над отношениями СОТРУДНИКИ и ОТДЕЛЫ. Задача оптимизации выполнения запроса в этом случае также ложится на СУБД. .

2.1.4 Проектирование БД. Организация связей между объектами БД

Цель работы:

1) получить практические навыки создания баз данных;

2) научится создавать таблицы базы данных;

3) приобрести практические навыки создания и изменения связей между таблицами;

4) освоить основные приемы заполнения и редактирования таблиц базы данных.

Задача. В туристической фирме требуется автоматизировать учет организуемых поездок и покупки путевок клиентами. Для этого необходимо создать базу данных, в которой будут содержаться все необходимые сведения о поездках, формируемых группах, клиентах, а также сведения о покупке путевок.

Для решения поставленной задачи требуется обеспечить хранение следующей информации в таблицах базы данных «Туристическая фирма»

? о странах: **КодСтраны,** название, столица, язык страны;

? о поездках: **КодПоездки**, *Страна*, маршрут, фото, количество дней поездки, тип поездки, вид транспорта, размещение, питание, стоимость;

? о группах: **№группы,** *код поездки*, дата отъезда, количество путевок;

? о клиентах: **№паспорта**, фамилия, имя, отчество, адрес, контактный телефон;

? о заявках: **№заявки**, **№группы**, **№паспорта**, количество путевок, скидка (нет скидки, 5 % или 10 %), необходимость визы.

Необходимо создать связи и обеспечить целостность данных

? между таблицами **Страны** и **Поездки** по полю КодСтраны и Страна, тип связи «один-ко-многим»;

? между таблицами **Поездки** и **Группы** по полю КодПоездки, тип связи «один-ко-многим»;

? между таблицами **Группы** и **Заявки** по полю №группы, тип связи «один-ко-

многим»;

? между таблицами **Клиенты** и **Заявки** по полю №паспорта, тип связи «один-ко-многим».

2.1.4.1 Проектирование таблиц базы данных

Задание 1. Создайте новую базу данных – **Туристическая фирма.**

Порядок выполнения:

1. Загрузите *MS Access*.
2. В области задач перейдите по ссылке **Новая база данных**.
3. В диалоговом окне *Файл новой базы данных* выберите свою папку в списке папок и щелкните в поле ввода **Имя файла**. Введите имя базы данных – **Туристическая фирма** – и нажмите кнопку **Создать**.

Примечание: После создания нового файла БД в поле окна *Access* появится окно базы *данных*. В окне базы данных каждый ярлык соответствует объекту базы данных. Можно работать как с базой данных целиком (создание, открытие и закрытие базы данных), так и с каждым ее объектом в отдельности. Для работы с любым объектом предусмотрено два режима: оперативный режим, в котором осуществляется просмотр и изменение информации (кнопка **Открыть**), и режим Конструктора, в котором изменяется макет, структура объекта (кнопка **Конструктор**). Набор пунктов горизонтального меню и состав панели инструментов зависят от активного окна, т. е. от того, какому объекту соответствует активное окно и в каком режиме происходит работа с этим объектом.

Задание 2. Создайте в режиме Конструктора таблицу базы данных **Страны**, структура которой приведена в табл. 10 (значок  в схеме таблицы означает, что поле является ключевым). Заполните таблицу данными.

Таблица 1

Структура таблицы Страны

Имя поля	Тип данных	Свойства поля		Описание
		Общие	Подста-новка	
КодСтраны	**Текстовый**	**Размер поля – 2**	–	**Уникальный код страны**
Название	**Текстовый**	**Размер поля – 100; Обязательное поле – *Да;* Пустые строки – *Нет***	–	**Название государства**
Столица	**Текстовый**	**Размер поля – 30**	–	**Столица государства**
Язык страны	**Текстовый**	**Размер поля – 30**	–	**Государственный язык страны**

Порядок выполнения:

1. В окне базы данных щелкните по ярлыку *Таблицы*.
2. Нажмите кнопку **Создать** в верхней части окна базы данных.
3. В окне *Новая таблица* выберите пункт **Конструктор.**

Примечание: Окно таблицы в режиме Конструктора состоит из 2-х областей: области *проекта таблицы* и области *свойств поля*. Переход между областями осуществляется клавишей **[F6]** или с помощью мыши. Область проекта содержит таблицу из 3-х столбцов: **Имя поля, Тип данных, Описание.** Переход между столбцами осуществляется одним из способов: мышкой, нажатием на клавишу **[Enter]** либо **[Tab],** клавишами управления курсором. Чтобы активизировать поле, достаточно щелкнуть мышью в строке, в которой описывается данное поле. В нижней области *свойств поля* располагаются 2 вкладки свойств: *Общие* и *Подстановка*. На вкладке *Общие* указаны отдельные характеристики каждого поля, необходимые для определения параметров сохранения данных в поле, их дальнейшего отображения и редактирования. Вкладка *Подстановка* содержит список некоторых дополнительных параметров, необходимых, в частности, для настройки связей с полями других таблиц.

4. В окне проекта таблицы создайте поле ***КодСтраны***. Для этого:

- введите в первую строку столбца **Имя поля** наименование поля ***КодСтраны*** и нажмите клавишу **[Enter];**
- во втором столбце **Тип данных** оставьте выводящееся по умолчанию значение **Текстовый** (любой другой выбирается из раскрывающегося списка типов данных);
- поле ***КодСтраны*** сделайте ключевым. Для этого активизируйте поле и нажмите кнопку **Ключевое поле** на панели инструментов **Конструктор таблиц** либо выполните команду **Правка→ Ключевое поле**;
- в колонке **Описание** добавьте текст в соответствии с табл. 10;
- переключитесь в область *Свойства поля* и на вкладке *Общие* измените размер поля на значение 2.

5. Введите имя поля ***Название*****,** перейдите в *Область свойств* и измените размер поля в соответствии с табл. 10, затем установите следующие значения *свойств поля*:

- обязательное поле – *Да***;**
- пустые строки – *Нет*.

Значения следует выбирать из раскрывающего списка в строке нужного свойства.

6. Аналогично добавьте в таблицу поля ***Столица***, ***ЯзыкСтраны*** (свойства полей установите по табл. 10).
7. Выполните команду **Файл→ Сохранить** либо нажмите одноименную кнопку на панели инструментов и в отрывшемся окне введите имя таблицы – **Страны**.
8. Перейдите в режим таблицы, выполнив команду **Вид→ Режим таблицы**.

Примечание: В левой части окна таблицы расположена область выделения. Треугольный маркер (>) в области выделения указывает на активную запись, а звездочка (*) – на пустую. Для обозначения записи, в которой осуществляется ввод, используется обозначение карандаша. Передвижение по таблице можно производить с помощью клавиш управления курсором, **[Tab]** и **[Enter]**, а также щелчком мыши. Можно пользоваться стандартными для *Windows* комбинациями клавиш для быстрого продвижения по таблице. Записи в таблице можно перемещать, копировать и удалять теми же способами, что и в электронных таблицах. Столбец можно выделить щелчком мыши по заголовку. Столбцы можно перемещать вправо и влево, пользуясь методом *drag and drop*.

9. Введите следующие 7 записей в таблицу:

КодСтраны	Название	Столица	Язык страны
01	Польша	Варшава	Польский
02	Великобритания	Лондон	Английский
03	Италия	Рим	Итальянский
04	США	ашингтон	Английский
05	Болгария	София	Болгарский
06	Франция	Париж	Французский
07	Чехия	Прага	Чешский

10. Закройте таблицу.

Задание 2.

1. С помощью *Конструктора* создайте таблицу **Клиенты** в соответствии с табл. 2.

Таблица 2

Структура таблицы Клиенты

Имя поля	Тип данных	Свойства поля	
		Общие	Подстановка
НомерПаспорта	**Текстовый**	**Размер поля – 30** **Подпись – Номер паспорта**	**–**
Имя	**Текстовый**	**Размер поля – 50** **Обязательное поле – *Да*** **Пустые строки – *Нет***	
Отчество	**Текстовый**	**Размер поля – 50**	
Фамилия	**Текстовый**	**Размер поля – 50** **Обязательное поле – *Да*** **Пустые строки – *Нет***	
Адрес	**Текстовый**	**Размер поля – 255**	**–**
КонтактныйТелефон	**Текстовый**	**Размер поля – 30** **Обязательное поле – *Да*** **Пустые строки – *Нет***	**–**

2. Перейдите в режим таблицы, нажав кнопку **Вид**  на панели инструментов *Конструктор таблиц*. Введите произвольные 8 записей.

Задание 3. С помощью *Конструктора* создайте таблицу **Поездки** в соответствии с табл. 3.

Таблица 3

Структура таблицы Поездки

Имя поля	Тип данных	Свойства поля	
		Общие	Подстановка
КодПоездки	**Числовой**	**Размер поля – *Целое***	-
Страна	**Текстовый**	**Размер поля – 2**	-
Маршрут	**Текстовый**	**Размер поля – 255** **Пустые строки – *Нет***	-
КоличествоДней	**Числовой**	**Размер поля – *Целое***	-
ТипПоездки	**Текстовый**	**Размер поля – 30**	**Список значений: экскурсия; отдых; учеба; работа; шопинг. Ограничиться списком –*Нет***
ВидТранспорта	**Текстовый**	**Размер поля – 30**	**Список значений: автобус; авиа; ж/д; самостоятельно. Ограничиться списком – *Да***
Размещение	**Текстовый**	**Размер поля – 30**	**Список значений: "**"; "***"; "****"; апартаменты, по выбору. Ограничиться списком – *Да***

Имя поля	Тип данных	Свойства поля	
		Общие	Подстановка
Питание	**Логический**	**Формат поля – *Да/Нет***	
Стоимость	**Денежный**	**Формат поля – *Евро***	

Методическме рекомендации:

1. Установите свойства полей ***КодПоездки, Страна*** **и** ***Маршрут*** *в* соответствии с табл. 3.
2. Для поля ***ТипПоездки*** используйте **Мастер** подстановок**:**

- в столбце **Тип данных** выберите из списка значение **Мастер подстановок**;
- на первом шаге Мастера в окне *Создание подстановки* установите переключатель **Будет введен фиксированный набор значений**;
- в следующем окне диалога оставьте без изменений число столбцов – 1 и введите в **Столбец1** значения, указанные в табл. 3. При этом получится столбец значений, представленный на рис. 9.

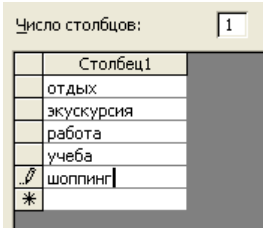


Рис. 9. Создание подстановки

Примечание: Переход к следующей строке столбца осуществляется с помощью клавиш **[Tab],** управления курсором либо щелчком мыши. Нажатием клавиши **[Enter],** так же, как и кнопки **Далее**, выполняется переход в следующее диалоговое окно.

- нажмите кнопку **Готово** по завершении процедуры создания столбца подстановок.

3. Измените свойства поля согласно табл. 3.
4. Аналогично вышеописанной технологии создайте поля со списком: ***Размещение*** и ***ВидТранспорта***. Списки значений этих полей указаны в табл. 3.
5. Активизируйте поле ***КоличествоДней***. Задайте размер поля – *Целое* – в области *свойств*.

6. Для поля ***Питание*** измените тип данных – *логический*, формат поля – **Да/Нет.**
7. Для поля ***Стоимость*** установите формат *Евро.*
8. Закройте таблицу **Поездки**, сохранив изменения.

Задание 4. В режиме Конструктора создайте таблицу **Группы,** структура которой приведена в табл. 4. Заполните таблицу данными.

Таблица 4

Структура таблицы Группы

Имя поля	Тип данных	Свойства поля	
		Общие	Подстановка
НомерГруппы	**Числовой**	**Размер поля – *Целое***	-
Маршрут	**Числовой**	**Размер поля – *Целое*** **Обязательное поле – *Да***	**Поле** КодПоездки **из таблицы** Поездки
ДатаОтъезда	**Дата/время**	**Краткий формат даты** **Маска ввода-** **00/00/0000;0;#**	-
КоличествоПутевок	**Числовой**	**Размер поля – *Целое*** **Условие на значение – >0**	**-**

Порядок выполнения:

1. Дважды щелкните по ярлыку **Создание таблицы в режиме конструктора** в окне базы данных
2. Создайте ключевое поле ***НомерГруппы*** по технологии, описанной в задании 1.
3. Создайте поле со списком ***Маршрут***, в качестве источника строк которого используются значения поля ***КодПоездки*** из таблицы **Поездки.** Для этого:

- введите имя поля, в столбце **Тип данных** выберите из списка значение **Мастер подстановок**;
- в окне *Создание подстановки* установите переключатель **Объект «столбец подстановки» будет использовать значения из таблицы или запроса**;
- в следующем диалоговом окне выберите из списка таблицу, из которой будет

осуществляться подстановка – **Поездки**;

- на следующем шаге Мастера из списка *Доступные поля* в список *Выбранные поля* с помощью кнопки

> переместите поля ***КодПоездки*** и ***Маршрут*** (рис. 10);

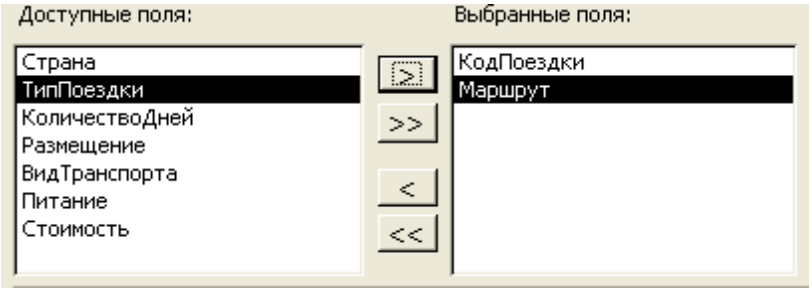


Рис. 10. Выбор полей при создании подстановки

- в следующем окне оставьте без изменения флажок **Скрыть ключевой столбец** и измените, если нужно, ширину столбца подстановки;
- на последнем шаге Мастера оставьте без изменения подпись, которую содержит столбец подстановок, и нажмите кнопку **Готово**;
- подтвердите создание связи в ответ на запрос *MS Access*.

4. Добавьте в таблицу поле ***ДатаОтъезда***, задайте для него тип данных – **Дата/время**. Установите для поля формат – *Краткий формат даты.*
5. Используя **Мастер масок ввода**, задайте для поля ***ДатаОтъезда*** маску ввода. Для этого*:*

- щелкните мышью в поле свойства **Маска ввода** и нажмите кнопку Построителя **[…],** которая находится справа поля;
- подтвердите сохранение таблицы, имя таблицы – **Группы**;
- в окне *Создание масок ввода* выберите значение *Краткий формат даты****;***
- на следующем шаге мастера выберите из списка символ заполнителя #. Для проверки ввода щелкните в поле **Проба** и введите произвольную дату;
- на последнем шаге мастера нажмите кнопку **Готово**.

Примечание: *Маска ввода* – это шаблон, позволяющий вводить в поле значения, имеющие одинаковый формат. Маска ввода автоматически изображает в поле постоянные символы, при вводе данных достаточно заполнить пустые позиции в маске ввода.

6. Добавьте в таблицу поле ***КоличествоПутевок*.** Задайте тип данных – *числовой*.
7. Перейдите в область *Свойств* и установите формат поля – *Целое,* в строке свойства **Условие на значение** введите выражение *>0*. В поле свойства **Сообщения об ошибке** введите текст *«Количество не может быть отрицательным числом»*.
8. Сохраните таблицу. Заполнять таблицу данными пока не следует.

Задание для самостоятельного выполнения. *В режиме Конструктора создайте таблицу* ***Заявки*** *согласно табл. 5.*

Указания к выполнению: *чтобы определить составной ключ, необходимо в области маркирования мышью выделить все три поля и любым из описанных ранее способов сделать их ключевыми.*

Таблица 5

Структура таблицы Заявки

Имя поля	Тип данных	Свойства поля	
		Общие	Подстановка
НомерЗаявки	**Счетчик**	**Индексированное поле – *Да* (совпадения не допускаются)**	-
НомерГруппы	**Числовой**	**Размер поля – *Целое*** **Индексированное поле – *Да* (совпадения допускаются)**	**Поле** НомерГруппы **из таблицы** Группы
НомерПаспорта	**Текстовый**	**Размер поля – 20** **Индексированное поле – *Да* (совпадения допускаются)**	**Поле** НомерПаспорта **из таблицы** Клиенты
КоличествоПутевок	**Числовой**	**Размер поля – *Байт***	
Скидка, %	**Числовой**	**Размер поля – *Байт***	
Необходимость визы	**Логический**	**Формат поля – *Вкл/Выкл***	**-**

2.1.4.2 Создание связей между таблицами

Задание 5. Установите связи между таблицами согласно поставленной задаче.

Порядок выполнения:

1. В окне базы данных выполните команду **Сервис→ Схема данных** либо нажмите кнопку **Схема данных** на панели инструментов.
2. При первом обращении к схеме данных на экран автоматически будет выведено окно *Добавление таблицы*. Если необходимо добавить таблицы в существующую схему, нажмите кнопку **Отобразить таблицу** на панели инструментов либо выполните команду **Связи→ Добавить таблицу**.
3. Выделите название таблицы **Страны** в списке и нажмите кнопку **Добавить**. Закройте окно *Добавление таблицы*.
4. В окне *Схема Данных* с помощью мыши перетяните поле ***КодСтраны*** из таблицы **Страны** на поле ***Страна*** таблицы **Поездки.**
5. В появившемся окне *Изменение связей* настройте параметры связи – установите флажки в области **Обеспечение целостности данных** (см. рис. 11) и нажмите кнопку **Создать**.

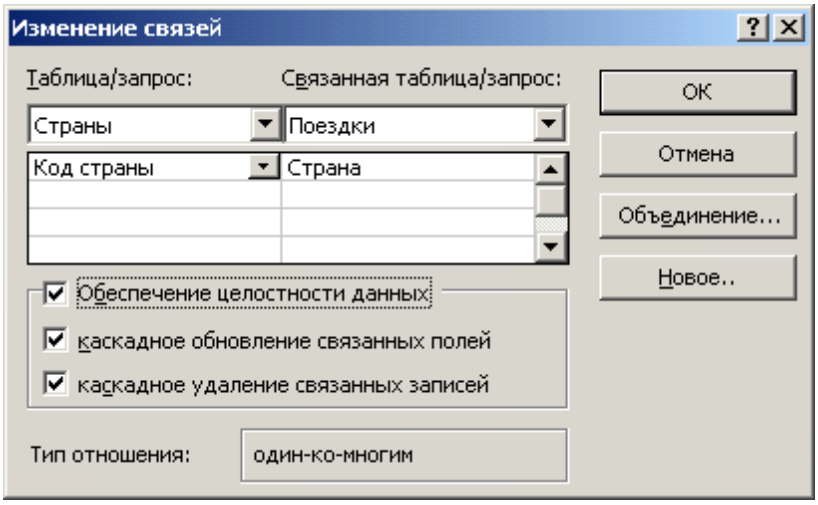


Рис. 11. Окно Изменение связей

Примечание: Включение флажка **Обеспечение целостности данных** позволяет защититься от случаев удаления записей из одной таблицы, при которых связанные с ними данные других полей окажутся без связи. Чтобы условие целостности могло

существовать, поле основной таблицы обязательно должно быть ключевым и оба поля должны иметь одинаковый тип. Флажки **Каскадное обновление связанных полей** и **Каскадное удаление связанных полей** обеспечивают одновременное обновление или удаление данных во всех подчиненных таблицах при их изменении в главной таблице.

6. Между таблицами **Поездки** и **Группы** связь уже была создана (в процессе настройки подстановочного поля), но для обеспечения целостности данных требуется ее дополнительная настройка. Измените связь между таблицами **Страны** и **Поездки**. Для этого:

- выделите с помощью мыши связь между этими таблицами;
- выполните команду **Связи→ Изменить связь**;
- в окне *Изменение связей* установите все нужные флажки и нажмите **ОК**.

Примечание: Обратите внимание, что концы линии связи в окне схемы данных после включения флажка **Обеспечения целостности данных** помечены знаками **"1"** и **"• "**. Это означает, что в качестве значений поля из связанной таблицы могут выступать только значения из соответствующего поля основной таблицы и каждое значение из поля основной таблицы может много раз встречаться в поле связанной таблицы (связь "один ко многим"). Поле со стороны **"• "** является внешним ключом к основной таблице.

7. Дважды щелкните по линии связи между таблицами **Заявки** и **Клиенты,** в открывшемся окне *Изменение связей* установите нужные флажки.
8. По вышеописанной технологии определите связи между таблицами **Заявки** и **Группы**. В результате должна получиться схема данных, представленная на рис. 12.
9. Закройте окно схемы данных.
10. Закройте базу данных, выполнив команду **Файл→ Закрыть.**

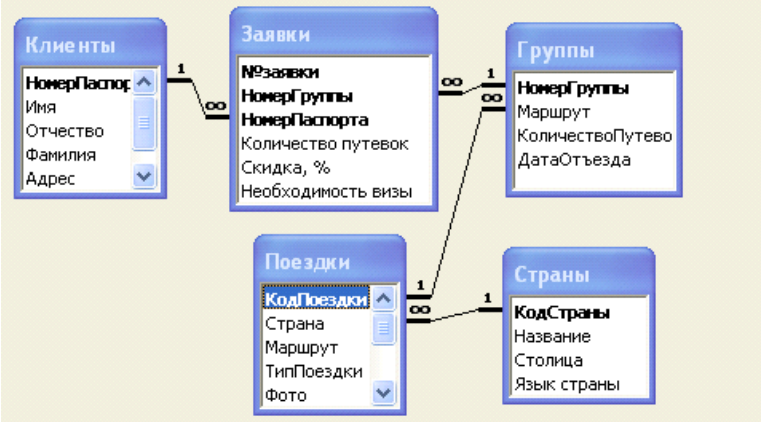


Рис. 12. Схема данных

2.1.5 Функциональные возможности СУБД

Презентация: Функциональные возможности СУБД

2.1.6 Среда СУБД MS Access. Проектирование таблиц и работа с ними

Задание 6. Заполните данными таблицу **Группы** данными. Отсортируйте таблицу в порядке возрастания по полю ***ДатаОтъезда***.

Порядок выполнения:

1. В окне базы данных выделите имя таблицы **Группы** и щелкните кнопку **Открыть.**
2. Введите 7 записей согласно табл. 15. Проверьте, как работает проверка вводимых значений в поле ***КоличествоДней*.**

Таблица 6

Группы

Номер группы	Маршрут*	Дата Отъезда**	Количество путевок
1	**Краков – Аквапарк – Соляные Копи – "Величка"**	**3 дня назад**	**25**
2	**Карловы Вары**	**Через 8 дней**	**30**
3	**Прага – Париж – Ницца – Берлин**	**Через 3 дня**	**25**
4	**Бостон**	**Через 30 дней**	**10**
5	**Краков – Аквапарк – Соляные Копи – "Величка"**	**Через 15 дней**	**25**

Номер группы	Маршрут*	Дата Отъезда**	Количество путевок
7	**Лондон – Йорк – Глазго – Амстердам – Берлин**	**Через 10 дней**	**20**
8	**Рим – Венеция – Флоренция**	**Через 20 дней**	**15**

Внимание*!* ******При вводе данных значения поля ***Маршрут*** следует выбирать из раскрывающегося списка, созданного подстановкой значений из таблицы **Поездки**.

*******В поле ***ДатаОтъезда*** указаны прошедшие либо будущие даты относительно текущей. Например, если сегодня 17.02.2005, то в первой записи в поле ***ДатаОтъезда*** следует ввести значение 14.02.2005.

3. Отсортируйте записи в таблице по полю *ДатаОтъезда* в порядке возрастания:

- активизируйте нужное поле;
- выполните команду **Записи → Сортировка→ Сортировка по возрастанию** или нажмите кнопку


на панели инструментов *Таблица в режиме таблицы.*

4. Закройте таблицу **Группы**, сохранив изменения.

Задание 7. Оформите таблицу **Поездки**. Выполните поиск записей по образцу и фильтрацию данных в таблице **Поездки**.

Порядок выполнения:

1. В окне базы данных выполните двойной щелчок на имени таблицы **Поездки**.
2. Выполните команду **Формат→ Шрифт** и установите для таблицы шрифт Arial, полужирный курсив, размер – 12 пт.
3. Выполните команду **Формат→ Режим таблицы** и установите значения параметров диалогового окна *Формат таблицы* по своему усмотрению.

Примечание: Для оформления таблицы можно использовать соответствующие кнопки на панели инструментов *Формат (режим таблицы).*

4. Измените ширину поля **Маршрут** таким образом, чтобы информация полностью помещалась в столбце. Для этого:

- выделите весь столбец, щелкнув по его названию;

- выполните команд **Формат→ Ширина столбца→ По ширине данных.**

5. Откройте таблицу **Поездки** в режиме таблицы.
6. Найдите в таблице все записи, в которых поле **ТипПоездки** имеет значение *Отдых*. Для этого:

- выполните команду **Правка→ Найти**
- в окне *Поиск и замена* введите в поле **Образец** значение, которое требуется найти; для поля **Поиск в** задайте значение *Поездки:таблица*, остальные параметры оставьте без изменения;
- нажмите кнопку **Найти далее**, чтобы перейти к искомому значению;
- после просмотра всех найденных записей закройте окно *Поиск и замена***.**

7. Аналогично найдите в таблице все записи, в которых поле ***ВидТранспорта*** имеет значение *а**. Символ * заменяет любое количество символов, т. е. должны быть найдены записи, в которых поле ***ВидТранспорта*** имеет значение *авиа* или *автобус*.
8. Используя фильтр по выделенному фрагменту, отберите все записи таблицы, которые содержат данные об экскурсионных поездках, т. е. в которых поле ***ТипПоедзки*** содержит значение *экскурсия.* Для этого:

- активизируйте в таблице ячейку со значением *экскурсия* и выполните последовательность команд **Записи→ Фильтр→ Фильтр по выделенному** либо нажмите одноименную кнопку на панели инструментов *Режим таблицы*;
- просмотрите результат фильтрации;
- чтобы отменить фильтр, нажмите кнопку **Удалить фильтр** на панели инструментов либо выберите одноименную команду в контекстном меню.

9. Измените фильтр, просмотрите записи обо всех поездках, целью которых является отдых, размещение – отель****, вид транспорта – авиа:

- нажмите кнопку **Изменить фильтр** на панели инструментов *Режим таблицы***.** *Access* запоминает последний применяемый фильтр, поэтому в поле ***ТипПоездки*** будет отображаться значение *Экскурсия*;
- в поле ***ТипПоездки*** раскройте список и выберите значение *отдых*, аналогично

установите значения для полей ***Размещение*** и ***ВидТранспорта*****;**

10. Для просмотра результатов фильтрации нажмите кнопку **Применение фильтра** либо выполните команду **Фильтр→ Применить фильтр.**
11. Отмените фильтр для таблицы.
12. Закройте таблицу **Поездки.**

Задание для самостоятельного выполнения. *Введите в таблицу* ***Заявки*** *записи таким образом, чтобы во все группы, за исключением группы №5, оформили заявки по несколько клиентов и несколько клиентов оформили заявки более 1 раза в разные группы. При вводе данных значения полей* ***НомерПаспорта*** *и* ***НомерГруппы*** *следует выбирать из списков, созданных подстановкой полей из таблиц* ***Клиенты и Группы.*** *Оформите таблицу по своему усмотрению.*

2.1.7 Формирование запросов в СУБД MS Access

Цель работы:

1) получить навыки конструирования запросов различного типа для выборки данных из таблиц и управления данными;
2) освоить технологию создания запросов с помощью Мастеров создания запросов**.**

Задача. Требуется получить определенные сведения по установленным критериям из одной или нескольких таблиц базы данных. Также нужно получить новые данные, вычисляемые по значениям, хранящимся в таблице. В ряде случаев требуется удалять или обновлять записи в имеющихся таблицах базы данных и создать новую таблицу на основе условий отбора.

Для решения поставленной задачи нужно сформировать запросы на выборку данных из таблиц и запросы действия для изменения данных

в таблицах.

Откройте из своей папки базу данных **Туристическая фирма**.

2.1.7.1 Задание условий отбора в запросе

Задание 1. Сформируйте запрос на выборку, выбирающий из таблицы **Страны** записи о странах, основной язык в которых – английский.

Порядок выполнения:

1. В окне базы данных щелкните по ярлыку *Запросы*.
2. Дважды щелкните по ярлыку **Создание запроса в режиме конструктора** в окне базы данных. Откроется бланк запроса и окно *Добавление таблицы*.

Примечание: Для добавления таблицы в бланк запроса можно выделить нужную таблицу и нажать кнопку **Добавить** или дважды щелкнуть мышью по таблице. Если нужно добавить несколько таблиц, эту операцию следует повторить для каждой таблицы. Для удаления таблицы из бланка запроса нужно выделить таблицу и нажать кнопку **Delete**. В верхней области окна конструктора запросов отображаются выбранные таблицы и связи между ними. Нижняя область – бланк запроса – отображает поля и условия вывода для данных из таблиц.

3. Добавьте в запрос таблицу **Страны** и закройте окно *Добавление таблицы*.
4. Перетащите поле ***Название*** из таблицы в строку **Поле** первого столбца бланка запроса.
5. В области таблиц дважды щелкните по полю ***Столица.*** Поле переместится в следующий свободный столбец бланка запроса.
6. В третьем столбце бланка запроса раскройте список в строке **Поле** и выберите поле ***ЯзыкСтраны***.
7. Для задания условий отбора на пересечении строки **Условие отбора** и столбца ***ЯзыкСтраны*** наберите *английский*.
8. Запустите запрос на выполнение с помощью кнопки **Запуск (!)** на панели инструментов.
9. Сохраните запрос под именем **Страны с английским языком**.

Задание 2. Сформируйте запрос на выборку для отбора записей из таблицы **Страны** о странах, основной язык в которых французский или итальянский.

Порядок выполнения:

1. В окне базы данных нажмите кнопку **Создать** и в окне диалога *Новый запрос*

выберите пункт **Конструктор**.

2. Добавьте таблицу **Страны** в бланк запроса**.**
3. Включите поля ***Название***, ***Столица***, ***ЯзыкСтраны*** в бланк запроса**.**
4. Для задания условий отбора на пересечении строки **Условие отбора** и столбца ***ЯзыкСтраны*** наберите *французский*. Для задания второго условия отбора на пересечении строки **или** и столбца ***ЯзыкСтраны*** наберите *итальянский.*
5. Запустите запрос на выполнение, выполнив команду **Запрос→ Запуск**.
6. Сохраните запрос под именем **Страны с французским и итальянским языком**.

Задание 3. Создайте запрос на основе таблицы **Поездки,** который выбирает все поля таблицы и выводит все записи. Добавьте в запрос две новых записи.

Порядок выполнения:

1. Создайте новый запрос в режиме Конструктора.
2. Чтобы включить в запрос все поля таблицы **Поездки**, переместите символ * из списка полей таблицы в первый столбец бланка запроса.
3. Сохраните запрос под именем **Новые поездки** и запустите на выполнение.
4. Нажмите кнопку **Новая запись** на панели инструментов **Запрос (режим таблицы)** либо выполните команду **Записи→ Ввод данных**. Введите следующую запись:

Код поездки	Страна	Маршрут	Тип поездки	Количество дней	Размещение	Вид транспорта	Питание	Стоимость
11	**02**	**Оксфорд**	**Учеба**	**90**	**самостоятельно**	**авиа**	**нет**	**2768**

5. Закройте окно запроса и в окне базы данных щелкните по ярлыку *Таблицы*.
6. Откройте таблицу **Поездки** и просмотрите записи.
7. Закройте таблицу и вернитесь к работе с запросами.

Задание 4. Сформируйте запрос на выборку записей из связанных таблиц **Клиенты, Заявки, Группы, Поездки** и **Страны**, выбирающий записи о клиентах, которым нужна виза, с указанием страны и срока оформления.

Порядок выполнения:

1. Создайте новый запрос в режиме Конструктора.
2. Добавьте таблицы **Клиенты**, **Заявки**, **Группы**, **Поездки** и **Страны** в бланк запроса**.**
3. Из таблицы **Клиенты** в бланк запроса включите поля**:** ***НомерПаспорта, Фамилия*****,** ***Имя*** и ***Отчество***, из таблицы **Страны** – поле ***Название*****,** из таблицы **Группа** включите поле ***ДатаОтъезда*****,** а из таблицы **Заявки – *НеобходимостьВизы***.
4. Для задания условий отбора в ячейке **Условие отбора** для поля ***НеобходимостьВизы*** наберите *Да*.
5. Для того чтобы столбец ***НеобходимостьВизы*** не отображался на экране, в строке **Вывод на экран** данного поля снимите флажок.
6. Запустите запрос на выполнение.
7. Сохраните запрос под именем **Клиенты, которым нужна виза.**

Задание для самостоятельного выполнения. *Сформируйте запрос на основе таблицы* ***Поездки****, выбирающий записи о поездках стоимостью < 300 Евро, продолжительностью > 10 дней и типом поездки – отдых. Сохраните запрос под именем* ***Недорогой отдых.***

2.1.7.2 Создание параметрических запросов

Задание 5. Сформируйте запрос на выборку записей из таблицы **Поездки**, запрашивающий тип поездки и выдающий информацию о маршруте, количестве дней, транспорте, размещении, питании и стоимости.

Порядок выполнения:

1. Создайте новый запрос в режиме Конструктора.
2. Добавьте таблицу **Поездки** в бланк запроса.
3. В бланк запроса включите поля: ***Маршрут*****,** ***КоличествоДней*****,** ***ВидТранспорта*****,** ***Размещение*****,** ***Питание*****,** ***Стоимость*** и ***ТипПоездки*****.**
4. В строке **Условие отбора** для поля ***ТипПоездки*** введите текст [Введите тип поездки].

Внимание! В параметрических запросах обращение должно быть задано в квадратных скобках!

5. Запустите запрос на выполнение.

6. В появившемся диалоговом окне задайте одно из значений типа поездки (например, *отдых*).
7. Просмотрите записи.
8. Сохраните запрос под именем **Поездки по типам.**

Задания для самостоятельного выполнения. *Сформируйте параметрический запрос на основе связанных таблиц* ***Страны*** *и* ***Поездки****, запрашивающий название страны и выводящий полную информацию о поездках, организованных в эту страну. Сохраните запрос под именем* ***Поездки по странам*****.**

2.1.7.3 Групповые вычисления в запросе

Задание 6. Сформируйте запрос, который позволяет определить, сколько раз каждый клиент воспользовался услугами фирмы.

Порядок выполнения:

1. Нажмите кнопку **Создать** и в окне *Новый запрос* выберите способ создания **Конструктор**.
2. Добавьте таблицы **Клиенты** и **Заявки** в бланк запроса.
3. Из таблицы **Клиенты** в бланк запроса включите поле ***Фамилия, Имя, Отчество***, а из таблицы **Заявки** – поле ***НомерЗаявки***.
4. Выполните команду **Вид→ Групповые операции.**
5. Щелкните мышью в строке **Групповая операция** поля ***НомерЗаявки***. В раскрывающемся списке выберите функцию ***Count*** (статистическая функция ***Count*** вычисляет количество значений).
6. Сохраните запрос под именем **Постоянные клиенты.**
7. Запустите запрос на выполнение.
8. Просмотрите записи. Обратите внимание, что появилось поле ***Count_ НомерЗаявки*****,** в котором подсчитывается количество заявок каждого клиента.

Задание 7. Сформируйте запрос, который позволяет посчитать количество купленных путевок в каждую группу.

Порядок выполнения:

1. Создайте новый запрос в режиме Конструктора.
2. Добавьте таблицы **Группа** и **Заявки** в бланк запроса**.**

3. Из таблицы **Группа** в бланк запроса включите поле ***НомерГруппы***, из таблицы **Заявки** – поле ***КоличествоПутевок***.
4. Нажмите кнопку **Групповые операции** (математический знак суммы) на панели инструментов **Конструктор запросов**.
5. Щелкните мышью в строке **Групповая операция** в поле ***КоличествоПутевок***. В раскрывающемся списке выберите функцию ***Sum*** (статистическая функция ***Sum*** суммирует значения определенного поля).
6. Чтобы изменить подпись поля ***Sum_*КоличествоПутевок,** установите курсор перед именем поля и наберите **Куплено путевок:** (имя поля должно быть следующим – **Куплено путевок: КоличествоПутевок**).
7. Сохраните запрос под именем **Количество купленных путевок.**
8. Запустите запрос на выполнение.
9. Просмотрите записи. Обратите внимание: поле, в котором подсчитывается количество купленных путевок называется **Куплено путевок.**

2.1.7.4 Запрос на удаление

Задание 11. Сформируйте запрос на удаление записей из таблицы **Группы.** Необходимо удалить приехавшие группы (дата возвращения раньше текущей даты).

Порядок выполнения:

1. Создайте новый запрос в режиме Конструктора.
2. Добавьте таблицу **Группы** и запрос **Дата возвращения групп** в бланк запроса.
3. Из таблицы **Группы** в бланк запроса включите все поля**.** Из запроса **Дата возвращения групп** в бланк запроса включите поле ***ДатаВозвращения***.
4. Для задания условий отбора на дату возвращения в ячейке ***Условие отбора*** для этого поля наберите *<Date()* (можно использовать построитель выражений).
5. Щелкните мышь по включенному в запрос полю **Группа**.
6. Выполните команду **Запрос→ Удаление**.
7. Сохраните запрос под именем **Приехавшие группы.**
8. Запустите запрос на выполнение.

9. В появившемся диалоговом окне подтвердите удаление записей.

2.1.8 Проектирование форм в СУБД MS Access

Цель работы:

1) освоить основные приемы создания форм для отображения, редактирования и управления данными реляционных таблиц;
2) изучить основные элементы форм и получить практические навыки их создания;
3) научиться модифицировать записи в режиме формы.

Задача. Необходимо создать формы для представления данных, позволяющие просматривать и редактировать данные из нескольких связанных таблиц базы данных, предоставить пользователю простейшие элементы для управления данными.

Откройте базу данных **Туристическая фирма.**

2.1.8.1 Создание формы с помощью Автоформы

Задание 1. Создайте форму на основе таблицы **Клиенты.**

Порядок выполнения:

1. В окне базы данных щелкните по ярлыку *Формы.*
2. Нажмите кнопку **Создать** в верхней части окна базы данных.
3. В окне *Создание форм* выберите из раскрывающегося списка в качестве источника данных таблицу **Клиенты.**
4. В списке способов создания форм выберите элемент **Автоформа: Ленточная**.
5. В результате откроется готовая форма в *режиме формы*. Просмотрите записи в форме, используя кнопки перехода, расположенные на нижней границе окна формы.

Примечание: Кнопки ◂ и ▸ позволяют перейти на предыдущую и следующую запись, кнопки ¦◂ и ▸|– на первую и последнюю, при нажатии кнопки ▸* происходит переход к новой, пустой записи. Те же результаты можно получить, выполнив команду **Правка→ Перейти** и выбрав одну из команд подменю. Для быстрого перехода к новой записи можно воспользоваться командой **Записи→ Ввод данных**. Заполнив запись, перейти к новой записи можно, нажав клавишу **[Enter].** Переход между полями в форме осуществляется с помощью клавиш **[Tab]** или управления курсором. Активная запись

маркируется символом ▸, пустая – *.

6. Перейдите к новой записи любым из описанных выше способов и введите новую запись.
7. Закройте форму, сохранив изменения.
8. В окне базы данных щелкните по ярлыку **Таблицы**.
9. Откройте таблицу **Клиенты** и просмотрите записи.
10. Вернитесь к работе с формами.

2.1.8.2 Создание формы с помощью Мастера

Задание 2. Создайте с помощью мастера форму на основе таблицы **Поездки.**

Порядок выполнения:

1. В окне базы данных щелкните по ярлыку *Формы* и нажмите кнопку **Создать.**
2. Выберите в качестве источника данных таблицу **Поездки**.
3. Укажите способ создания формы – *Мастер форм*.

Примечание: *Microsoft Access* по умолчанию использует выбранную таблицу или запрос как базовый источник данных для формы. Однако мастер позволяет изменить источник данных, а также выбрать поля из других таблиц или запросов.

4. В диалоговом окне *Создание форм* с помощью кнопки

переместите из списка **Доступные поля** в список **Выбранные поля** все поля таблицы **Поездки,** кроме поля ***Страна.***

5. Далее выберите внешний вид формы *в один столбец* и любой из предлагаемых стилей.
6. На последнем шаге задайте имя формы – **Поездки,** установите переключатель **Открыть форму для просмотра и ввода данных** и нажмите **Готово**.
7. В результате откроется созданная форма в режиме формы. Просмотрите записи в форме и закройте ее.

Задание для самостоятельного выполнения. *Создайте с помощью мастера форму на основе таблицы* ***Заявки.*** *Добавьте в форму поля* ***НомерЗаявки, НомерПаспорта, КоличествоПутевок*** *и* ***Необходимость визы****. Внешний вид – в один столбец****,*** *оформление – произвольно****.***

2.1.8.3 Создание формы в режиме конструктора

Задание 3. Создайте в режиме конструктора сложную форму на основе таблицы **Группы.** Форма должна содержать подчиненную форму **Заявки.**

Порядок выполнения:

1. В окне базы данных нажмите кнопку **Создать**.
2. В диалоговом окне *Новая форма* укажите способ создания – Конструктор и источник данных – таблица **Группы**. Откроется новая форма в режиме конструктора.
3. Выполните команду **Вид→ Заголовок/примечание формы.** В окне конструктора появятся области *Заголовка и примечания формы*.
4. Вставьте надпись в область **Заголовка**. Для этого:

- нажмите кнопку **Надпись** на панели элементов (если панель элементов отсутствует на экране, нажмите кнопку **Панель элементов** на панели инструментов **Конструктор** либо выполните команду **Вид→ Панель элементов**) и щелкните в области заголовка;
- наберите заголовок формы **«Оформление заявки»** и нажмите [**Enter].** Надпись будет выделена;
- с помощью кнопок на панели инструментов **Формат (форма/отчет)** установите для подписи шрифт **Courier New**, размер – **20**, полужирный курсив;
- выполните команду **Формат→ Размер→ По размеру данных**.

5. Переместите поле ***НомерГруппы*** из списка полей в *Область данных* формы (если списка полей нет на экране, выполните команду меню **Вид→ Список полей**). Для этого выделите нужное поле в списке полей, нажмите левую кнопку мыши и перетяните в область данных формы.
6. Измените размер созданного поля и его подписи таким образом, чтобы их содержимое отображалось полностью. Для этого подведите курсор мыши к любому (кроме верхнего левого угла) из квадратиков в углу выделенного объекта (курсор мыши примет вид двунаправленной стрелки) и расширьте рамку до нужного размера (рис. 14).

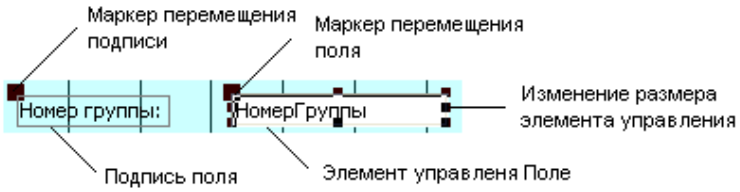


Рис. 14. Выделенный элемент управления

Примечание: Чтобы переместить элемент управления вместе с подписью, поместите указатель мыши в любую точку на границе выделенного элемента, отличную от маркеров изменения размера. Указатель мыши изменит свой вид на изображение руки. Перетяните указатель мыши в нужное место. Чтобы переместить элемент управления отдельно от связанной с ним подписи либо отдельно переместить подпись, следует перетаскивать за маркер, находящийся левом верхнем углу перемещаемого объекта (маркер перемещения).

7. Перетяните в область данных поле со списком **Маршрут.** Откорректируйте размеры поля и его подписи.
8. В списке полей выделите поле ***ДатаОтъезда*** и, удерживания нажатой клавишу **[Shift]**, поле ***КоличествоПутевок***.
9. Переместите выделенные поля в область данных. Откорректируйте размеры и расположение полей.
10. Выделите подпись поля ***ДатаОтъезда*** и нажмите кнопку **Свойства** на панели инструментов **Конструктор форм**.
11. В открывшемся окне свойств надписи на вкладке *Макет* измените свойство *Подпись* на Дата отъезда.
12. Не закрывая окна свойств, выделите подпись поля ***КоличествоПутевок*** и измените подпись для этого поля на Количество.
13. С помощью мыши выделите добавленную группу полей и выполните команду **Формат→ Выровнять→ По узлам сетки**.
14. Сохраните форму с именем **Оформление заявки.**
15. Добавьте в форму подчиненную форму **Заявки.** Для этого:

- нажмите кнопку **Подчиненная форма/отчет** на панели элементов и щелкните в

области данных;

- на первом шаге Мастера активизируйте переключатель **Имеющиеся формы** и выберите из списка форму **Заявки**;
- далее оставьте выбранную по умолчанию связь между формами и именем подчиненной формы;
- нажмите **Готово** для завершения работы с Мастером.

16. Создайте в подчиненной форме **Заявки** группу переключателей для ввода значения **Скидки**. Для этого:

- выберите на панели элементов элемент управления **Группа переключателей** и перетащите его в область данных (на панели инструментов должна быть нажата кнопка **Мастера**);
- в первом диалоговом окне Мастера создания группы нужно ввести текст надписей, которые будут размещаться справа от переключателей: **нет скидки, 5 %, 10 %;**
- на следующем шаге Мастера чтобы задайте значение, используемое по умолчанию: установите переключатель **Да, выбор по умолчанию** и выберите из списка – **Нет скидки**;
- далее определите соответствующие значения для каждого параметра: **0, 5, 10** (рис.15);

	Подписи:	Значения:
▶	нет скидки	0
	5 %	5
	10 %	10

Рис. 15. Задание значений параметров

- в следующем диалоговом окне установите переключатель **Сохранить значение в поле** и выберите из списка поле **Скидка, %**;
- далее выберите тип и вариант оформления группы переключателей на свое усмотрение, задайте подпись **Скидка** и нажмите **Готово**.

17. Увеличьте размер области **Примечания формы** подчиненной формы **Заявки**. Для этого перетащите с помощью мыши нижнюю границу области вниз.

18. Добавьте в область **Примечания формы** подчиненной формы поле, в котором вычисляется количество купленных путевок в группе.

Для этого:

- выберите на панели элементов элемент **Поле** и щелкните мышью в области **Примечания**. После размещения подпись поля будет содержать его номер, а само поле – ссылку Свободный;
- дважды щелкните по новому полю. В открывшемся окне *свойств* поля перейдите на вкладку *Данные*;
- в строке **Данные** нажмите кнопку построителя выражений **[…];**
- в окне *Построителя,* пользуясь папкой **Функции \ Встроенные функции** (категория *Статистические*) и списком полей формы, создайте следующее выражение: **=Sum ([КоличествоПутевок]);**
- для того чтобы в режиме формы нельзя было изменить значение поля**,** установите для свойства **Доступ** значение **Нет**, а для свойства **Блокировка** – **Да**;
- не закрывая окна свойств поля, щелкните мышью по подписи поля. В появившемся окне *свойств* **Надписи** на вкладке *Макет* измените свойство *Подпись* на **Количество купленных путевок;**
- закройте окно свойств.

19. Измените размеры подчиненной формы таким образом, чтобы все поля отображались на экране.

20. Добавьте в создаваемую форму подчиненную на основе запроса **Количество свободных мест в группе.** Для этого:

- нажмите кнопку **Подчиненная форма/отчет** на панели элементов и щелкните в области данных;
- на первом шаге Мастера оставьте активным переключатель **Имеющиеся таблицы и запросы;**
- в следующем диалоговом окне выберите из списка **Таблицы и запросы** источник данных – запрос **Количество свободных мест**

  и переместите из списка **Доступные поля** в список **Выбранные поля** следующие поля запроса: ***НомерГруппы*** и ***Количество свободных мест***;
- далее оставьте выбранную по умолчанию связь между формами и задайте имя для формы – **Наличие путевок**.

21. Удалите из подчиненной формы **Наличие путевок** поле ***НомерГруппы*** (в процессе создания это поле было необходимо для установки связи). Для этого выделите в подчиненной форме нужное поле и нажмите кнопку **[Delete].**

22. Запретите доступ для подчиненной формы **Наличие путевок** и отключите полосы прокрутки и кнопки перехода по записям. Для этого:

- с помощью мыши выделите подчиненную форму и нажмите кнопку **Свойства**;
- в окне Подчиненная форма / отчет на вкладке *Данные* установите значение **Нет** для свойства *Доступ* и значение **Да** для свойства *Блокировка*;
- на вкладке *Макет* для свойства *Полосы прокрутки* установите значение **Отсутствуют** и значение **Нет** для свойства *Кнопки перехода.*

Примечание: При проектировании формы в учебных целях допущена некоторая избыточность данных. В общем случае достаточно либо вычисляемого поля ***Куплено путевок,*** либо подчиненной формы **Наличие путевок**.

23. Создайте в главной форме **Оформление заявки** кнопку для открытия формы **Клиенты**, чтобы добавить нового клиента в одноименную таблицу, не возвращаясь в окно базы данных. Для этого:

- выберите элемент **Кнопка** на панели элементов и щелкните в области данных;
- в первом диалоговом окне Мастера создания кнопок в списке категорий выберите категорию **Работа с формой**, в списке действий – пункт **Открыть форму;**
- на следующем шаге укажите в списке открываемую форму – **Клиенты;**
- далее оставьте без изменения установленный по умолчанию переключатель **Открыть форму и показать все записи;**
- разместите на кнопке текст **Новый клиент** и завершите работу с Мастером.

24. Оформите элементы форм. Для этого:

- выделите с помощью мыши нужный элемент;
- пользуясь кнопками на панели инструментов **Формат (форма/отчет)**, задайте на свое усмотрение шрифт, размер и цвет текста, цвет заливки/фона, утопленное или приподнятое оформление.

25. Оформите форму, используя элементы панели инструментов **Прямоугольник** и **Линия**.

26. Задайте в качестве фона главной формы рисунок. Для этого:

- выделите всю форму, выполнив команду **Правка→ Выделить форму**;
- нажмите кнопку **Свойства** на панели инструментов **Конструктор форм.** Появится окно *свойств* формы;
- перейдите на вкладку *Макет* и в строке **Рисунок** нажмите кнопку *Построителя*;
- в диалоговом окне *Выбор рисунка* укажите файл *Природа_фон.jpg* (путь к файлу уточните у преподавателя);
- для свойства *Тип рисунка* установите **Внедренный**, для свойства *Масштабы* рисунка – **Вписать в рамку.**

27. Перейдите в режим формы, нажав кнопку **Вид** на панели инструментов.

28. Отредактируйте, если нужно, размеры и расположение элементов формы, вернувшись в режим конструктора.

29. Используя кнопки перехода по записям, просмотрите записи в форме. Закройте форму, сохранив изменения.

2.1.9 Проектирование отчетов в СУБД MS Access

Цель работы:

1) освоить основные приемы создания отчетов для вывода данных реляционных таблиц;

2) изучить основные элементы отчетов и получить практические навыки их создания.

Задача. Необходимо сформировать отчеты, позволяющие представить информацию в удобном для пользователя виде, сгруппировать данные и подвести итоги.

Откройте базу данных **Туристическая фирма.**

2.1.9.1 Создание отчета с помощью Автоотчета

Задание 1. Создайте отчет на основе таблицы **Клиенты.**

Порядок выполнения:

1. В окне базы данных щелкните по ярлыку *Отчеты*.

2. Нажмите кнопку **Создать**.

3. В окне *Новый отчет* выберите из раскрывающегося списка в качестве источника данных таблицу **Клиенты.**

4. В списке способов создания форм выберите элемент **Автоотчет:Ленточный**.

5. В результате откроется готовый отчет в *режиме просмотра*.
6. Просмотрите отчет.
7. Сохраните отчет под именем **Клиенты**.

2.1.9.2 Редактирование отчета в режиме конструктора

1. В окне базы данных выделите отчет **Поездки** и нажмите кнопку **Конструктор**.
2. Измените с помощью мыши размеры надписи в области заголовка отчета.
3. В области *Верхнего колонтитула* откорректируйте подписи полей по аналогии с формами.
4. Удалите поле ***Страна*** в области *данных*. Подпись поля в области *Заголовка группы* удалять не следует.
5. Создайте элемент управления **Поле со списком**, который будет использовать значения поля ***Страна*** таблицы **Страны**. Технология создания элементов управления в отчетах аналогична этой технологии в формах, описанной в предыдущей лабораторной работе. Подпись поля удалите.
6. В области *Примечания группы «Страна»* измените подпись поля *Avg* на «Средняя стоимость путевки».
7. Выполните команду **Вид→ Предварительный просмотр.**
8. Просмотрите отчет. Если нужно, отредактируйте размеры и расположение полей и их подписей, вернувшись в режим конструктора.
9. Завершите редактирование и закройте отчет.

2.1.9.3 Создание кнопочной формы

Задание 5. Создайте кнопочную форму для работы с базой данных

Туристическая фирма. Форма должна загружаться при открытии БД.

Порядок выполнения:

1. Создайте новую форму в режиме конструктора (в качестве источника данных ничего не указывайте).

Примечание: Создать главную кнопочную форму можно также с помощью **Диспетчера кнопочных форм**, который загружается в окне базы данных командой **Сервис→ Служебные программы→ Диспетчер кнопочных форм.**

2. Создайте кнопку для открытия формы **Поездки** по технологии, описанной в пункте 22 задания 3 лабораторной работы № 8. Укажите имя открываемой формы – **Поездки**

, установите переключатель **Открыть форму и показать все записи,** разместите на кнопке соответствующий текст.

3. Аналогично создайте кнопки для открытия форм **Горящие путевки и Оформление заявки.**
4. Создайте кнопки для запуска на выполнение следующих запросов: **Клиенты, которым нужна виза, Стоимость поездки со скидкой, Поездки по типам, Поездки по странам.** Для этого в мастере создания кнопки выберите категорию **Разное** и действие **Выполнить запрос**. Разместите на кнопках соответствующие подписи, указывающие, какой запрос открывает кнопка.
5. Создайте кнопки для просмотра отчетов **Поездки** и **Поездки клиентов**. Для этого в мастере создания кнопки выберите категорию **Работа с отчетом** и действие **Просмотр отчета**. Разместите на кнопках соответствующие подписи, указывающие, какой отчет открывает кнопка.
6. Аналогично создайте кнопки для просмотра отчета **Письма для клиентов,** задайте действие – **Печать отчета**.
7. Создайте кнопку для закрытия БД и выхода из *Microsoft Access.* Для этого в мастере создания кнопки необходимо выбрать категорию **Приложение** и действие **Выйти из приложения**.
8. Оформите форму и ее элементы по технологиям, описанным в разделах 3 и 4 предыдущей лабораторной работы. Вставьте рисунок из коллекции *Microsoft Office* в форму, используйте элементы **Линия** и **Прямоугольник**.
9. Выполните команду **Правка→ Выделить форму,** вызовите окно *свойств* формы и на вкладке *Макет* задайте следующие свойства для формы:

Полосы прокрутки – отсутствуют; Область выделения – нет; Кнопки перехода – нет; Разделительные линии – нет; Кнопка оконного меню – нет; Кнопки размеров окна – отсутствуют; Кнопка закрытия – нет.

10. Сохраните форму с именем **Главная форма**.
11. Для загрузки кнопочной формы одновременно с открытием базы данных в окне *Параметры запуска,* вызываемого с помощью команды **Сервис→ Параметры Запуска**, выберите в списке *Вывод формы/страницы* главную форму.

12. Завершите работу с *Microsoft Access*.
13. Откройте базу данных **Туристическая фирма.** Проверьте работу объектов БД, используя кнопочную форму.

2.1.9.4 Создание отчета с помощью Мастера

Задание 2. Создайте с помощью мастера отчет на основе таблицы **Поездки.** Отчет должен содержать группировку по полю ***Страна***, сортировку по полю ***Маршрут***, среднюю стоимость поездки в страну.

Порядок выполнения:

1. Нажмите кнопку **Создать.**
2. Выберите в качестве источника данных таблицу **Поездки**.
3. Укажите способ создания отчета – *Мастер отчетов*.
4. В диалоговом окне *Создание отчетов* с помощью кнопки

переместите из списка **Доступные поля** в список **Выбранные поля** следующие поля таблицы **Поездки:** ***Страна, Маршрут, Количество Дней, ТипПоездки, Размещение, ВидТранспорта, Питание, Стоимость.***

5. На следующем шаге *Мастера отчетов* необходимо определить уровни группировки в отчете. *Мастер отчетов* предлагает группировку по полю ***Страна***. Оставьте группировку без изменения.
6. На четвертом шаге *Мастера отчетов* выберите порядок сортировки и вычисления, выполняемые для записей:

- для задания группировки в раскрывающемся списке выберите поле ***Маршрут*** и укажите порядок сортировки – *по возрастанию*.
- для вычисления средней стоимости поездки в страну нажмите кнопку **Итоги**. В появившемся окне *Итоги* установите флажок Avg для поля ***Стоимость***. Для возврата к диалоговому окну *Мастера отчетов* нажмите кнопку **ОК**.

7. Нажмите кнопку **Далее** для перехода к следующему шагу *Мастера отчетов*.
8. На следующем шаге *Мастер отчетов* предложит варианты оформления сгруппированных данных. Выберите в группе *Макет* переключатель **ступенчатый**. Для задания ориентации страницы установите переключатель **альбомная**.
9. На шестом шаге *Мастер отчетов* предложит стили оформления отчета. Выберите

понравившийся стиль.

10. На последнем шаге задайте имя отчета – **Поездки** и нажмите **Готово**.
11. В результате откроется созданный отчет в режиме просмотра. Просмотрите и закройте отчет.

2.1.10 Организация запросов в языке SQL

Введение в язык SQL

2.1.11 Программирование в среде MS Access

Êîíñòðóèðîâàíèå ìàêðîñîâ

Ìàêðîñ – íàáîð èç îäíîé èëè íåñêîëüêèõ ìàêðîêîìàíä, êàæäàÿ èç êîòîðûõ âûïîëíÿåò ïîðåäåëåííîå äåéſòâèå.

Îòêðûòèå òàáëèö

1. Îòêðûòü áàçó äàííûõ «Òóðôèðìà».
2. Âûçâàòüêîìàíäû **Ñîçäàíèå, Ìàêðîñ** . Îòêðîåòſÿ êîíſòðóêòîð äëÿ ſîçäàíèÿ ìàêðîſà.
3. Sîçäàòü Ìàêðîſ èç êîìàíäû «**Îòêðûòü òàáëèöó**». «**Èìÿ òàáëèöû** » – âûáðàòü òàáëèöó *Ïîåçäêè* .
4. Sîõðàíèòü ìàêðîſ.

Sîçäàåì åùå ÷åòûðå ìàêðîſà äëÿ îòêðûòèÿ îſòàëüſûõ òàáëèö.

Ìàêðîſû íå ſâÿçàſſûå ſſîáûòèÿìè. Èſſïëüçîâàſíèå óſëîâèé â ìàêðîſå.

Îòáîð çàïèſåé ïî óſëîâèþ

5. Îòêðûòü áàçó äàííûõ «Òóðôèðìà».
6. Âûçâàòüêîìàíäû **Sîçäàíèå, Ìàêðîſ** . Îòêðîåòſÿ êîíſòðóêòîð äëÿ ſîçäàíèÿ ìàêðîſà.
7. Sîçäàòü Ìàêðîſ èç êîìàíäû «**Îòêðûòü ôîðìó** ». «**Èìÿ ôîðìû** » – âûáðàòü ôîðìó *Ïîåçäêè* , â «**Óſëîâèå îòáîðà** », èſſïîëüçóÿ êîſſîêſ « **Ïſſòðîèòåëü** » ſſòðîèòü âûðàæåíèå. Èç òàáëèöû ſſîâåſſòâóþſſåé äàſſſîé ôîðìå âûáðàòü ðåêâèçèò è çàäàòü óſëîâèå, íàïðèìåð [Âèä_Òðàſſïîðòà]="àâòîáóſ".
8. Sſîõðàíèòü ìàêðîſ.

9. Çàïóñòèòüìàêðîñ. Äëÿ âûïîëíåíèÿ ìàêðîñà ïî øàãàì: Îòêðûòü ìàêðîñ â ðåæèìå Êîíñòðóêòîðà è âûïîëíèòü: **Ïî øàãàì, Âûïîëíèòü** .

Ìàêðîñ, ñâÿçàííûå ñîáûòèÿìè ýëåìåíòîâ óïðàâëåíèÿ â ôîðìå

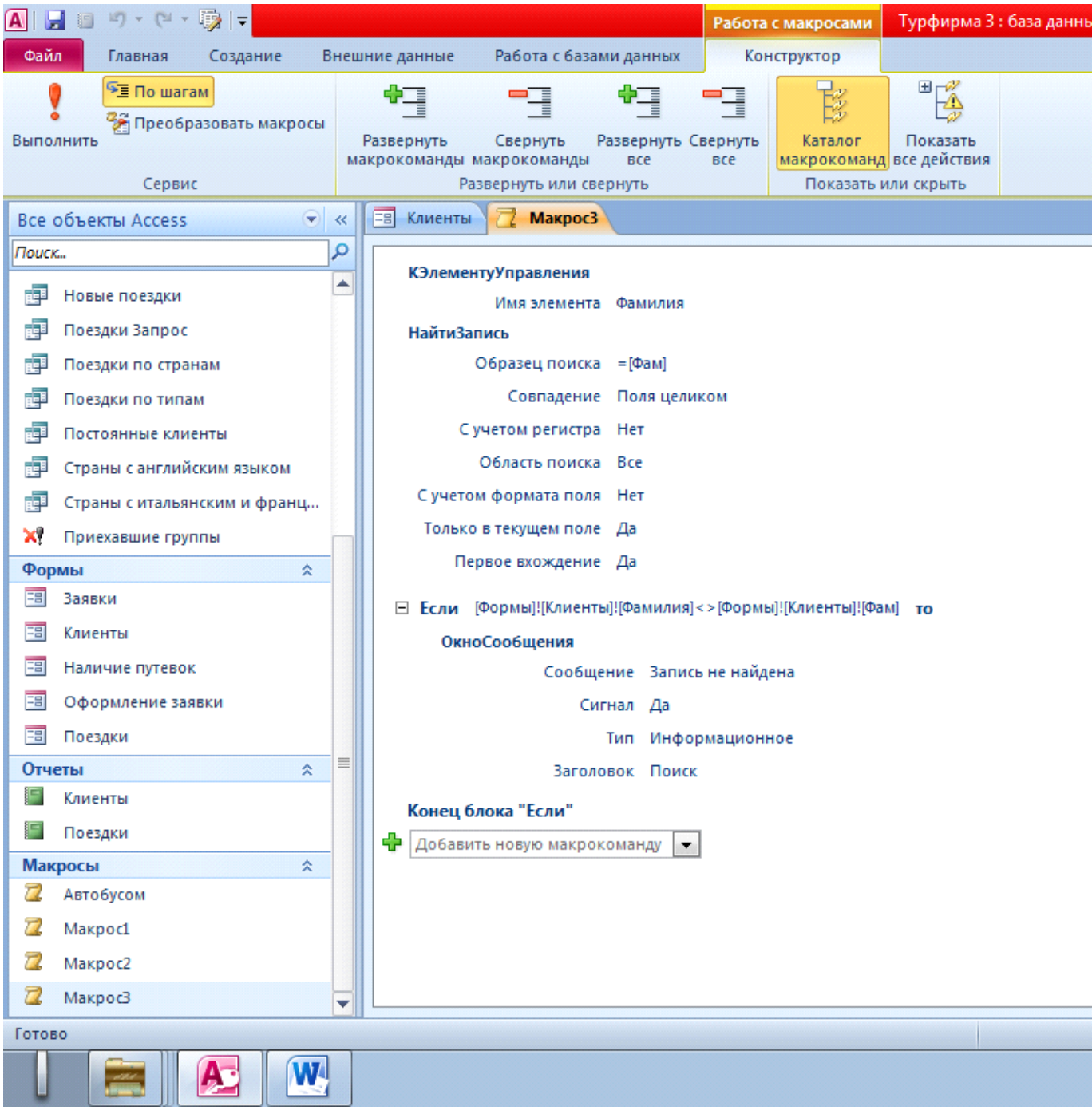
Îòâîð çàïèñåé ïî óñëîâèþ

1. îòêðûòü ôîðìó *Ïîåçäêè* â ðåæèìå êîíñòðóêòîðà.
2. Ñîçäàòü ïîëå «êíîïêà» â ôîðìå.
3. ïîäïèñàòü êíîïêó «Îòáîð».
4. äëÿ êíîïêè çàäàòüñâîéñòâà: **Ñîáûòèÿ, Íàæàòèå êíîïêè, Âíåøíåíûé ìàêðîñ (…)**.
5. Ñîçäàòü Ìàêðîñ èç êîìàíä «**Ïðèìåíèòü ôèëüòð**», â «**Óñëîâèå îòáîðà** », èñïîëüçóÿ **Ïîñòðîèòåëü** ïîñòðîèòü âûðàæåíèå. Èç òàáëèöû ñîîòâåòñòâóþùåé äàííîé ôîðìå âûáðàòü ðåêâèçèòè è çàäàòü óñëîâèå, íàïðèìåð [Âèä_Òðàíñïîðòà]="àâòîáóñ".

Äëÿ ïðîâåðêè ðàáîòû ìàêðîñà îòêðûòü ôîðìó â ðåæèìå ôîðìà.

Ïîèñê çàïèñè â ôîðìå

1. îòêðûòü ôîðìó Êëèåíòû â ðåæèìå êîíñòðóêòîðà.
2. Ñîçäàòü ñâîáîäíîå ïîëå ââîäà ab â ôîðìå.
3. Â íàäïèñè íàïèñàòü «Ââåäèòå òàìëëþ»
4. Äëÿ ïîëÿ «Ñâîáîäíûé» çàäàòüñâîéñòâî Èìÿ: Òàì.
5. Ñîçäàåì ìàêðîñ, ñîñòîÿùèé èç ìàêðîêîìàíä: **«ÊÝëåìåíòóÓïðàâëåíèÿ», «ÍàéòèÇàïèñü», «Åñëè»,«ÎêíîÑîîáùåíèÿ»:**
6.



7. Ìàêðîñ3 äîëæåí íà÷èíàòü ñâîþ ðàáîòó ïîñëå ââîäà êîíêðåòíîé ôàìèëèè â ïîëå «Ôàì». Ïîÿâèòñÿ åãî ñëåäóþùèì ñîáûòèåì «Ïîñëå îáíîâëåíèÿ» äëÿ ýòîãî ïîëÿ. Äëÿ ñâÿçè íåîáõîäèìî: Îòêðûòü ôîðìó â ðåæèìå êîíñòðóêòîðà, îòêðûòü îêíî Ñâîéñòâ äëÿ ïîëÿ «Ôàì» è âûáðàòü èìÿ ìàêðîñà «Ìàêðîñ3» èç ñïèñêà â ñòðîêå «Ïîñëå îáíîâëåíèÿ» íà âêëàäêå «Ñîáûòèÿ». Çàêðûòü îêíî ñâîéñòâ.

Ïðîâåðêà ðàáîòû ìàêðîñà

1. Îòêðûòü ôîðìó â ðåæèìå ôîðìà.
2. Â ïîëå «Ôàì» ââîäèì ôàìèëèþ ñòóäåíòà, êîòîðàÿ íåîáõîäèìà.
3. Çàòåì ââîäèì ôàìèëèþ â ïîëå «Ôàì», êîòîðàÿ îòñóòñòâóåò.

Ñàìîñòîÿòåëüíî. Ñîçäàòü ìàêðîñû, ñâÿçàííûå ñîîáùèòåëüíûé ýëåìåíòû óïðàâëåíèÿ â îò÷åòå.

2.1.12 Объектно-ориентированные базы данных

Объектно-ориентированные базы данных

2.1.13 Администрирование баз данных

Управление БД в СУБД MS Access

- Пользователи БД. Функции администратора БД.
- Методы защиты БД (защита паролем, разграничение прав доступа, шифрование).
- Создание резервных копий БД. Сжатие БД. Восстановление БД.
- Оптимизация работы БД.
- Установка параметров запуска

1. Пользователи БД. Функции администратора БД

При работе с БД можно выделить две категории людей: *пользователи БД* и *обслуживающий персонал*.

Пользователь (User) – это физическое или юридическое лицо, которое пользуется услугами компьютерной системы для получения информации либо решения соответствующих задач.

Существуют различные категории пользователей:

- аналитик
- системный программист
- прикладной программист
- администратор системы
- оператор компьютера
- конечный пользователь *(это люди, которым требуется информация из БД для выполнения прямых служебных обязанностей, например, бухгалтер, руководитель, менеджер отдела продаж*)
- пользователями базы данных могут быть прикладные программы и программные

комплексы.

База данных функционирует в соответствующей социальной среде со множеством пользователей. Пользователи могут предъявлять противоречивые требования к базе данных, поэтому нужно постоянно искать компромиссное решение. База данных находится под централизованным управлением. Доступ пользователя к базе данных обеспечивается СУБД. Архитектура СУБД создается по концепции многоуровневой организации. Четкость и эффективность ее работы обеспечиваются администрированием базы данных.

Обслуживающий персонал – люди, ответственные за работу информационной системы, использующей базы данных, и соответствующего прикладного программного обеспечения.

К обслуживающему персоналу относятся системные администраторы, разработчики системного и прикладного обеспечения, администраторы БД.

Администратор базы данных – это лицо или группа лиц, задачей которых является поддержка нормальной работы приложения в течение всего периода его использования.

В обязанности администратора базы данных входит:

- определение информационного содержания базы данных, внутренней структуры
- определение стратегии доступа к базе данных
- обеспечение охраны информации, ее целостности
- контроль производительности работы базы данных
- реагирование на требования пользователей

Поскольку спектр этих задач достаточно широк, функции администратора баз данных объединяют в следующие группы:

- администрирование предметной области
- администрирование баз данных
- безопасности данных и приложений

Поддерживаются функции администрирования специальными служебными программами - утилитами. Для администрации баз данных предусмотрены соответствующие инструкции.

В соответствии с разными функциями пользователи и администратор БД имеют разные права доступа к БД.

Пользователи обычно имеют право открывать БД, просматривать и модифицировать только отдельные объекты БД в рамках предоставленных им полномочий.
Администратор БД наделен всеми правами по управлению БД и ее модификации, а также по изменению прав доступа пользователей.

Просмотр и печать сведений о базе данных и ее объектах

Просмотр списка объектов текущей базы данных.
Для просмотра всей иерархии объектов файла базы данных следует при открытой базе данных дать команду *Файл - Свойства базы данных*

Просмотр объектов
Скрытые объекты по умолчанию можно отобразить в окне базы данных, не отменяя их атрибуты

скрытия. Для этой цели используется команда *Сервис – Параметры* - вкладка *Вид.* На вкладке *Вид* нужно установить или снять флажок *Скрытые объекты.*

Отображение или скрытие системных объектов.
Отображение или скрытие системных объектов в окне базы данных устанавливается следующим образом:

- выполнить команду *Сервис – Параметры*
- выбрать вкладку *Вид* и установить или снять флажок *системные объекты.*

При создании новой базы данных автоматически создаются системные объекты, чьи имена начинаются символом *Usys.*

Просмотр и печать характеристик объекта базы данных
Просмотр и печать характеристик объекта базы данных выполняется следующим образом:

- выполнить команду *Сервис - Анализ – Архивариус* - вкладка соответствующего типа объекта, который желательно рассмотреть или вывести на печать (для полного списка объектов – вкладка *Все типы объектов*).
- выбрать объекты для просмотра (печати), затем нажать кнопку *Параметры* и *ОК*
- выполнить требуемое действие: *для печати* – кнопка *Печать* на панели инструментов, для сохранения описания в виде таблицы – в меню *Файл - Сохранить как - таблица.*

2. Методы защиты БД (защита паролем, разграничение прав доступа, шифрование)

Для предотвращения несанкционированного доступа к объектам базы данных предназначены средства защиты информации, которые входят в СУБД. С их помощью системе указываются пользователи, которым дано или не дано разрешение на доступ к объектам базы данных.

При большом количестве пользователей можно определить их группы и разрешить им доступ к базе данных

Стандартный файл рабочей группы (System.mdw) создается при установке Access программой Setup. Файл System.mdw содержит информацию о всех пользователях и группах и применяется по умолчанию при запусках Access. Последний сохраняет информацию о правах доступа при создании базы данных в файле базы данных. Разрешение на доступ к конкретным объектам базы данных сохраняется в файле рабочей группы.
База данных, созданная пользователем рабочей группы, принадлежит ее создателю и рабочей группе. Каждая рабочая группа и каждый пользователь имеют уникальные внутренние идентификаторы. При копировании базы данных в другую папку или компьютер вместе с файлом базы данных перемещаются и права на доступ.
Компьютерные системы защиты информации бывают открытыми или закрытыми. В открытой системе информация открыта для всех пользователей (кроме информации, имеющей специальную защиту), в закрытой – только по назначении доступа.

Являясь корпоративным ресурсом значительной ценности, БД требует контроля и защиты. Для защиты БД, работающих в локальной сети, можно предложить следующий комплекс мер обеспечения безопасности, сочетающий в себе возможности, предоставляемые сетевой операционной системой (ОС) и СУБД.

На сетевую ОС возлагаются следующие функции:

- Каждый пользователь сети перед получением доступа к ресурсам сети должен пройти процедуру идентификации. Для идентификации пользователя используется уникальное имя пользователя и секретный личный пароль пользователя. Пользователи не должны сообщать свой пароль другим лицам, смену пароля следует обязательно проводить каждые 60-90 дней.
- Каждый идентифицированный пользователь сети должен подтверждать свое право на доступ к отдельным элементам сети, например, папкам на сервере или другим разделяемым ресурсам.
- Контроль действий пользователя сети с целью регистрации попыток доступа к тем ресурсам сети, права доступа к которым у данного пользователя отсутствуют. Те пользователи, которые регулярно пытаются преодолеть защиту сети, должны отключаться от сети до принятия соответствующих административных мер.
- Сеть должна быть защищена от взлома и внедрения вирусов с помощью особых систем безопасности.
- Данные, хранящиеся на серверах сети, должны быть защищены от искажений и потерь, адекватным и своевременным резервированием. Системы резервирования позволяют восстановить данные на момент последнего резервирования.

С помощью средств СУБД осуществляются следующие меры:

- Содержимое БД должно быть зашифровано, чтобы предотвратить просмотр данных с помощью программ просмотра файлов или средств работы с диском (например, входящих в состав Norton Utilities) или других программ. Шифрование – преобразование читаемого текста в нечитаемый при помощи некоторого алгоритма. Зашифрованные данные невозможно прочитать не зная метода шифрования. Шифрование БД предотвращает чтение содержащихся в ней данных с помощью иных средств, чем средства СУБД. Шифрование несколько замедляет работу с БД, поскольку на процесс шифрования / дешифрования требуется определенное время.
- Перед тем, как открыть файл БД, пользователь должен проходить повторную идентификацию. При этом используется пароль, отличный от пароля доступа к сети (защита паролем). Установка пароля является простейшим способом частичного обеспечения безопасности, который предотвращает (затрудняет) доступ к БД лиц, не имеющих права на работу с БД.
- Каждый пользователь БД должен иметь определенные права доступа к БД. СУБД должна предоставлять возможности по разграничению доступа.

Разграничение прав доступа

Разграничение прав доступа позволяет предоставить пользователю доступ только к ограниченной части БД и наделить его только определенными правами по работе с данными (право чтения, право ввода, право обновления, право удаления).

Наиболее гибкий и распространенный способ защиты базы данных называется защитой на уровне пользователей. От пользователей требуется идентифицировать себя и ввести пароль, когда они открывают БД. Обычно каждому пользователю предоставляются полномочия (разрешения), определяющие возможность их доступа к каждому объекту базы данных.

Пользователю нужно идентифицировать себя и ввести пароль. Пользователи идентифицируются как члены группы в файле рабочей группы. Рабочая группа – это группа пользователей, которые в сети совместно пользуются базой данных.
Файл рабочей группы содержит данные о пользователях группы и считывается во время запуска. Этот файл содержит следующую информацию: имена учетных записей пользователей, пароли пользователей, имена групп, в которые входят пользователи.

При установке MS Access создается стандартная рабочая группа, которая содержит один встроенный код пользователя с названием Admin и два встроенных кода группы (User и Admins). Access автоматически загружает пользователя с его кодом и предоставляет все права и

привилегии.
Встроенный код группы User включает в себя всех пользователей (членов) группы и автоматически представляет полные права доступа к объектам базы данных. Пользователям предоставляется разрешение на доступ к каждому объекту базы данных.

Для каждого файла рабочей группы при создании файла устанавливается ее уникальный внутренний идентификатор. По умолчанию в эту группу включен только пользователь Admin. Пользователи группы Admin могут определять и изменять учетные записи пользователей и групп, устанавливать и менять личные пароли, имеют полный доступ к любым базам данных, созданным при использовании этого файла рабочей группы. Любая база данных, которая создана с применением конкретного файла этой группы, наследует ее код.

Типы разрешений, которые могут быть назначены базе данных или ее объектам: открытие/запуск, монопольный доступ, чтение макета, изменение макета, разрешение администратора, чтение данных, обновление данных, вставка данных, удаление данных.

Проверка разрешения пользователю или группе на доступ к объектам базы данных выполняется в такой последовательности:

- открыть нужную базу данных. Необходимо быть собственниками базы данных и ее объектов или иметь разрешение администратора на доступ
- дать команду Сервис-Защита - Разрешение. В окне имеется две вкладки: Разрешение и Смена владельца. Выбрать вкладку Разрешение. В окне будут отображены пользователи и группы, которые определены в базе данных
- выбрать Admin (группа) или Users (пользователи) – в списке Имя объектов будут выведены объекты базы данных. В окне можно изменить тип объекта базы данных в раскрывающемся списке Тип объекта. После этого в части окна Разрешения: флажки укажут на явные разрешения доступа

Вкладка Смена владельца позволяет назначить нового владельца (пользователя или группу)

Мастер защиты базы данных

Для установки защиты базы данных на уровне пользователя Access имеет Мастер защиты. Перед непосредственным запуском Мастера защиты необходимо:

- сформировать новую рабочую группу с уникальным кодом, отличным от кода рабочей группы, где создавалась база данных
- ввести новый код пользователя, отличный от Admin (для владельцев базы данных Admin нельзя установить защиту). Если пользователь зарегистрирован в первичной рабочей группе как пользователь Admin, то Мастер предложит создать новую рабочую группу
- зарегистрироваться под именем владельца базы данных или в файле рабочей группы, который применялся при создании базы данных, и быть членом группы Admins
- удалить для всех объектов разрешения с группы Users

Запуск мастера защиты данных

- дать команду *Сервис – Зашита – Мастер.* В первом окне диалога Мастер защиты содержится краткое описание работы Мастера и предлагается два варианта: *Создать файл рабочей группы* или *Изменить текущий файл рабочей группы.*
  При выборе опции *Создать файл рабочей группы* Мастер создаст новый файл рабочей группы, создаст в этой рабочей группе новый код пользователя (не Admin) и установит защиту для базы данных в новой рабочей группе и нового пользователя
- перейти во второе окно Мастера путем нажатия кнопки *Далее*. По умолчанию Мастер защищает все объекты базы данных. Можно отменить (снять флажок) защиту отдельных

объектов
- нажать кнопку *Далее* – перейти в третье окно. Мастер предлагает создать одну или несколько дополнительных групп. В списке имен групп можно выбрать нужную группу и увидеть разрешения на доступ для этой группы
- нажать кнопку *Далее* – перейти в четвертое окно Мастера, где представлены разрешения универсальной группы Users
- нажать кнопку *Далее* – перейти в пятое окно Мастера и добавить пользователей (если нужно) в файл рабочей группы
- для каждого пользователя задать пароль и уникальный личный код
- нажать кнопку *Далее* – перейти в шестое окно Мастера, где можно включить пользователей в нужные группы
- нажать кнопку *Далее* – перейти в седьмое окно Мастера, где задать имя для резервной копии файла базы данных
- нажать кнопку *Готово* для завершения работы Мастера

Защита базы данных Access с помощью пароля

Самый простой способ защиты базы данных – с помощью пароля. Можно назначить пароль базе данных Access, который будет требоваться всякий раз при её открытии.

Установка и снятие пароля защиты базы данных

Чтобы установить пароль для защиты базы данных:
1. Закройте базу данных. Если база данных совместно используется в сети, убедитесь, что остальные пользователи её закрыли.
2. Сделайте резервную копию базы данных и сохраните её в надёжном месте.
3. В меню Access выберите команду *Файл - Открыть.*
4. Выделите файл базы данных.
5. Щёлкните по стрелке, расположенной справа от кнопки *Открыть*. В раскрывающемся списке режимов открытия базы данных выделите элемент Монопольно. База данных откроется в режиме монопольного доступа.
6. Выберите команду *Сервис – Защита - Задать пароль базы данных*. Появится диалоговое окно *Задание пароля базы данных*.
7. Введите в поле *Пароль* пароль для защиты базы данных с учётом регистра символов.
8. Введите пароль ещё раз в поле *Подтверждение*.
9. Нажмите кнопку OK.

Теперь база данных защищена паролем и всякий раз, когда пользователь будет открывать базу данных, будет отображаться диалоговое окно с требованием ввести пароль. Запомните или сохраните пароль в надёжном месте. Если пароль будет забыт, невозможно будет открыть базу данных.

Если база данных защищена на уровне пользователей, установить пароль для её открытия может только пользователь, обладающий административными правами
Установка пароля не влияет на систему защиты на уровне пользователя. Эти два способа защиты могут использоваться одновременно. Пароль базы данных сохраняется в базе данных, а не в файле рабочей группы.

Чтобы удалить пароль защиты базы данных:

10. Откройте базу данных в режиме монопольного доступа.
11. В диалоговое окно Необходимо ввести пароль введите пароль.

12. Выберите команду *Сервис – Защита - Удалить пароль базы данных*. Появится диалоговое окно *Удаление пароля базы данных*.
13. Введите текущий пароль базы данных.
14. Нажмите кнопку ОК.

Установка связи с таблицами базы данных, защищенной паролем

Чтобы установить связь с таблицами базы данных, защищенной паролем, требуется ввести пароль. Если пароль был введён верно, он сохраняется вместе с другой информацией о ссылках на таблицы. После этого любой пользователь, работающий с базой данных со связанными таблицами, может открыть эти таблицы без указания пароля. Если пароль защищенной базы данных будет изменен, в следующий раз при открытии базы данных, содержащей связанные таблицы, потребуется ввести пароль.


Microsoft Access сохраняет пароль в базе данных, содержащей связанные таблицы защищенной базы данных, в незашифрованном виде. Если это уязвляет систему защиты базы данных, не используйте средства защиты с помощью пароля. Установите систему защиты на уровне пользователей, чтобы ограничить доступ к объектам базы данных.

Защита информации с помощью шифрования

Защита информации в базе данных Access может производиться с помощью операции шифрования. Операция шифрования приводит к сжатию файла базы данных. После выполнения операции шифрования просмотр данных с помощью служебных программ и текстовых редакторов становится невозможен. Операция шифрования отменяет результаты операции шифрования. Для проектов Access 2002 операция шифрования не реализована.


Операция шифрования или дешифрования неприменима к открытой базе данных. Если база данных используется в сети, то, прежде чем приступить к выполнению этих операций, убедитесь, что она не открыта ни одним из пользователей.

Чтобы выполнить операцию шифрования или дешифрования:

1 . Запустите Access.
2 . Не открывая базу данных, выполните команду *Сервис - Защита - Шифровать/дешифровать*
3. Появится диалоговое окно *База данных для шифрования или дешифрования*. Укажите имя базы данных, которую требуется зашифровать или дешифровать. Кнопка ОК.
4. Если выбранная на предыдущем шаге база данных не является шифрованной, появится диалоговое окно *Шифрование базы данных под именем,* в другом случае появится диалоговое окно *Дешифрование базы данных под именем*. Укажите имя, диск и папку для конечной базы данных и нажмите кнопку ОК.

 Можно указать новое имя или имя, совпадающее с именем исходного файла. Если указаны те же имя, диск и папка, а шифрование или дешифрация выполнены успешно, то исходный файл будет заменён зашифрованным или дешифрованным. Если же операция приводит к ошибке, то исходный файл удалён не будет. Необходимо иметь достаточно места на диске для двух версий базы данных – шифрованной и дешифрованной.

Если для базы данных определена защита на уровне пользователей, то для её шифрования или дешифрования необходимо иметь разрешение на изменение Макета для всех таблиц базы данных. В противном случае попытка шифрования или дешифрования окажется неудачной. Шифрование базы данных не препятствует доступу к ней легальных пользователей или групп. Разрешение на доступ к объектам являются частью системы защиты базы данных на уровне пользователей.

3. Создание резервных копий БД. Сжатие БД. Восстановление БД.

Резервирование БД

Одним из способов защиты БД от разрушения является ее регулярное резервирование, суть которого состоит в копировании БД на другой носитель информации (например, дискету, стример). Резервные копии создаются на случай порчи БД, вызванной либо физическими причинами, либо вводом неверных данных, либо использованием программ, содержащих ошибки.

При создании резервных копий придерживайтесь следующих принципов:

- интервал между операциями резервирования определяется количеством данных, которое нужно ввести заново в случае порчи БД;
- делайте резервные копии на различные комплекты носителей, так как на одном из устройств могут обнаружиться повреждения, не позволяющие восстановить информацию;
- храните часть копий вне основного офиса на случай стихийных бедствий.

Кроме резервного копирования может быть использовано следующее:

- Обновление БД протоколируется (предпочтительно в другой БД) с указанием имени пользователя, производившего изменения, даты и времени изменения. Такие протоколы – средство контроля и восстановления информации в случае ее потери.
- Операции по обновлению данных выполняются с использованием техники транзакций, которые могут быть в случае необходимости отменены, а ситуация возвращена в исходное состояние.

Резервирование есть процесс создания копии программы, базы данных для создания архива или сохранения особенно ценных файлов.

Резервирование как процесс является стратегией в современных СУБД: восстановление в базе данных последней операции после того как определенная аппаратная либо программная ошибка сделала базу данных непригодной для работы. Этот процесс начинается с последней, резервной, копии базы данных.

Создание резервной копии базы данных:

1. закрыть базу данных. При работе в сети убедиться в том, что все пользователи закрыли базу данных
2. скопировать файл базы данных в резервный каталог папки «Мой компьютер» с помощью проводника Windows или программой архивации файлов MS Backup, либо другой программой архивации

Для архивации отдельных объектов базы данных (таблиц, запросов, форм, отчетов, макросов, модулей) можно создать пустую базу данных, а затем импортировать в нее нужные объекты из исходной базы данных.

Понятие репликации базы данных

Репликация (replication – копия, копирование*)* – это создание не одной копии, а набора копий базы данных*.*

Репликация позволяет пользователям, работающим за разными компьютерами, обмениваться изменениями, вносимыми в базу данных, и осуществлять одновременный доступ к собственным копиям. Изменения, вносимые в основную реплику, распределяются по всем реплицируемым объектам – таблицам, формам, запросам, отчетам, макросам, модулям.

В процессе работы с репликами (копиями базы данных) можно синхронизировать данные с помощью специальных программ Access.
Синхронизация (*synchronization*) – организация действий в определенной последовательности или параллельно – поддерживает целостность базы данных при ее использовании многими пользователями, обеспечивает временное упорядочение действий параллельных процессов с базой данных.

Отдельные экземпляры копий базы данных используются различными пользователями: можно взять базу данных с собой в командировку на портативном компьютере, распространить базу данных по многочисленным филиалам фирмы и др. После работы с копиями базы данных ее можно синхронизовать.

В создаваемом наборе копий базы данных одна копия является *основной репликой (копией)* – именно в основную копию разрешается вносить изменения (в структуру таблиц, форм, запросов, отчетов, макросов и модулей). На основе любой реплики можно создавать новые реплики.

Копии базы данных могут находиться на исходном компьютере и на любом компьютере локальной сети, а также на компьютере, подключенном через Internet к локальной сетке.

С копиями базы данных можно работать, изменять данные. Для поддержания базы данных, которая в виде ее копий используется многими пользователями, выполняется ее синхронизация. Если в одной и той же записи в две реплики внесены различные изменения, то возникает конфликтная ситуация, которую помогает разрешить синхронизация.

Для организации работы с реплицированной базой данных, отслеживания вносимых изменений в структуру и данные Access автоматически добавляет скрытые системные таблицы.

Перед преобразованием базы данных в основную реплику рекомендуется сделать в другой папке запасную копию базы данных. Если этого не сделать, то Access сам предложит создать резервную копию базы данных (.bak). Реплики можно создавать, только имея право на открытие базы данных в монопольном режиме.

Для преобразования базы данных в основную реплику и создания реплики следует:

- открыть базу данных
- дать команды *Сервис – Репликация - Создать реплику*

В окне диалога система предупреждает, что перед созданием реплики базу данных необходимо закрыть, и настоятельно рекомендуется создать резервную копию базы данных перед ее преобразованием.

После создания реплики Access выдает следующее сообщение «Приложение Access преобразовало (имя базы данных .mdb) в основную реплику набора. Изменения структуры базы данных допускаются, как уже упоминалось, только в основной реплике, изменение данных могут выполняться как в основной, так и в любой другой реплике набора.

Дополнительные реплики создаются той же командой *Создать реплику*.

Синхронизация реплик – это процесс обновления двух реплик при взаимной передаче обновленных записей и объектов.

Реплики обмениваются данными как в одностороннем, так и двустороннем порядке.

После внесения изменений в структуру основной реплики (например, перед печатью объектов базы данных) следует проводить синхронизацию данных в основной и не основной реплике. По команде *Синхронизация* происходит обмен данными между элементами набора реплик.

Последовательность действий

- открыть основную или любую другую реплику
- команда *Сервис-Репликация-Синхронизация*

Конфликты при синхронизации базы данных просматриваются и устраняются командой *Сервис – Репликация - Устранить конфликты*. Выбирается нужный вариант, например из двух различных изменений по одной и той же записи.

Выход из конфликтной ситуации между элементами набора реплик:

- открыть проверяемую базу данных
- указать таблицу-источник, которая могла вызвать конфликт, дать команду *Сервис – Репликация - Удалить конфликт*
- при наличии конфликта система задает вопрос о необходимости его разрешения, ответить *Да*
- для каждой конфликтной записи принять решение и нажать соответствующую кнопку *Сохранить существующую* или *Заменить на конфликтную*

Восстановление поврежденной БД

СУБД Access следит за состоянием базы данных: при появлении повреждений во время манипуляций с ней дает соответствующее сообщение; имеет средства для восстановления базы данных.

При получении сообщения о повреждении БД или необычном поведении можно предположить, что один или несколько объектов БД повреждены.

БД может быть повреждена в следующих случаях:

- при возникновении аппаратных ошибок во время записи БД
- при аварийной перезагрузке в момент, когда БД была открыта
- скачок напряжения в сети;

Восстановление открытой базы данных:
команды: *Сервис - Служебные программы - Восстановить базу данных* - выдается сообщение об успешном восстановлении базы данных и указывается путь к базе данных

Восстановление неоткрытой базы данных:

- закрыть активную базу данных
- подать команды *Сервис - Служебные программы - Восстановить базу данных*
- указать имя восстанавливаемой базы данных и каталог (папку)
- нажать кнопку *Восстановить.*

Сжатие базы данных или проекта Access 2002

При удалении данных или объектов файл базы данных или проекта становится фрагментированным. Это приводит к тому, дисковое пространство используется неэффективно. Сжатие базы данных или проекта позволяет получить копию, в которой данные и объекты сохраняются более рационально, что значительно экономит место на диске.
Чтобы застраховаться от потери данных, создайте резервную копию базы данных или проекта Access. Начиная с версии Access 2000, сжатие и восстановление базы данных объединены в единый процесс. Однако проект Access 2002 можно только сжать и нельзя восстановить.

Создание резервной копии базы данных или проекта Access 2002.

Есть несколько путей создания резервной копии базы данных или проекта.

При наличии достаточного объёма свободного места на диске можно создать резервную копию обычным копированием файла. Создать копию файла можно с помощью приложения Проводник (Explorer), входящего в состав операционной системы Windows, с помощью команды операционной системы *copy*, с помощью инструкции *FileCopy* в процедуре на VBA (только если база данных или проект в это время не открыт в другом окне приложения Access).Чтобы сэкономить место на диске, можно создать сжатую копию файла с помощью программы архивирования, например, с помощью стандартной утилиты архивирования файлов Backup, входящей в состав операционной системы Windows, или с помощью утилит сторонних производителей, таких как WinZip или WinRar. В случае повреждения файла исходной базы данных или проекта Access 2002, его можно будет заменить резервной копией.

Если в базе данных установлена защита на уровне пользователей, необходимо создать также резервную копию соответствующего файла рабочей группы операцией копирования или архивирования соответствующего файла с расширением MDW.

Сжатие базы данных

При удалении данных или объектов файл базы данных или проекта Access становится фрагментированным, это приводит к тому, что дисковое пространство используется неэффективно. Сжатие базы данных или проекта позволяет получить копию, в которой данные и

объекты сохраняются более рационально. Это экономит место на диске.

Сжатие повышает производительность баз данных и проектов Access 2002. Однако сжатие проекта не влияет на объекты, такие как виды и таблицы, хранящиеся в базе данных СУБД SQL SERVER. Сжатие проекта не влияет также на таблицы, содержащие поле счетчика, как это происходит с базами данных. Если перед сжатием базы данных из таблицы, содержащей поле счетчика, были удалены последние записи, то после сжатия номер первой пустой записи сбрасывается. Добавленная после этого запись получит в после счетчика номер, на единицу превышающий значение счетчика в последней оставшейся записи.


Сжатие баз данных, созданных в предыдущих версиях Microsoft Access, не приводит к их преобразованию в формат Microsoft Access 2002.
Перед сжатием совместно используемой базы данных, убедитесь, что она не открыта ни одним из пользователей в сети. Для сжатия базы данных необходимо обладать правами на ее открытие, запуск и открытие в монопольном режиме.

Чтобы сжать открытую базу данных или проект Access 2002:

15. Если нужно сжать сетевую базу данных, расположенную в общей папке или на сервере, убедитесь, что она не открыта другими пользователями.
16. Выберите команду *Сервис - Служебные программы - Сжать и восстановить базу данных.*

Чтобы сжать закрытую в данный момент базу данных или проект Access 2002*:*

Если в данный момент открыта другая база данных или проект Access 2002, закройте их. Если база данных или проект расположен в общей папке или на сервере, убедитесь, что они не открыты другими пользователями.

Выберите команду *Сервис - Служебные программы - Сжать и восстановить базу данных.*

Появится диалоговое окно *База данных для сжатия*. Укажите базу данных или проект, который необходимо сжать и нажмите кнопку *Сжать*.

Появится диалоговое окно *Сжатие базы данных под именем.* Выберете диск и папку и выведите имя для сохранения сжатой базы данных. Нажмите кнопку *Сохранить.*


Допускается сжатие файла базы данных или проекта Access 2002 в файл с тем же именем, что и для исходного файла, или создание файла с новым именем. При указании того же имени, диска и папки и при успешном сжатии базы исходный файл автоматически заменяется сжатым файлом.

Можно настроить Access 2002 так, чтобы конкретная база данных или Проект автоматически сжимались при закрытии.

Чтобы установить автоматическое сжатие базы данных :

17. Откройте базу данных, которую нужно сжать.
18. Выберете команду *Сервис - Параметры*. В появившемся диалоговом окне *Параметры* раскройте вкладку *Общие*.
19. Установите флажок *Сжимать при закрытии* и нажмите кнопку ОК.

Установка автоматического сжатия проекта Access 2002 выполняется аналогично.

Автоматическое сжатие не происходит, если при этом размер базы данных (или проекта) не будет уменьшен, по крайней мере, на 256К, а также, если эта база данных (или проект) в текущий момент открыта другим пользователем в сети.

4. Оптимизация работы БД.

Способы оптимизации работы БД определяются используемой СУБД. Часто используемый способ – это процедура сжатия БД, которая обеспечивает возвращение освободившегося пространства для дальнейшего использования и оптимизацию размещения объектов БД на внешних носителях.

Многие современные СУБД имеют средства оптимизации работы запросов.
СУБД Access имеет, в частности, Анализатор быстроты для оптимизации работы БД. Для его вызова используется команда (*Сервис – Анализ – Анализ быстродействия*).
Способ увеличения объёма работы – увеличение операционной памяти. Удаление файлов и осуществление дефрагментации диска. Если единственный пользователь БД, работай в монопольном режиме.

Access анализирует быстродействие базы данных с помощью Мастера анализатора.

Для запуска Мастера анализатора нужно при открытом окне базы данных:

- дать команду: *Сервис – Анализ - Быстродействие*.

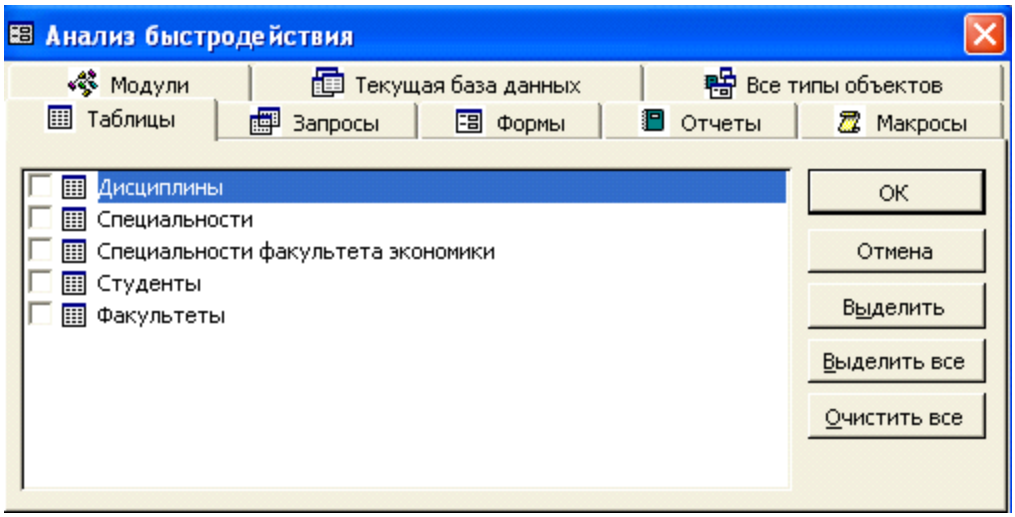


Рис 1. Окно анализатора быстродействия

- Затем выбрать категорию анализируемых объектов (текущая база данных, таблицы, отчеты и др.). Указать конкретный объект путем установки флажков рядом с их именами
- Для запуска анализатора быстродействия щелкнуть кнопку ОК. Мастер анализатор выполнит задание и выдаст

сообщение о результатах анализа в виде списка пожеланий и предложений и “мыслей”.

- Выбрать (выделить) подходящее предложение и нажать в окне анализатора кнопку *Оптимизировать* – Мастер оптимизатор внесет в базу данных соответствующие изменения.

К «мыслям» анализатора следует относиться критически, потому что принятие отдельных «мыслей» может привести к появлению большого объема дополнительной работы и мизерному повышению быстродействия.

5. Установка параметров запуска

Выполнение установки параметров запуска осуществляется командой *Сервис – Параметры запуска*. На экране появляется диалоговое окно *Параметры запуска*, представленное на рис.2.

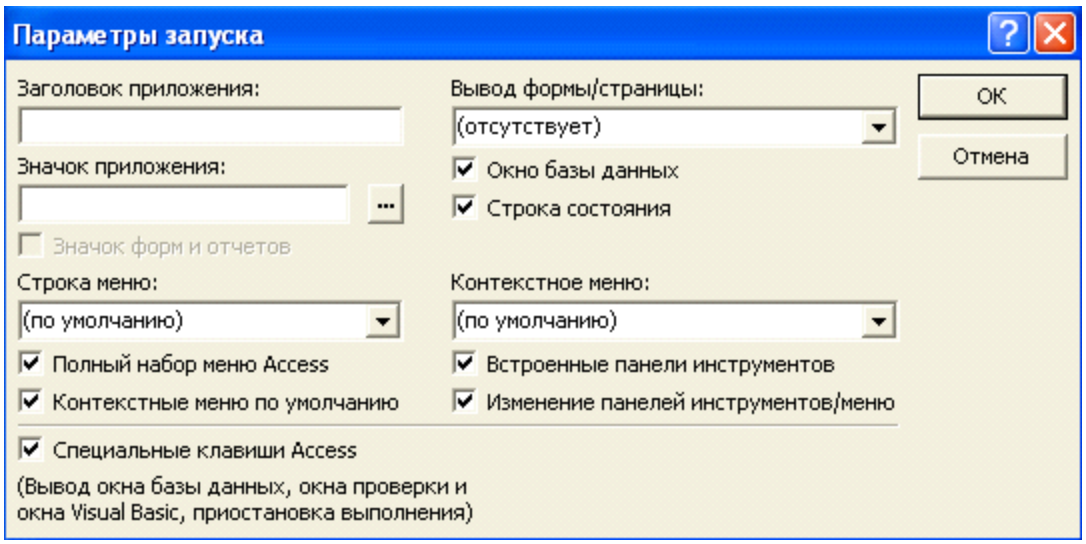


Рис. 2 Диал. окно Параметры запуска

В диалоговом окне *Параметры* запуска можно задать:

- Значок и заголовок Вашего приложения, которые будут отображаться в главном окне приложения и на панели задач вместо стандартного значка и заголовка Microsoft Access. Эти параметры вступают в силу сразу после закрытия диалогового окна Параметры запуска
- Меню, которое будет появляться при запуске вместо стандартного меню Access и определять основные функции приложения
- Форму или страницу, которая будет появляться на экране при открытии базы данных
- Специальное контекстное меню, которое будет заменять встроенные контекстные меню во всех окнах приложения, кроме тех, с которыми связаны другие контекстные меню.

Флажки *Окно базы данных* и *Строка состояния* позволяют скрывать при запуске окно базы данных и строку состояния.

Флажки *Встроенные панели инструментов* и *Изменение панелей инструментов/меню* позволяет пользователям вносить изменения в разработанное приложение.

 Все эти параметры запуска вступают в силу только после следующего запуска приложения.

2.2 Задания для самостоятельной работы

№ п/п	Содержание вопроса	Литература основная/ дополнительная	Формы контроля
1.	**Законы алгебры логики** Логическая равносильность. Законы логики: закон тождества, закон противоречия, закон исключенного третьего, закон двойного отрицания, закон идемпотентности. Применение законов де Моргана для преобразования логических формул. Законы коммутативности, ассоциативности и дистрибутивности.	3,4,5,6/1	Комплект электронных материалов
1.	**Преобразование формул с помощью законов логики** Тождественные соотношения. Законы поглощения и законы склеивания. Следствия из законов алгебры логики. Способы построения переключательных схем по заданным логическим функциям.	3,4,5,6/1	Набор тестовых заданий
1.	**Структура построения БД** Причины, обусловившие появление баз данных. Файловая организация данных в АИС, её недостатки. Предметная область. Требования, предъявляемые к БД. Понятие целостности БД. Модель данных. Реляционная модель. Схема отношения. Логические связи между отношениями. Типы логических связей. Первичный ключ. Развитие моделей организации	1,2,7,9/3,4	Реферат

№ п/п	Содержание вопроса	Литература основная/ дополнительная	Формы контроля
	данных в БД.		
1.	**Проектирование реляционной базы данных** Вторая нормальная форма. Третья нормальная форма. Физическое проектирование. Вопросы, решаемые на этапе физического проектирования.	1,2, 9/3,4	Реферат
1.	**Системы управления базами данных** Взаимодействие СУБД с другими приложениями. Публикация баз данных в Internet. Обзор современных СУБД: Oracle, Informix, Sybase, Microsoft SQL Server. Тенденции развития СУБД. Хранение данных нетрадиционных типов: звука, видеоклипов графики.	1,7,9/3,4	Комплект электронных материалов
1.	**Общая характеристика СУБД Access** Справочная система. Виды справки и пользование ею. Инструментальные средства для создания БД и её объектов, для выполнения расчётов.	1,2,7,9/3	Реферат
1.	**Формирование запросов в СУБД** Создание запроса на основе нескольких таблиц. Технология создания запросов других типов.	1,7,8/3,4	Решения контрольных примеров
1.	**Проектирование отчётов** Технология проектирования отчёта с вычислениями в строках, с частными и с общими итогами. Составная форма (отчёт) и технология их проектирования. Оформление формы и отчёта.	1,7,8/3,4	Решения контрольных примеров

№ п/п	Содержание вопроса	Литература основная/ дополнительная	Формы контроля
1.	**Введение в SQL** Работа SQL со множеством пользователей. Транзакции и управление ими. Использование SQL с другими языками программирования.	1,2,8,9/3	Решения контрольных примеров
1.	**Управление базой данных** Актуальность защиты БД. Методы защиты: защита с помощью пароля, защита на уровне пользователя.	1,7,9/3	Презентация

3 Раздел контроля знаний

В разделе контроля знаний представлены:.

- Материалы текущей аттестации (вопросы, тест)
- Тематика контрольных работ (пример задания по контрольной работе,

  методические рекомендации по выполнению контрольной работы - Репозиторий БНТУ:

  http://rep.bntu.by/handle/data/24104

3.1 Материалы текущей аттестации

Вопросы для текущей аттестации:

- Приведите примеры множеств.
- Как можно задать множество (привести примеры и объяснить).
- Приведите примеры объединения множеств.
- Приведите примеры пересечения множеств.
- Приведите примеры разности множеств.
- Приведите примеры симметрической разности множеств.
- Что такое универсальное множество.

- Приведите примеры дополнения множества.
- Перечислите основные законы алгебры логики.
- Что такое отношения, совместимые по типу.
- Перечислите операторы переименования атрибутов.
- Что такое теоретико-множественные операторы.
- Системы обработки информации, их классификация.
- Общее представление о БД.
- Ориентировочные этапы проектирования БД.
- Информационно-логическая модель БД.
- Требования к СУБД.
- СУБД Access. Экранный интерфейс Access. Создание таблиц БД.
- Использование объектов OLE, диаграмм и специальных объектов в программных пакетах Access.
- Управление файлами в Access.
- Справочная системы в программном пакете Access.
- Ввод, модификация и удаление данных в СУБД Access.
- Организация запросов.
- Создание запросов на выборку.
- Элементы управления и их свойства.
- Создание и изменение форм ввода данных.
- Создание и изменение отчётов.
- Работа с внешними данными в СУБД Access.
- Создание макросов.
- Связывание и встраивание объектов.

Тест

(Задания с выбором одного правильного ответа, задания открытой формы, задания на установление соответствия)

1. Реляционная модель базы данных

1) использует некоторые положения релятивистской теории относительности Эйнштейна для организации данных в базе
2) использует организацию связей между таблицами, информация в которых сгруппирована в виде блоков записей (строк).
3) вобрала в себя всё лучшее имевшееся в иерархической и сетевой моделях и представляет сложный конгломерат обеих моделей

2. База данных

1)поименованная и организованная (структурированная) совокупность взаимосвязанных данных, которые отражают состояние объектов конкретной предметной области
2)последовательно организованный набор данных
3) информация о конкретной предметной области

3. Основные модели представления данных используемые при построении баз данных – это _____, _____, _____._____

4. Перечислите этапы проектирования базы данных – 1)_____, 2)_____, 3)_____.

5. Установите соответствие между термином и его определением

Понятие	Расшифровка понятия
1. база данных	A) неструктурированный поток данных
2. таблица	B) организованная структура, предназначенная для хранения информации.
3. неформатированный текст	C) структурированные данные в виде ячеек

Ответ: 1__, 2__, 3__

6. Понятие концепции клиент/сервер

1)название сетевой топологии, когда к мощной машине подключено много клиентов предполагает разделённую обработку информации
2)значительная часть информации обрабатывается на мощном сервере незначительная часть на клиентской машине
3)тип программного обеспечения загруженного на сервере

7. Система управления базами данных (СУБД)

1) программа, конвертирующая информацию БД в доступный и понятный пользователю текстовый формат
2) программный комплекс, обеспечивающий функционирование базы данных и отвечающий за сохранность, безопасность, целостность, взаимное соответствие данных и обеспечивает доступ пользователей к этим данным.

3) программа встраиваемая в операционную систему для работы с очень большими файлами

8. Приведение БД к нормальной форме

1) осуществляется для приведения продаваемой СУБД коммерческой фирмой к законодательной базе страны покупателя
2) устраняет избыточность данных и способствует повышению производительности системы
3) улучшает интерфейс отчётов по требованию заказчика

9. Информационно логическая модель данных

1) служит навигатором для пользователя при осуществлении запросов
2) показывает структуру связей между таблицами реляционной модели
3) устраняет избыточность данных в больших таблицах

10. Ключевое поле

1) создаётся для криптографической защиты доступа к данным базы
2) Одно или несколько полей, значения которых однозначно идентифицируют Запись
3) обязательный уникальный и единственный элемент для опознания БД в системе стандартов

11. Индексное поле

1) индексирует и сохраняет настройки просмотра таблиц с момента вашего последнего посещения
2) является полем, значение которого СУБД при выполнении запроса просматривает в первую очередь
3) хранит встроенные указатели для записей сохраняющие отношения предок/потомок

12. Между таблицами реляционной базы данных существуют отношения _____, _____, _____.

13. Установите соответствие между основными понятиями реляционной модели

Формальный реляционный термин	Неформальный практический эквивалент
1) отношение	A). столбец или поле
2) кортеж	B). количество строк
3) кардинальность	C). количество столбцов
4) атрибут	D). таблица
5) степень	E). уникальный идентификатор
6) первичный ключ	F). совокупность допустимых в поле значений
7) домен	G). строка или запись

Ответ: 1__, 2__, 3__,4__, 5__, 6__, 7__.

14. Окно "Схема данных" используется

1) для редактирования введенных данных
2) для создания связей между таблицами
3) для создания индексированных полей

15. При удалении таблицы из "Схемы данных"

1) таблица удаляется из базы данных
2) удаляются связи этой таблицы с другими таблицами, а сама таблица остаётся в схеме данных
3) таблица сохраняется в базе данных, но удаляется из схемы данных

16. Отношение один ко многим

1) это когда запись одной таблицы продолжается сразу в нескольких других таблицах
2) это запись одной таблицы связана сразу с несколькими записями другой таблицы
3) составная запись сложного вида, фрагменты которой расположены по нескольким таблицам

17. Отношение один к одному

1) когда запись одной таблицы связана только с одной записью из другой таблицы
2) когда число записей в двух таблицах одинаково
3) когда в таблице только одно ключевое поле и одно индексное

18. Реляционные базы данных

1) содержат данные только символьного формата
2) обрабатывают информацию только числового формата
3) работают со многими форматами данных
4) работают со смешанным форматом, содержащим символы и числа

19. Вы видите таблицу из реляционной базы данных. Запись образует

1) поле в таблице
2) строку в таблице
3) имя поля
4) значение в ячейке

20. Тип данных "Поле MEMO"назначают полям, содержащим

1) текст размером до 255 символов
2) текст размером более 255 символов
3) результаты вычислений
4) форматированный текст

21. Перечислите объекты базы данных –_____ _____ _____._____._____._____._____

22. Запрос БД это

1) средство выборки необходимых данных из одной или нескольких таблиц БД
2) операция тестирования целостности таблиц
3) установление отношения один к одному между двумя таблицами

23. Параметрический запрос используются для

1) наложения на поле постоянных условий отбора
2) наложения на поле разных условий отбора при каждом исполнения запроса
4) задания свойств запроса

24. Таблицы предназначены

1)предназначена для ввода и просмотра данных
2)средство выборки необходимых данных
3)предназначены для хранения информации об объектах предметной области

25. Форма ввода данных

1) устанавливает шрифт, абзацный отступ и прочие элементы форматирования текста для их удобного размещения в таблицу
2) предназначена для ввода, просмотра и корректировки данных
3) осуществляет предварительную подготовку таблицы (проводит её форматирование). для последующей записи большого массива данных

26. Главная и подчинённая формы в СУБД Access

1) служат для ввода основных данных (главная форма). и второстепенных данных (подчинённая форма).
2) обеспечивают удобное редактирование составных записей таблиц, связанных отношением один ко многим
3) главная форма связана с основным ключевым полем, а подчинённая с остальными полями

27. Сколько подчинённых форм в СУБД Access может содержать главная

1) одну
2) две
3) много

28. Главная кнопочная форма

1) используется для создания всех остальных экранных форм базы данных
2) является элементом интерфейса, предоставляющего удобный доступ к объектам разработанного приложения
3) позволяет вводить данные в таблицы базы путём манипуляции с кнопками

29. Отчёт в базе данных

1) пишут для систематизации таблиц
2) генерируется на основе созданного запроса (или таблицы) и предназначен для вывода информации на печать с целью дальнейшего анализа и принятия решения
3) это log файл периодически генерируемый СУБД с перечнем предупреждений и ошибок

3.2 Тематика контрольных работ

Пример задания по контрольной работе.
Методические рекомендации по выполнению контрольной работы - Репозиторий БНТУ: http://rep.bntu.by/handle/data/24104

З А Д А Н И Е

к контрольной работе по дисциплине

«Модели данных и системы управления базами данных »

Вариант 1

Слушатель:	Иванов Иван Иванович	Группа 203XXX

1. Тема Проектирование базы данных в MS Access

2. Срок сдачи слушателем контрольной работы: XX.XX.20XX

3 Исходные данные для работы

3.1 Новиков, Ф.А. Дискретная математика для программистов : учебник для вузов / Ф.А. Новиков. – 2-е изд. – СПб. : Питер, 2004. – 300 с.

3.2 Бекаревич Ю. Microsoft Access 2013: Самоучитель / Бекаревич Ю.- СПб: БХВ-Петербург, 2016 – 452 с.

4 Содержание контрольной работы (список вопросов, которые подлежат разработке)

4.1 Объект предметной области:

Автопарк осуществляет обслуживание заказов на перевозку грузов, используя для этих целей свой парк автомашин и своих водителей. Водитель, выполнивший заказ, получает 25% от стоимости перевозки.

Спроектируйте (анализ данных, выбор объектов, нормализация, определение связей между таблицами) и реализуйте средствами MS Access базу данных обеспечивающую для Автопарка:

4.1.1. Хранение сведений:

- о водителях - табельный номер, ФИО, стаж, категория, адрес, год рождения, принят, уволен.
- о заказах – дата, код заказа, табельный номер водителя, номер машины, километраж, стоимость.
- об автомашинах – номер машины, марка, пробег на момент покупки, о списании/ремонте/на ходу.

4.1.2. Выдачу справок в виде запросов:

- По указанному водителю – количество и перечень выполненных заказов.
- По указанной марке автомобиля – список номеров автомашин.
- Подсчет общей стоимости заказов.
- Информация о количестве выполненных заказов на каждом автомобиле.
- Информация о выполненных водителем заказах (цена заказов, зарплата

водителя).

4.1.3. Возможность обновления с помощью форм хранящейся информации:

- Оформление личных дел сотрудников.
- Занесение в банк данных информации о новом автомобиле либо изменение сведений о старом.
- Оформление заказа.

4.1.4. Выдачу справок в виде отчетов:

- по автомашинам, сгруппированный по номеру машин, – ФИО и табельные номера водителей, даты выполненных заказов, в примечании группы – итоговый километраж по каждой автомашине.
- по водителям, сгруппированный по ФИО, – дата заказа, номер автомобиля, километраж, итоговая сумма заработка водителя и в конце отчета итоговая сумма заработка всех водителей.

Создайте кнопочную форму для базы данных, которая загружается при открытии базы данных и с помощью которой можно просмотреть формы, отчеты и запустить на выполнение запросы.

Требования к контрольной работе

Контрольная работа должна быть представлена в двух видах:

1. Расчетно-пояснительная записка, которая представляется на листах белой бумаги формата А4.
2. Электронная версия работы представляется в виде файла формата *.doc, имя файла – фамилия слушателя, а также файла базы данных формата *.mdb.

---

4 Вспомогательный раздел

Во вспомогательном разделе представлено:

- Учебно-методическая документация
- Информационно-методические и аналитические материалы
- Мультимедийные ресурсы

4.1 Учебно-методическая документация

В разделе учебно-методической документации представлены:

- Методические рекомендации по организации учебного процесса
- Методические рекомендации по изучению дисциплины
- Перечень учебных изданий и информационно-аналитических материалов
- Примеры заданий и примеры выполнения контрольных, курсовых и выпускных работ
- Технология коммуникации участников образовательного процесса

4.1.1 Методические рекомендации по организации учебного процесса

Цели

- Изучение основ теории множеств, математической логики, системы и приемы, применяемые при решении практических задач и задач в области программирования. Изучение целостности реляционных данных и языка доступа к реляционным данным – реляционной алгебре.
- Формирование у обучаемых концептуальных представлений об основных принципах построения баз данных (БД), моделях данных, систем управления базами данных (СУБД); принципах проектирования БД; а также анализ основных технологий реализации БД.

Задачи

- изучить основы теории множеств, математической логики и реляционной алгебры;
- овладеть приемами проектирования баз данных;
- получить практические навыки проектирования и разработки баз данных для решения прикладных задач.

Методы обучения

- лекции с процедурой «пауз»;
- работа с текстами и кодами программ;
- организация дискуссий;
- работа в малых группах;
- методы профессионально-педагогической рефлексии.

Средства обучения

- персональные компьютеры и программное обеспечение;
- презентации к занятиям;

- интерактивная доска;
- раздаточный материал в твердой копии (методические пособия, тексты заданий);
- раздаточный материал в электронном виде (электронные документы в различных форматах, гиперссылки на источники в сети Internet, компьютерные программы и их коды, аудио и видео записи учебных занятий или ссылки на них).

4.1.2 Методические рекомендации по изучению дисциплины

Слушатель, изучивший дисциплину должен знать:

- основные определения теории множеств и математической логики;
- основы реляционной алгебры;
- основы файловой организации данных;
- основные положения теории баз данных и их проектирования;
- назначение, архитектуру, функциональные возможности и тенденции развития современных СУБД;
- принципы организации СУБД;
- структуру иерархической, сетевой, реляционной, объектно-ориентированной модели данных;
- концепцию клиент-серверной системы и объектно-ориентированных баз данных;
- технологии разработки баз данных для Web;
- объекты БД, их назначение;
- функции администратора БД;

Слушатель, изучивший дисциплину должен уметь:

- использовать основы теории множеств и математической логики при разработке баз данных, что обеспечит математическую строгость реляционной модели данных;
- создавать базы данных и проектировать их объекты: таблицы, запросы, формы, отчёты, макросы в среде СУБД;
- использовать язык структурированных запросов(SQL) для создания объектов БД;
- связывание и встраивание объектов;
- оптимизировать работу БД.

Слушатель, изучивший дисциплину должен владеть:

- основами теории множеств и математической логики;
- основами реляционной алгебры;
- технологией создания базы данных и проектирования объектов:

таблиц, запросов, форм, отчётов, макросов в среде СУБД;
- языком структурированных запросов(SQL);
- технологией создания резервных копий БД, восстановления БД;
- методами защиты БД;
- администрированием БД.

4.1.3 Перечень учебных изданий и информационно-аналитических материалов

Список рекомендуемой литературы

Список использованных источников

Основной

1. Бекаревич Ю. Microsoft Access 2013: Самоучитель / Бекаревич Ю.- СПб: БХВ-Петербург, 2016 – 452 с.
2. Кошелев В.Е. Базы данных в ACCESS 2007/ Кошелев В.Е. - М.: Бином, 2015. - 592с.
3. Гаврилов М.В. Информатика и информационные технологии: Учебник для бакалавров / М.В. Гаврилов, В.А. Климов; Рецензент Л.В. Кальянов, Н.М. Рыскин. - М.: Юрайт, 2013. - 378 с.
4. Исаев Г.Н. Информационные технологии: Учебное пособие / Г.Н. Исаев. - М.: Омега-Л, 2013. - 464 с.
5. Информатика и информационные технологии: учебное пособие / Ю.Д. Романова, П.А. Музычкин и др.; под ред. Ю.Д. Романова.- 4-е изд., перераб. и доп.- М.: Эксмо, 2010.-688с.
6. Бекаревич Ю.: Самоучитель Access 2010 (+ CD-ROM) / Юрий Бекаревич , Нина Пушкина. - СПб: БХВ-Петербург, 2011. - 432 с.
7. Сурядный А. С. Microsoft Access 2010: Лучший самоучитель / А.С. Сурядный. - М.: Астрель, ВКТ, 2012. - 448 с.
8. Гроff Дж., Вайнберг П. SQL Полное руководство . – М.: Вильямс, 2015. – 959 с.

Дополнительный

9. Гроff, Дж. SQL: Полное руководство / Дж. Гроff, П. Вайнберг ; пер. с англ. – 2-е изд., перераб. и доп. – К. : Издательская группа BHV, 2001. – 816 с.
10. Гурвиц, Г. Microsoft Access 2010. Разработка приложений на реальном примере / Г. Гурвиц. - М.: БХВ-Петербург, 2010. - 496 с.
11. Молчина, Л.И. Методические рекомендации по выполнению контрольной работы по дисциплине «Модели данных и системы

управления базами данных». Репозиторий: http://rep.bntu.by/handle/data/24104

5 Программный блок

В разделе "программный блок" представлена учебно-программная документация – учебная программа с изложением её основных разделов и содержания

5.1 Учебная программа

1. Элементы теории множеств

Множества. Операции над множествами Декартово произведение множеств. Отношения. Бинарные отношения. Отношение эквивалентности. Отношения порядка. Функциональное отношение. N-арные отношения.

2. Базовые понятия математической логики в концепции баз данных

Язык логики высказываний. Операции над высказываниями. Формулы логики высказываний. Таблицы истинности.

3. Основные понятия реляционной алгебры

Замкнутость реляционной алгебры. Отношения и соответствия. Теоретико-множественные операции над отношениями. Отношения и домены. Домены и атрибуты. Ключи. Реляционная алгебра как система запросов. Нормализация отношений. Вторая нормальная форма. Третья нормальная форма.

4. Автоматизация обработки информации

Автоматизированная информационная система (АИС). Информационное обеспечение АИС. Понятие базы данных.
Трехуровневая модель организации баз данных. Иерархические и сетевые модели данных. Реляционная модель данных. Постреляционная, многомерная, объектно-ориентированная и объектно-реляционная модели данных.

5. Проектирование БД. Организация связей между объектами БД

Понятие процесса проектирования. Этапы проектирования реляционной БД. Словарь данных. Концептуальная модель. Логическое проектирование. Логические связи между отношениями. Типы логических связей. Первичный ключ. Вторичные ключи. Достоинства и недостатки реляционной модели данных. Развитие моделей организации данных в БД.

6. Этапы проектирования реляционной БД

Нормализация отношений. Первая нормальная форма. Создание логической модели БД.

7. Функциональные возможности СУБД

Операции над отношениями в реляционной СУБД. Режимы работы пользователя с СУБД. Функциональные возможности современных СУБД. Обеспечение работы с локальными и распределёнными БД.

8. Среда СУБД MS Access. Проектирование таблиц и работа с ними

Общая характеристика СУБД Access. Объекты БД, их модель в СУБД Access. Пользовательский интерфейс. Настройка рабочей среды. Файлы БД, расширение файлов БД. Типы данных, обрабатываемых СУБД. Таблица как объект БД: поля, записи, ключ. Инструментальные средства для создания таблицы. Описание структуры таблицы. Установка ключа. Заполнение и сохранение таблицы.

9. Записи в СУБД MS Access

Создание связи определённого типа между таблицами. Схема БД. Модификация структуры таблицы. Операции, выполняемые при работе с таблицей. Работа с таблицей в режиме заполнения. Сортировка и фильтрация данных. Средства ускорения доступа к данным.

10. Организация запросов в СУБД MS Access

Запросы, их типы и возможности. Способы создания запроса. Инструментальные средства для формирования запроса.

11. Задание условий отбора в запросах

Создание запроса выбора. Задание условий отбора. Вычисления над полями БД и итоговые вычисления. Выполнение и сохранение запроса.

Групповые операции в запросах. Перекрестные запросы. Запросы действия и параметрические запросы.

12. Проектирование форм

Интерфейс БД. Назначение формы. Инструментальные средства для проектирования формы, отчёта. Использование формы для работы с БД. Технология проектирования формы с различными элементами. Автоматическое создание форм на основе таблицы или запроса
Проектирование форм с помощью Мастера. Проектирование форм в режиме Конструктора: структура формы, элементы управления в формах, панель элементов, объекты других приложений в форме, главные и подчиненные формы, представления форм. Редактирование записей в формах
Создание кнопочных форм. Автоотчеты.

13. Проектирование отчетов

Назначение отчёта. Проектирование отчетов с помощью Мастера. Проектирование отчетов в режиме Конструктора: структура отчета, изменение вида отчета (нумерация страниц, колонтитулы, сортировка и группировка данных, объекты других приложений). Главные и подчиненные отчеты. Обработка отчетов (сохранение, открытие, модификация, просмотр, печать).

14. Многопользовательские технологии баз данных

Архитектуры файл-сервер и клиент - сервер. СУБД, используемые в архитектуре клиент - сервер. Создание пользовательского источника данных.

15. Организация запросов в языке SQL

Язык SQL – стандарт ANSI/ISO. Структура команды языка SQL. Описание данных (команда CREATE TABLE). Манипулирование данными: ввод, удаление и изменение значений полей (команды INSERT, DELETE, UPDATE).
Формирование запросов (команда SELECT). Использование реляционных, булевых и специальных операторов в условиях. Соединение таблиц.

16. Программирование в среде MS Access

Встроенный язык программирования VBA. Основные типы переменных.

Соглашения по идентификации. Простейшие операторы языка. Создание и выполнение макросов.

17. Объектно-ориентированные базы данных

Общие понятия объектно-ориентированного подхода и их реализация в ООБД. Языки программирования систем ООБД и языки запросов.
Объектно-ориентированные СУБД.

18. Администрирование баз данных

Уровень доступа к данным в базе. Пользователи БД. Администратор БД, его функции. Резервирование БД. Понятие репликации БД. Оптимизация работы БД. Восстановление БД.

Список использованных источников

Основной

1. Бекаревич Ю. Microsoft Access 2013: Самоучитель / Бекаревич Ю.- СПб: БХВ-Петербург, 2016 – 452 с.
2. Кошелев В.Е. Базы данных в ACCESS 2007/ Кошелев В.Е. - М.: Бином, 2015. - 592с.
3. Гаврилов М.В. Информатика и информационные технологии: Учебник для бакалавров / М.В. Гаврилов, В.А. Климов; Рецензент Л.В. Кальянов, Н.М. Рыскин. - М.: Юрайт, 2013. - 378 с.
4. Исаев Г.Н. Информационные технологии: Учебное пособие / Г.Н. Исаев. - М.: Омега-Л, 2013. - 464 с.
5. Информатика и информационные технологии: учебное пособие / Ю.Д. Романова, П.А. Музычкин и др.; под ред. Ю.Д. Романова.- 4-е изд., перераб. и доп.- М.: Эксмо, 2010.-688с.
6. Бекаревич Ю.: Самоучитель Access 2010 (+ CD-ROM) / Юрий Бекаревич , Нина Пушкина. - СПб: БХВ-Петербург, 2011. - 432 с.
7. Сурядный А. С. Microsoft Access 2010: Лучший самоучитель / А.С. Сурядный. - М.: Астрель, ВКТ, 2012. - 448 с.
8. Гроff Дж., Вайнберг П. SQL Полное руководство . – М.: Вильямс, 2015. – 959 с.

Дополнительный

9. Гроff, Дж. SQL: Полное руководство / Дж. Гроff, П. Вайнберг ; пер. с англ. – 2-е изд., перераб. и доп. – К. : Издательская группа

BHV, 2001. – 816 с.

10. Гурвиц, Г. Microsoft Access 2010. Разработка приложений на реальном примере / Г. Гурвиц. - М.: БХВ-Петербург, 2010. - 496 с.
11. Молчина, Л.И. Методические рекомендации по выполнению контрольной работы по дисциплине «Модели данных и системы управления базами данных». Репозиторий: http://rep.bntu.by/handle/data/24104