Министерство образования и науки Российской Федерации
Волгоградский государственный архитектурно-строительный университет

ИНФОРМАТИКА

Методические указания к контрольной работе

Составитель М. М. Юшкова

Волгоград
ВолгГАСУ
2014

УДК 004.5(076.5)
ББК 32-81я73
И741

И741 **Информатика** [Электронный ресурс] : методические указания к контрольной работе / М-во образования и науки Рос. Федерации, Волгогр. гос. архит.-строит. ун-т ; сост. М. М. Юшкова. — Электронные текстовые и графические данные (359 Кбайт). — Волгоград : ВолгГАСУ, 2014. — Систем. требования: PC 486 DX-33; Microsoft Windows XP; Internet Explorer 6.0; Adobe Reader 6.0. — Официальный сайт Волгоградского государственного архитектурно-строительного университета. Режим доступа: http://www.vgasu.ru/publishing/on-line/ — Загл. с титул. экрана.

Приведены два раздела. В первом даны определения, свойства, типы алгоритмов, примеры выполнения заданий с решением в системе Mathcad. Во втором разделе рассмотрены имена сущностей, атрибутов сущностей, определение типов баз данных, правила построения концептуальной схемы и пример проектирования реляционных баз данных с бинарными и триарными связями, реализованных в среде Access 2000. Изложена методика выполнения контрольной работы по дисциплине «Информатика».

Для студентов, обучающихся по профилю «Промышленное и гражданское строительство».

УДК 004(076.5)
ББК 32-81я73

ОГЛАВЛЕНИЕ:

Методические указания предназначено для выполнения контрольной работы студентов с использованием навыкам программирования в математической системе Mathcad, алгоритмизации и проектирование реляционной базы данных, позволяющим решать большой спектр практических задач. В процессе выполнения контрольной работы студенты осваивают средства создания пользовательского интерфейса приложений, а также средства ввода, редактирования и отладки программ.

В настоящее время наблюдается стремительный процесс компьютеризации общества, выражающийся как в многократном увеличении парка персональных компьютеров, так и во внедрении новых информационных технологий. По некоторым оценкам, 80% создаваемых и уже эксплуатируемых приложений предназначены для работы с данными. В связи с этим возникает задача массового обучения пользователей персональными компьютерами, владеющими технологиями баз данных. Современные системы управления базами данных (СУБД) такие как Access, Paradox и др. обладают исключительно "дружественным" интерфейсом, предоставляющим широкому кругу пользователей (не программистов) мощные интерактивные средства работы с реляционными базами данных. Работая в среде упомянутых выше СУБД, можно создавать достаточно персонифицированные приложения, не написав ни одной строчки программного кода, используя реализованный в них принцип объектно-ориентированного программирования. Обучение работе с этими СУБД студентов практически всех специальностей не составляет проблемы. Главная трудность состоит в обучении проектированию баз данных.

Содержимое реальной базы данных, динамически изменяется, поскольку данные могут быть добавлены, удалены или модифицированы в течении жизненного цикла базы данных. Поэтому, если данные структурированы неправильно, то такая база данных по истечении некоторого промежутка времени будет непригодна к использованию, так как неизбежно появится избыточная и устаревшая информация и утеряна

необходимая информация. Чтобы избежать этого, база данных должна быть правильно спроектирована.

Лабораторная работа № 1

1. АЛГОРИТМИЗАЦИЯ

1.1. Понятие алгоритма, его свойства и способы описания

Алгоритм — это понятное и точное предписание исполнителю совершить заданную последовательность действий, направленных на достижение поставленной цели. Понятие алгоритма также определяется как формально описанная процедура преобразования входных данных, в выходные данные, представляющие собой искомый результат.

Информационный процесс с известным начальным состоянием, конечным состоянием, исполнителем и последовательности операций из системы команд исполнителя, называется алгоритмическим процессом.

Принципиальным в этих определениях является формальный характер действий исполнителя, который может не понимать смысла того, что он делает, но обязательно получит необходимый результат, если точно выполнит все предписания, указанные в алгоритме. Примерами формального исполнителя алгоритма можно считать ЭВМ, а также кодовый замок.

Разработанные алгоритмы обладают рядом **свойств**, основными из которых являются:

Дискретность (конечность). Дискретность обуславливает дискретный (пошаговый) характер процесса получения результата, состоящий в последовательном выполнении конечного числа заданных алгоритмом действий. Только закончив выполнение одного действия, можно приступит к выполнению следующего.

Понятность. Понятность означает, что алгоритм не должен содержать ни одного предписания, которое исполнитель не может выполнить. В алгоритме должны использоваться только те предписания, которые имеются в системе команд исполнителя.

Детерминированность (определенность). Детерминированность определяет однозначность получаемого результата при одних и тех же исходных данных. Для обеспечения этого алгоритм не должен содержать предписаний, смысл которых может восприниматься исполнителем неоднозначно, или ему неясно какое следующее предписание он должен выполнять.

Результативность. Результативность означает обязательность получения результата после выполнения над исходными данными заданной последовательности действий или сообщения о невозможности решения.

Массовость. Массовость означает, что алгоритм должен обеспечивать решение не одной конкретной задачи, а некоторого класса задач данного типа. Массовость обеспечивает возможность использования различных исходных данных.

Существует много способов описания (представления) алгоритма. Существенными требованиями к способу представления алгоритма является понятность, однозначность и наглядность структуры. Наибольшее распространение получили словесный, графический, а также описание на алгоритмическом языке, который в литературе также называют языком псеводокода.

Словесный способ описания алгоритма

При словесном способе записи алгоритм представляет собой обычную математическую запись выражений, зависимостей необходимых пояснений к ним и указаний последовательности выполняемых действий.

Графический способ описания алгоритма

Наиболее наглядным способом записи алгоритма является изображение в виде последовательностей блоков, каждый из которых предписывает выполнение определенных действий.

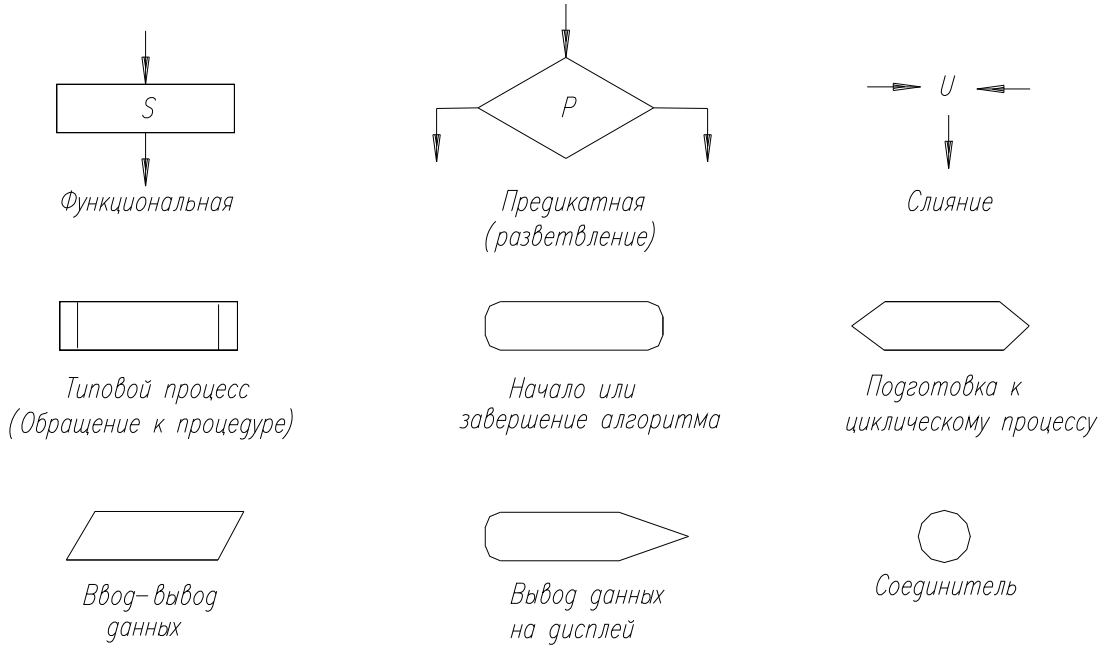


Рис. 1

При графическом способе задания алгоритм представляется в виде блок-схем, которые состоят из символов (см. рис.1) и линий, связывающих эти символы.

Этот способ обладает наглядностью структуры, обеспечивает высокую «читаемость» алгоритма и явное отображение управления в нем. Поэтому блок-схема является естественным представлением алгоритма. Блок-схема — это ориентированный граф, указывающий порядок исполнения команд алгоритма.

Блок-схемы алгоритмов должны выполняться в соответствии с государственными стандартами, изложенными в ГОСТ 19.002-80. "Схемы алгоритмов и программ. Правила выполнения" и ГОСТ 19.003-80 "Схемы алгоритмов и программ. Обозначения условные и графические".

Линии, связывающие символы, называют линиями потока. Они должны проводиться по вертикали и горизонтали и подводиться к середине символа. Нормальным направлением линий потока считается направление сверху вниз и слева направо; стрелками эти направления могут не обозначаться. Изображенная на рис. 1 вершина слияния на блок-схемах обозначается как пересечение линий потока.

Графический способ компактен, формализован, имеет хорошую наглядность используемых структур и допускает различные степени детализации алгоритма. Недостаток — плохо выражено свойство дискретности алгоритма, то есть трудно понять пошаговый характер получения результата (какое конкретное действие за каким следует).

1.2. Способ описания алгоритма на алгоритмическом языке

Достаточно распространенным способом представления алгоритма является его запись на алгоритмическом языке, представляющем в общем случае систему обозначений и правил для единообразной и точной записи алгоритмов и исполнения их. Отметим, что между понятиями "алгоритмический язык" и "язык программирования" есть различие; прежде всего, под исполнителем в алгоритмическом языке может подразумеваться не только компьютер, но и любое другое устройство. Программа, записанная на алгоритмическом языке, не обязательно предназначена компьютеру. На алгоритмическом языке, осуществляется пошаговая запись алгоритма, которая называется программой на псевдокоде или псевдокодом. Программа на псевдокоде — это алгоритм, записанный на языке алгоритмическом языке. Как и каждый язык, алгоритмический язык имеет свой словарь. Основу этого словаря составляют слова, употребляемые для записи команд, входящих в систему команд исполнителя алгоритма. Кроме того, в алгоритмическом языке используются слова, смысл и способ употребления которых задан раз и навсегда. Эти слова называются служебными. Использование служебных слов делает запись алгоритма более наглядной, а форму представления различных алгоритмов — единообразной.

Рассмотрим пока самые общие правила написания программы на алгоритмическом языке.

Программу, написанную на алгоритмическом языке (на псевдокоде), нельзя непосредственно ввести в ЭВМ, однако ее легко переписать на любой другой процедурный язык программирования. Таким образом, программа для

ЭВМ — это алгоритм, записанный на языке программирования. Основу программы составляют инструкции, указывающие, какие операции следует осуществлять над данными. Элементами данных программы являются целые и вещественные числа, символы, булевские значения true и false.

Однако в программе, написанной на алгоритмическом языке, слабо выражена структура алгоритма. Для повышения наглядности структуры программы широко используются отступы, величина которых соответствует уровню вложенности (структурной иерархии). Далее, как правило, будут использоваться два способа описания алгоритма — и блок-схемы, и псевдокод.

1.3. Индивидуальные задания

Выберете индивидуальное задание из таблицы. Номер варианта выбирается по сумме двух последних цифр зачетной книжки. Цифра 0 в последних цифрах зачетной книжки берется за число 10.

№ варианта	№ задачи	№ варианта	№ задачи
1	1.11.21.31	11	1,12,24,33
2	2,12,22,32	12	2,13,25,35
3	3,13,23,33	13	3,14,26,36
4	4,14,24,34	14	4,15,23,37
5	5,15,25,35	15	5,16,27,38
6	6,16,26,36	16	6,17,28,39
7	7,17,27,37	17	7,18,29,40
8	8,18,28,38	18	8,19,30,33
9	9,19,29,39	19	9,20,24,35
10	10,20,30,40	20	10,12,25,36

Задача 1. Даны векторы x и y: $x^T = (9, 5, 8, 4)$, $y^T = (6, 3, 4, 7)$
Используя значения координат векторов x и y вычислить значения элементов матрицы P, показанные на рисунке.

$$P = \begin{pmatrix} 54 & 27 & 36 & 63 \\ 30 & 15 & 20 & 35 \\ 48 & 24 & 32 & 56 \\ 24 & 12 & 16 & 28 \end{pmatrix}$$

Задача 2. Дана матрица A.

$$A = \begin{pmatrix} 2 & 5 & 3 \\ 1 & 7 & 9 \\ 4 & 5 & 6 \end{pmatrix}$$

Переписать элементы матрицы A по строкам в вектор x:

$x^T = (2, 5, 3, 1, 7, 9, 4, 5, 6)$

Задача 3. Дан вектор x(9):

$x^T = (2, 4, 7, 9, 1, 3, 5, 4, 9)$

Переписать элементы вектора x в матрицу A по столбцам

$$A = \begin{pmatrix} 2 & 9 & 5 \\ 4 & 1 & 4 \\ 7 & 3 & 9 \end{pmatrix}$$

Задача 4. Дан вектор x. Переписать элементы вектора x в вектор y в обратном порядке, например:

$x^T = (5, 3, 4, 1, 4, 2, 6, 8, 9)$

$y^T = (9, 8, 6, 2, 4, 1, 4, 3, 5)$

Задача 5. Дана матрица A(3,3). Вычислить элементы вектора V(3), равные среднему арифметическому соответствующих строк матрицы A.

Задача 6. Даны векторы x и y. Вычислить элементы матрицы A по формуле

$A_{i,j} = x_i\, y_j$ и найти наибольший элемент матрицы A, а также его номер строки и номер столбца.

Задача 7. Вычислить сумму элементов квадратной матрицы B, расположенных на диагонали, параллельной главной и на один элемент выше главной.

Задача 8. Дана квадратная матрица A. Найти наибольший элемент главной диагонали матрицы A и вывести на печать всю строку, в которой он находится.

Задача 9. Дана квадратная матрица A. Вычислить сумму элементов главной и побочной диагоналей матрицы A.

Задача 10. Определить количество положительных *k* и отрицательных *c* элементов массива A(3,4). Переписать по строкам положительные элементы массива A в вектор P(k).

Задача 11. Дана матрица A(5,4). Вычислить количество положительных элементов последнего столбца матрицы A.

Задача 12. Вычислить сумму элементов каждой строки матрицы A(5,5). Переписать в вектор P1 положительные значения этих сумм, а номера строк, в которых находятся положительные суммы переписать в вектор F.

Задача 13. Дан массив B(4,4). Получить массив A(4,4), отличающийся от данного тем, что в каждой строке минимальный элемент и элемент на главной диагонали поменялись местами.

Задача 14. Дан массив A(3,5) и число C. Переписать элементы каждой строки массива A большие C в вектор B. Если в строке нет ни одного элемента большего C, то записать в вектор B один нуль.

Задача 15. Дана матрица A(5,5). Вычислить сумму и произведение элементов главной диагонали матрицы A.

Задача 16. Дана матрица A(4,4). Вычислить элементы матрицы B(4,4) по формуле $B_{i,j} = \begin{cases} A_{i,j}, & если\ A_{i,j} \geq 0 \\ 1, & если\ A_{i,j} < 0 \end{cases}$. Найти сумму матриц $C_{i,j} = A_{i,j} + B_{i,j}$. Найти элементы вектора F, равные элементам последнего столбца матрицы C.

Задача 17. Дана матрица A(4,5). Найти наибольший элемент матрицы A, а также номер строки и номер столбца, в котором он находится.

Задача 18. Дана матрица C(4,4). Элементы каждой строки матрицы C разделить на максимальный элемент этой строки.

Задача 19. Дана матрица C(4,5). В каждой строке матрицы C переставить местами максимальный и минимальный элементы.

Задача 20. Дана матрица A(4,3). Вычислить сумму положительных элементов каждой строки матрицы A, найти наибольшую из этих сумм, а также номер строки, в которой находится максимальная сумма.

Задача 21. Дана матрица A(4,4). Получить вектор F(4), состоящий из максимальных элементов строк матрицы A.

Задача 22. Дана матрица A(4,4). Найти столбец, сумма элементов которого минимальна. Вычислить значение минимальной суммы и определить номер столбца, в котором находится минимальная сумма.

Задача 23. Дана матрица A(4,4). Переписать в виде вектора положительные (отрицательные) элементы главной диагонали матицы A.

Задача 24. Дана матрица A(4,3). Найти разность между наибольшим и наименьшим элементами матрицы A.

Задача 25. Дана матрица A(4,5). Получить вектор X1(5), компоненты которого равны второй строке матрицы A, и вектор Y1(4), компоненты которого равны третьему столбцу матрицы A.

Задача 26. Дана матрица A(4,4). Определить количество положительных элементов каждого столбца матрицы A и переписать их в вектор.

Задача 27. Дана матрица A(4,4). Вычислить произведение элементов матрицы A, больших 1.

Задача 28. Дана матрица B(4,5). Вычислить количество элементов больших нуля в каждой строке матрицы B.

Задача 29. Дана матрица A(4,4). Переписать элементы главной диагонали матрицы A в вектор V(4).

Задача 30. Дана матрица A(4,4). В матрице A найти пять максимальных элементов.

Задача 31. Дана матрица A(4,4). Определить количество положительных и отрицательных элементов матрицы A.

Задача 32. Дана матрица A(4,4). Пронормировать матрицу A по ее максимальному элементу, т.е. каждый элемент матрицы A разделить на ее максимум. Вывести на печать полученную матрицу, а также ее максимум и номер строки и столбца, где он находится.

Задача 33. Дана матрица A(4,4). Получить матрицу B(4,4) каждый элемент которой равен соответствующему элементу матрицы A, умноженному на ее максимальный элемент.

Задача 34. Вычислить значение $S = \sum_{i=1}^{20} x_i$,

где x_i вычисляется по формуле

$$x_i = \begin{cases} 1, & если\ i = 1 \\ (i-1)\cdot i & если\ i \neq 1 \end{cases}$$

Задача 35. Вычислить значение $S = \sum_{i=1}^{20} (x_i + 1)^3$,

где $x_i = \begin{cases} i^2 - 1 & если\ i - четное \\ i^3 + 1 & если\ i - нечетное \end{cases}$

Задача 36. Дана матрица A(n, m). Упорядочить строки матрицы по убыванию сумм элементов строк.

Задача 37. Из матрицы A(N,N) сформировать одномерный массив B(N*N), обходя матрицу по спирали.

Задача 38. Отсортировать (упорядочить) значения элементов вектора X(9) по убыванию.

Задача 39. Определить среднее значение элементов массива X(5). Найти далее индекс массива, наиболее близкого к среднему значению.

Задача 40. Задан массив X(10). Сформировать два массива Y(5) и Z(5), включая в массив Y элементы с четными индексами, а в массив Z — элементы с нечетными индексами.

Задача 41. Заданы два одномерных массива различных размеров. Объединить их в один массив, включая второй массив между q-м и (q+1)-м элементами первого (q задано).

Задача 42. Дана квадратная матрица A(*n*, *n*). Вычислить сумму элементов, расположенных над ее главной диагональю.

Задача 43. Дана квадратная матрица A(*n*, *n*). Найти вектор V(n), каждая координата которого равна среднему арифметическому положительных элементов соответствующего столбца матрицы A.

Задача 44. Дан вектор X(n). Отсортировать вектор X по убыванию «Методом пузырька».

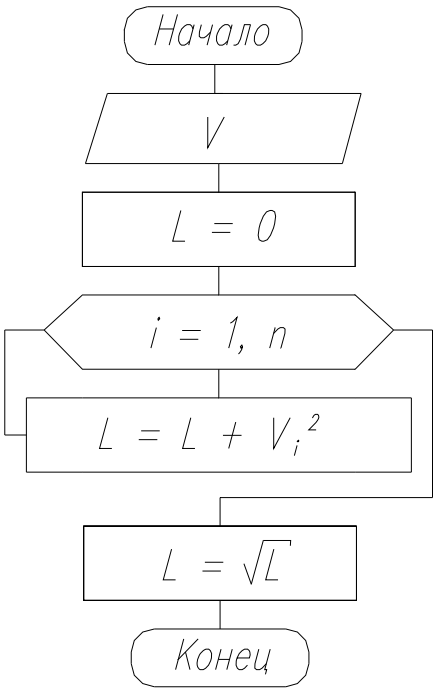
Задача 45. Дана квадратная матрица A(*n*, *n*). Вычислить сумму элементов, расположенных над ее побочной диагональю.

Задача 46. Дана квадратная матрица A(*n*, *n*). Получить вектор V(n - 1), координатами которого являются элементы матрицы A, расположенные на одну позицию выше ее побочной диагонали.

1.4.ПРИМЕРЫ ВЫПОЛНЕНИЯ КОНТРОЛЬНОЙ РАБОТЫ

Пример 1. Вычисление длины вектора

Алгоритм задачи в виде блок-схемы и в виде программы на алгоритми ческом языке на рис. 1.




```
ПРОГРАММА 3
НАЧАТЬ ПРОГРАММУ
ПИСАТЬ ("ВВЕДИТЕ ВЕКТОР V(n), n)
ЧИТАТЬ (V(n), n)
L = 0
      НЦ ДЛЯ i = 1 ДО n
          L = L + V_i^2
      КЦ
L = √L
КОНЕЦ ПРОГРАММЫ
```

Рис. 1

$$V := \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix}$$

ORIGIN := 1

n := 3

Проверка

L :=
L ← 0 |V| = 7.071
for i ∈ 1..n
L ← L + (V_i)^2
L L := √L L = 7.071

Рис. 2

На рис. 2 показано решение задачи в системе MathCAD в соответствии с алгоритмом рис. 1.

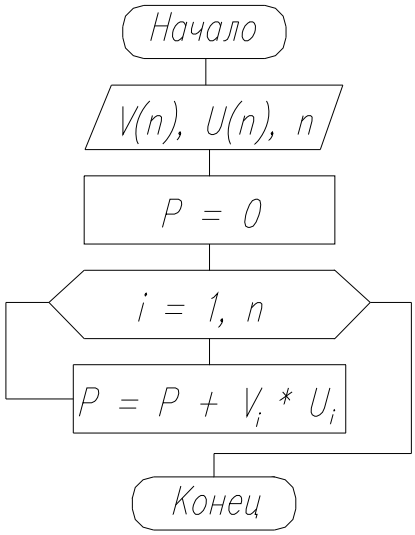
При каждом *i*-м заходе в тело цикла происходит сложение содержимого ячейки памяти *L* с величиной V_i^2 (квадратом *i*-й координаты вектора *V*). В результате этого в ячейке памяти *L* происходит накопление суммы

$$L = 0 + L_1^2 + L_2^2 + \ldots + L_n^2$$

После выхода из цикла из содержимого ячейки памяти (сумма квадратов координат вектора V) извлекается квадратный корень, и результат помещается снова в ячейку *L*.

Пример 2. Вычисление скалярного произведения двух векторов

Алгоритм задачи в виде блок-схемы и в виде программы на алгоритмическом языке на рис. 3.




```
ПРОГРАММА 3
  НАЧАТЬ ПРОГРАММУ
  ПИСАТЬ ("ВВЕДИТЕ ВЕКТОРЫ V(n), U(n), n)
  ЧИТАТЬ (V(n), U(n), n)
  P = 0
       НЦ ДЛЯ i = 1 ДО  n
           P = P + Vi * Ui
       КЦ
КОНЕЦ ПРОГРАММЫ
```

Рис. 3

$$U := \begin{pmatrix} 2 \\ 7 \\ 1 \end{pmatrix} \quad V := \begin{pmatrix} 3 \\ 2 \\ 5 \end{pmatrix}$$

ORIGIN := 1

n := 3

Проверка

U·V = 25

```
P := | P ← 0
     | for i ∈ 1..n
     |   P ← P + U_i·V_i
     | P
```

P = 25

Рис. 4

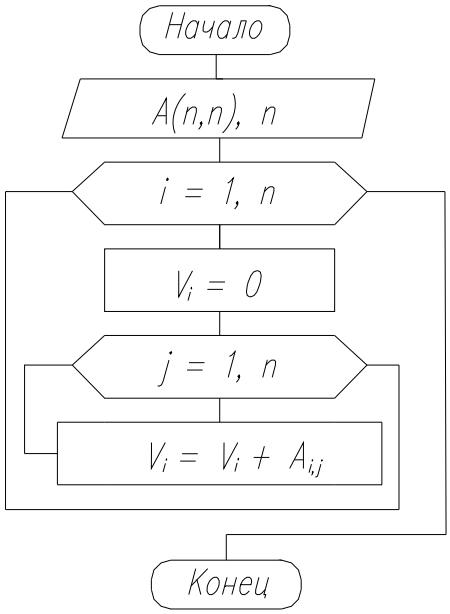
Вид документа MathCAD, содержащего ввод исходных данных и программный блок, соответствующий алгоритму, рис. 46, показан на рис. 4.

При каждом *i*-м заходе в тело цикла происходит сложение содержимого ячейки памяти *P* с величиной U_iV_i (произведением i-х координат векторов U и V). В результате этого в ячейке памяти *P* происходит накопление суммы

$$P = 0 + U_1 \cdot V_1 + U_2 \cdot V_2 + ... + U_n \cdot V_n$$

Пример 3. Вычислить координаты вектора, равные суммам элементов соответствующих строк данной матрицы

Алгоритм задачи в виде блок-схемы и в виде программы на алгоритмическом языке на рис. 5.




```
ПРОГРАММА 3
НАЧАТЬ ПРОГРАММУ
ПИСАТЬ ("ВВЕДИТЕ МАТРИЦУ A(n,n), n")
ЧИТАТЬ (A(n,n), n)
    НЦ ДЛЯ i = 1 ДО n
        Vi = 0
        НЦ ДЛЯ j = 1 ДО n
            Vi = Vi + Ai,j
        КЦ
    КЦ
КОНЕЦ ПРОГРАММЫ
```

Рис. 5

На рис. 6 показано решение задачи в системе MathCAD в соответствии с алгоритмом рис. 5

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 3 & 4 & 8 \end{pmatrix} \qquad ORIGIN := 1 \qquad n := 3$$

$$V := \left| \begin{array}{l} \text{for } i \in 1..n \\ \quad \left| \begin{array}{l} V_i \leftarrow 0 \\ \text{for } j \in 1..n \\ \quad V_i \leftarrow V_i + A_{i,j} \end{array} \right. \\ V \end{array} \right. \qquad V = \begin{pmatrix} 6 \\ 15 \\ 15 \end{pmatrix}$$

Рис. 6

Пример 4. Вычисление произведения двух матриц

Математический смысл произведения двух матриц показан на рис. 7

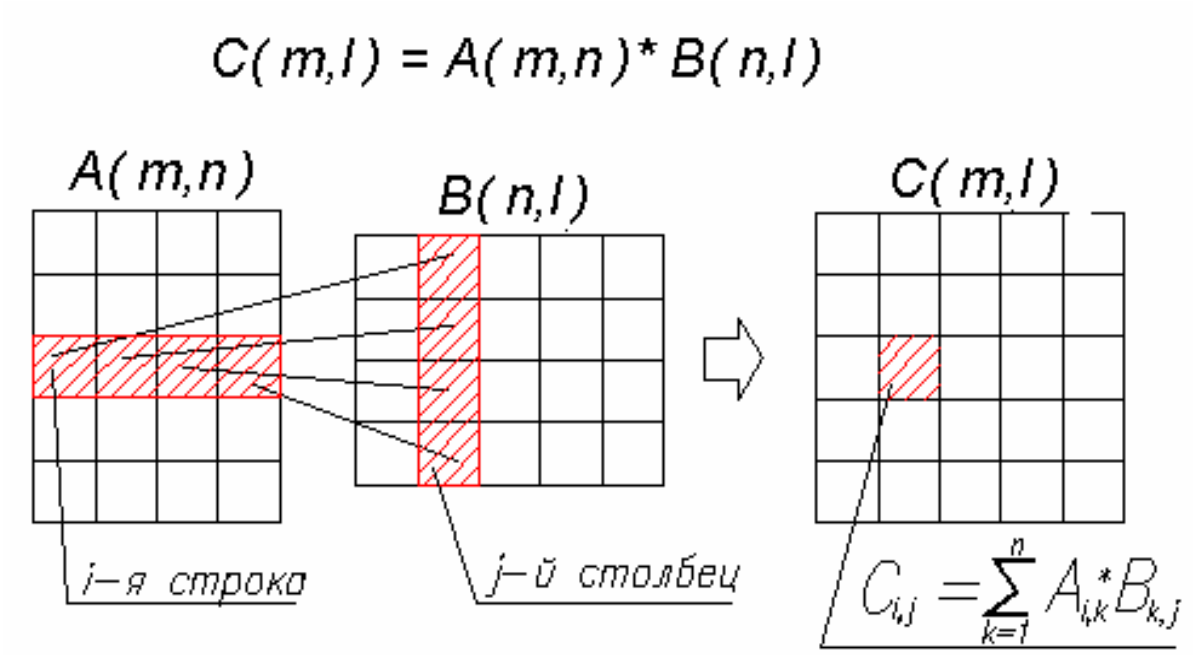
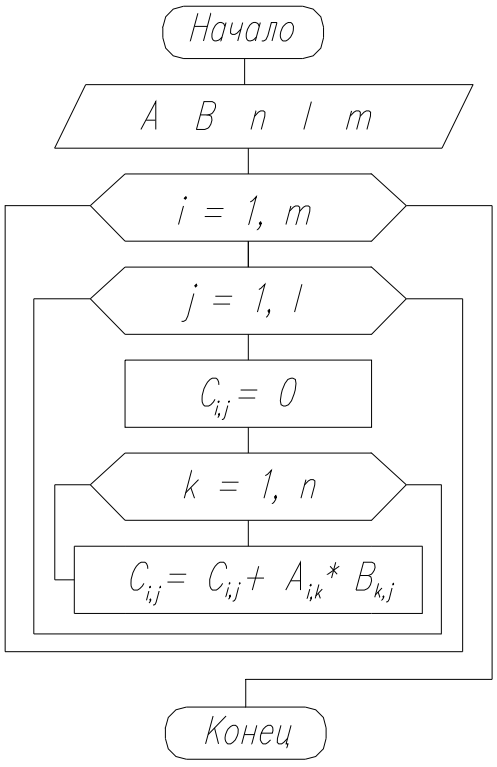


Рис. 7

Алгоритм задачи в виде блок-схемы и в виде программы на алгоритмическом языке на рис. 8



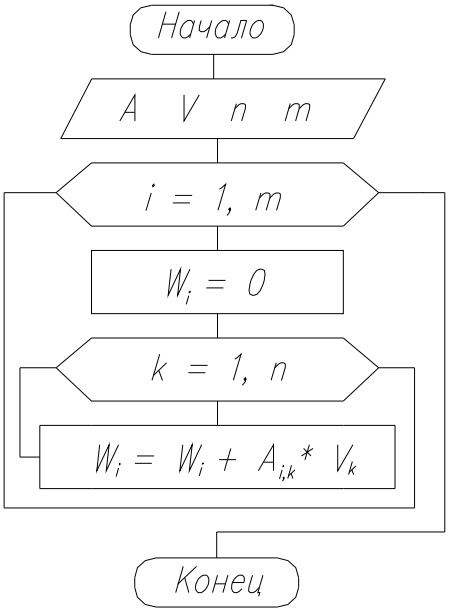

```
ПРОГРАММА 3
НАЧАТЬ ПРОГРАММУ
ПИСАТЬ ("ВВЕДИТЕ A, B, n, l, m")
 ЧИТАТЬ (A, B, n, l, m)
   НЦ ДЛЯ i = 1 ДО  m
       НЦ ДЛЯ j = 1 ДО  l
          C[i,j]= 0
          НЦ ДЛЯ k = 1 ДО  n
               C[i,j]= C[i,j]+ A[i,k]* B[k,j]
          КЦ

       КЦ
   КЦ
 КОНЕЦ ПРОГРАММЫ
```

Рис. 8

Пример 5. Вычисление произведения матрицы на вектор

Алгоритм задачи в виде блок-схемы и в виде программы на алгоритмическом языке на рис. 53.




```
ПРОГРАММА 3
НАЧАТЬ ПРОГРАММУ
ПИСАТЬ ("ВВЕДИТЕ A, V, n, m")
ЧИТАТЬ (A, B, n, I, m)
  НЦ ДЛЯ i = 1 ДО m
        Wi = 0
        НЦ ДЛЯ k = 1 ДО n
            Wi = Wi + Ai,k * Vk
        КЦ
  КЦ
КОНЕЦ ПРОГРАММЫ
```

Рис. 9

На рис. 54 показано решение задачи в системе MathCAD в соответствии с алгоритмом рис. 5.3.

$$m := 5 \qquad n := 4 \qquad ORIGIN := 1$$

$$A := \begin{pmatrix} 3 & 4 & 5 & 4 \\ 6 & 7 & 8 & 5 \\ 2 & 5 & 4 & 5 \\ 9 & 2 & 3 & 5 \\ 6 & 8 & 1 & 8 \end{pmatrix} \qquad V := \begin{pmatrix} 9 \\ 6 \\ 3 \\ 8 \end{pmatrix} \qquad A \cdot V = \begin{pmatrix} 98 \\ 160 \\ 100 \\ 142 \\ 169 \end{pmatrix}$$

$$W := \left| \begin{array}{l} \text{for } i \in 1..m \\ \quad \left| \begin{array}{l} W_i \leftarrow 0 \\ \text{for } k \in 1..n \\ \quad W_i \leftarrow W_i + A_{i,k} \cdot V_k \end{array} \right. \\ W \end{array} \right. \qquad W = \begin{pmatrix} 98 \\ 160 \\ 100 \\ 142 \\ 169 \end{pmatrix}$$

Рис. 10

Контрольные вопросы:

1. Что такое алгоритм?
2. Перечислите свойства алгоритма.
3. Перечислите и охарактеризуйте способы описания алгоритма.
4. Как вызвать панель Programming?
5. Как с помощью панели Programming ввести структуру *Следование*?
6. Как с помощью панели Programming ввести структуру разветвления *Обход*?
7. Как с помощью панели Programming ввести структуру разветвления с двумя ветвями?
8. Что такое арифметический цикл? Как он задается?
9. Как с помощью панели Programming создать структуру арифметического цикла?
10. Что такое рекурсия?

Лабораторная работа № 2

2.1. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

База данных должна отвечать следующим требованиям:

- Оперативность извлечения информации из базы данных.
- Полная доступность (вся информация в базе данных доступна для всех допущенных к ней пользователей).
- Гибкость (возможность внесения изменений в базу данных).
- Целостность (обеспечивается правильность взаимосвязей).
- Независимость (прибавление или изъятие части информации может производиться без нарушения работы базы данных).

Для того чтобы база данных удовлетворяла этим требованиям, информационная база перед размещением ее в памяти ЭВМ должна быть специальным образом преобразована (спроектирована), а управление базой данных должно осуществляться специальной компьютерной программой, которая называется *системой управления базами данных* (СУБД).

СУБД — это совокупность лингвистических (языковых) и программных средств, обеспечивающих создание, ведение и использование баз данных многими допущенными к ней пользователями.

Лингвистические средства СУБД — это язык определения данных (ЯОД) и язык манипулирования данными. Современные СУБД работают, главным образом, в интерактивном режиме (являются программами-оболочками). Это означает, что пользователь может не знать языков ЯОД и ЯМД. он воздействует с помощью клавиатуры и мыши на элементы пользовательского интерфейса СУБД, которая реагирует на эти воздействия, автоматически генерируя команды ЯОД и ЯМД, непосредственно воздействующие на базу данных.

СУБД может взаимодействовать не только с пользователем, но и непосредственно со специализированными прикладными программами, предназначенными для вычислений и обработки данных. *Взаимодействующие друг с другом база данных, СУБД и специализированные прикладные программы образуют информационную систему.*

Информационную систему рассматривают как программно-аппаратную систему, предназначенную для автоматизации целенаправленной деятельности конечных пользователей, обеспечивающую, в соответствии с заложенной в нее логикой обработки, возможность получения, модификации и хранения информации[5].

Структура информационной системы показана на рис. 11.

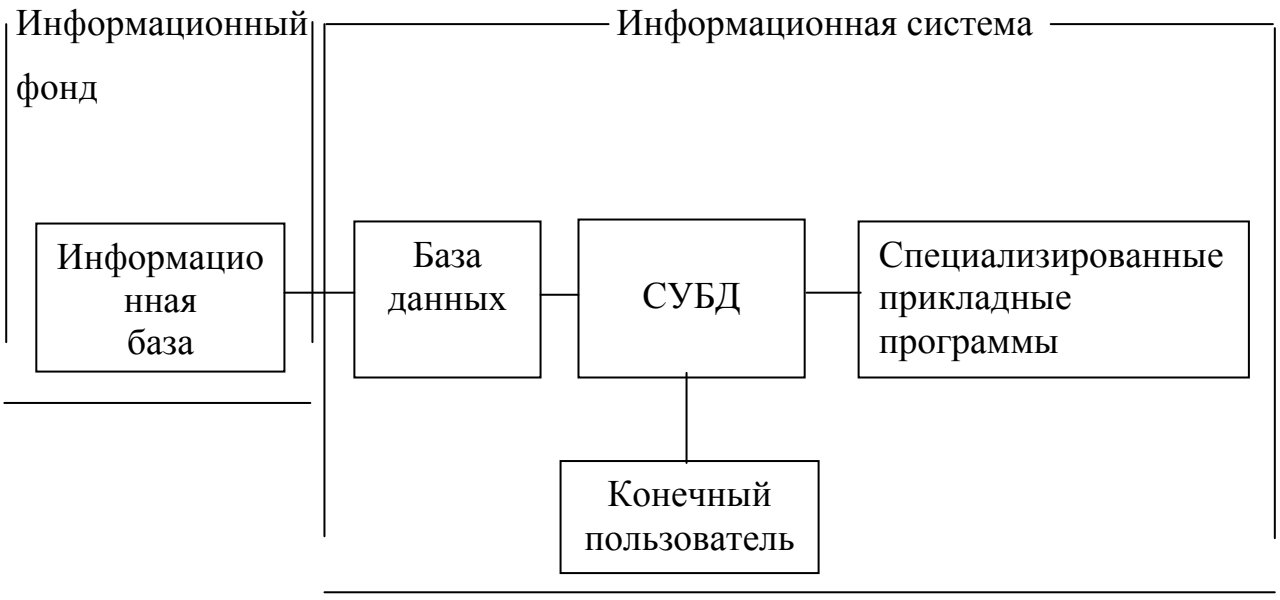


Рис. 11

Если база данных располагается в памяти одной ЭВМ, то такая база называется локальной. Если база данных располагается в памяти множества ЭВМ, размещенных в сети, то такая база данных называется распределенной.

Распределенная база данных состоит из набора узлов, связанных коммуникационной сетью, в которой:

каждый узел — это полноценная СУБД сама по себе;

узлы взаимодействуют между собой таким образом, что пользователь любого из них может получить доступ к любым данным в сети так, как будто они находятся на его собственном узле.

Каждый узел сам по себе является системой базы данных. Любой пользователь может выполнить операции над данными на своём локальном узле точно так же, как если бы этот узел вовсе не входил в распределённую систему. Распределённую систему баз данных можно рассматривать как партнёрство между отдельными локальными СУБД на отдельных локальных узлах.

Фундаментальный принцип создания распределённых баз данных: *Для пользователя распределённая система должна выглядеть так же, как нераспределённая система.*

Базой данных может пользоваться как один пользователь (см. рис. 1), так и несколько пользователей, располагающихся удаленно на разных узлах сети. При использования данных несколькими пользователями, применяется архитектура «КЛИЕНТ – СЕРВЕР». При этом сама база данных располагается на компьютере, который называется сервером. Удаленный пользователь управляет программой КЛИЕНТ, которая соединена средствами телекоммуникаций с программой СЕРВЕР, которая располагается на компьютере, содержащем базу данных. Пользователь с помощью программы КЛИЕНТ по средствам телекоммуникации посылает запрос программе СЕРВЕР. программа СЕРВЕР извлекает требуемую

информацию их базы данных и по средствам телекоммуникации направляет ее на программу КЛИЕНТ.

Проектирование реляционных баз данных представляет собой единый процесс, состоящий из последовательности следующих этапов:

- Инфологическое проектирование.
- Датологическое проектирование.
- Проверка результатов проектирования.

Далее описывается проектирование баз данных методом СУЩНОСТЬ-СВЯЗЬ или ER методом (от англ. Entity-сущность, Relationship связь).

2.2. Инфологическое проектирование

На инфологическом этапе осуществляется изучение и осмысление предназначенной для хранения в базе данных информации и ее взаимосвязей. Целью инфологического этапа проектирования баз данных является обобщенное отображение информационной базы. На этапе инфологического проектирования решаются следующие задачи:

- определение сущностей;
- определение атрибутов сущностей;
- определение типов связи между сущностями;
- построение концептуальной схемы.

Определение сущностей и атрибутов

Предназначенную для хранения в базе данных информацию следует распределить (разбить, разделить) на смысловые группы так, чтобы в каждой группе находилась информация о всех объектах одного типа. Для обозначения такой группы однотипных объектов вводится понятие *Тип сущности* или *Сущность*. Сущность имеет имя (как правило, существительное). Имя сущности объединяет информацию в отдельную

свою смысловую группу, указывает на нее. Для обозначения одного из объектов этой группы вводится понятие *Экземпляр сущности*. Например, кирпич, фундаментный блок, кафельная плитка - это экземпляры сущности СТРОИТЕЛЬНАЯ_ДЕТАЛЬ. Слова СТРОИТЕЛЬНАЯ_ДЕТАЛЬ - это имя сущности.

Для обозначения свойств сущности вводится понятие *Атрибут*. Сущность определяется набором характеризующих ее атрибутов. Каждый атрибут определяет какое-либо свойство своей сущности и имеет имя. Например, сущность СТРОИТЕЛЬНАЯ_ДЕТАЛЬ может иметь следующий набор атрибутов: *название_детали, вес, цена*. Слова *название_детали, вес, цена* - это имена атрибутов сущности СТРОИТЕЛЬНАЯ_ДЕТАЛЬ. Все экземпляры одной сущности характеризуются одинаковым набором имен атрибутов. Для каждого экземпляра сущности каждый атрибут принимает конкретное значение. Например, *кирпич, 6 кг, 100 руб*, это значения атрибутов *название_детали, вес, цена* соответственно одного из экземпляров сущности СТРОИТЕЛЬНАЯ_ ДЕТАЛЬ. Значения атрибутов называются *Данными*; для их хранения и проектируется *База данных*.

Атрибут, значения которого однозначно идентифицируют экземпляры сущности, называется *Ключевым атрибутом*. Остальные атрибуты сущности называются описательными атрибутами. Например, такой атрибут, как название детали может являться ключевым атрибутом (*Ключом, Простым ключом*), а вес не может, так как могут быть и другие детали того же веса. Значения ключевого атрибута всегда различны.

В некоторых случаях сущность не имеет ни одного атрибута, с различными значениями. Пусть, например, имеется сущность СОТРУДНИК с именами атрибутов: *Фамилия, Имя, Отчество, Год_рождения, Образование*. Любой из перечисленных атрибутов, взятый отдельно, не является ключевым, так как его значение может повториться. В этих случаях для однозначной идентификации экземпляров сущности вводится *Составной ключ* — то есть совокупность атрибутов, *комбинация* значений которых не

повторяется. В нашем примере таким составным ключом служит совокупность четырех атрибутов *Фамилия, Имя, Отчество, Год_рождения*

2.3. Датологическое проектирование

Логические модели данных

Прежде чем размещать данные в памяти ЭВМ, их надо упорядочить. Порядок расположения данных определяет логику их поиска. Порядок расположения (структура) данных, определяющий способ (логику) их поиска называется логической моделью данных.

После завершения инфологического этапа проектирования базы данных следует приступить к датологическому этапу. Цель датологического этапа - расположить в соответствии с логической моделью данных взаимосвязанные данные вместе с именами их атрибутов и сущностей, указанные в концептуальной схеме. Существует три логических модели данных: иерархическая, сетевая и реляционная.

В иерархической модели элементы данных разбиваются на иерархические уровни (в порядке старшинства происхождения). В результате образуются структуры, называемые деревьями. На самом верхнем уровне располагается элемент, называемый корнем. Каждый элемент, кроме корня, связан только с одним элементом на верхнем уровне. Каждый элемент имеет смысл, если рассматривается в контексте с более высокими по иерархии элементами. Доступ к каждому элементу происходит только через исходный элемент - корень. Для того чтобы найти элемент данных, надо указать, через какие элементы более высоких уровней надо пройти (путь доступа). Файловую систему MS DOS можно рассматривать как упрощенный вариант базы данных с логической моделью иерархического типа.

2.4. Задание на проектирование базы данных

Для обеспечения бесперебойного снабжения строек необходимыми стройматериалами со склада требуется хранить следующую информацию:

- Данные о заводах-поставщиках: название завода Н_ЗАВ, адрес завода А_ЗАВ, номер телефона директора ТЛФ_Д;
- Данные о стройматериалах: название Н_СТРМ, цена ЦЕНА, вес ВЕС и габариты ГАБАРИТ единицы стройматериала.
- Данные о стройках-потребителях стройматериалов: название стройки Н_СТР, адрес стройки А_СТР, номер телефона прораба ТЛФ_П.
- Количество производимых каждым заводам строительных материалов ПРОИЗ и ее распределение между стройками.
- Количество потребляемых стройками строительных материалов ПОТР и ее распределение между заводами.

Для хранения этой информации спроектировать, создать и в дальнейшем использовать многотабличную базу данных. Для этого:

Назначить имена типов сущностей и имена типов связей между сущностями. Для каждой сущности указать имена ключевых и описательных атрибутов. В соответствии со своим вариантом индивидуального задания начертить ER-диаграммы экземпляров и ER-диаграммы типов в двух вариантах:

— без учета распределения строительных материалов с заводов по стойкам (обезличенное хранение) с использованием бинарных связей;

— с учетом распределения строительных материалов с заводов по стойкам (адресное хранение) с использованием трехсторонних связей;

Написать совокупность отношений, как с учетом, так и без учета распределения строительных материалов по заводам и стойкам. Значения элементов доменов отношений, не указанные в индивидуальном задании, — вымышленные.

Варианты индивидуальных заданий

В дальнейшем будут использоваться следующие сокращенные обозначения: завод железобетонных изделий (з_жби), силикатный завод (сил_з), керамический завод (кер_з), деревообрабатывающий завод (дер_з), силикатный кирпич (с_кир), фундаментные блоки (ф_бл), кафельная плитка (каф_п), оконные блоки (о_бл), красный кирпич (кр_к), паркет (пар), керамическая плитка (кер_п), железобетонная панель (жел_п), школа (шк), жилой дом (жил_д), детский сад (д_сад), поликлиника (плк).

1) Варианты поставок стройматериалов заводами - поставщиками:

1. з_жби — ф_бл • сил_з — с_кир • кер_з — каф_п • дер_з — о_бл.
2. з_жби — ф_бл • сил_з — с_кир, кер_п • кер_з — каф_п, кр_к, с_кир • дер_з — о_бл, пар.
3. з_жби — ф_бл • сил_з — с_кир, кр_к • кер_з — каф_п, с_кир, кер_п • дер_з — о_бл, пар.
4. з_жби — ф_бл • сил_з — с_кир • кер_з — с_кир • дер_з — о_бл.
5. з_жби — ф_бл • сил_з — с_кир • кер_з — кер_п, каф_п, кр_к • дер_з — о_бл, пар.
6. з_жби — жел_п, ф_бл • сил_з — с_кир • кер_з — каф_п • дер_з — о_бл.
7. з_жби — жел_п • сил_з — с_кир • кер_з — каф_п • дер_з — о_бл.
8. з_жби — жел_п • сил_з — с_кир • кер_з — каф_п, кр_к, с_кир, кер_п • дер_з — о_бл, пар.
9. з_жби — жел_п, ф_бл • сил_з — с_кир • кер_з — каф_п, кер_п • дер_з — пар.
0. з_жби — жел_п • сил_з — с_кир • кер_з — с_кир • дер_з — о_бл.

2) Варианты потребления стройками стройматериалов:

1. с_кир — д_сад • ф_бл — жил_д • каф_п — плк • о_бл — шк.

2. с_кир — д_сад, жил_д • ф_бл — жил_д • каф_п — плк • о_бл — шк • кр_к — д_сад • пар — плк • кер_п — шк.

3. с_кир — шк, д_сад • ф_бл — жил_д • каф_п — плк • о_бл — шк, жил_д • кр_к — д_сад • пар — плк • кер_п — плк.

4. с_кир — шк, плк • ф_бл — жил_д • кр_к — д_сад • кер_п — жил_д.

5. с_кир — шк, д_сад • ф_бл — жил_д • каф_п — плк • о_бл — шк.

6. с_кир — д_сад • жел_п — жил_д • каф_п — плк • о_бл — шк.

7. с_кир — д_сад • жел_п — жил_д • каф_п — плк • о_бл — шк • кр_к — д_сад • пар — плк.

8. с_кир — шк, д_сад • жел_п — жил_д • каф_п — плк; о_бл — шк, жил_д • кр_к — д_сад • пар — плк.

9. с_кир — шк, плк • жел_п — жил_д • кр_к — д_сад • кер_п — жил_д.

0. с_кир — шк, д_сад • жел_п — жил_д • каф_п — плк • о_бл — шк • кр_к — жил_д • кер_п — плк.

Варианты индивидуальных заданий

Таблица 1

Номер студента по журналу	1	2	3	4	5	6	7	8	9	10	11	12
Производство	8	8	8	8	8	8	8	2	2	2	2	2
Потребление	5	2	3	4	8	9	0	2	3	4	5	8
Распределение	1	2	3	4	5	4	5	6	7	4	8	3
Номер студента по журналу	13	14	15	16	17	18	19	20	21	22	23	24
Производство	2	2	3	3	3	3	3	3	0	0	0	0
Потребление	9	0	3	4	5	8	9	0	3	8	2	0
Распределение	9	12	1	11	3	7	10	8	7	12	2	7

2.5. Пример выполнения задания на проектирование базы данных с обезличенным хранением

Инфологическое проектирование

В задании речь идет о заводах, которые производят стройматериалы, и стройках, которые эти стройматериалы потребляют. Поэтому данные сгруппируем в виде сущностей с именами ЗАВОД, СТРОЙМАТ, СТРОЙКА и соединим их бинарными типами связей с именами ПРОИЗВ, ПОТРЕБ. Запишем эти имена в одну строку, как это показано на рис. 1. Из задания следует, что нужно хранить значения атрибутов со следующими именами:

- для сущности ЗАВОД – Н_ЗАВ, А_ЗАВ, ТЛФ_Д;
- для сущности СТРОЙМАТ – Н_СТРМ, ВЕС, ГАБАРИТ, ЦЕНА;
- для сущности СТРОЙКА – Н_СТР, ТЛФ_П, А_СТР.

Очевидно, что из этих атрибутов ключевыми будут соответственно: Н_ЗАВ, Н_СТРМ, Н_СТР. В соответствии со своим индивидуальным заданием, приведенным на стр. 4, запишем значения ключевых атрибутов (это будут представители экземпляров сущностей) под именами своих сущностей и укажем линиями связи (экземплярами связей), какой завод что производит, и какая стройка что потребляет. В некоторых вариантах индивидуальных заданий значения ключевых атрибутов сущности СТРОЙМАТ могут повториться. В этом случае это значение повторно записывать не следует, а линию связи нужно провести к уже имеющемуся значению. Это будет означать, что два завода изготавливают одинаковый стройматериал. Если в строке потребностей строек появится название стройматериала, которого нет под именем сущности СТРОЙМАТ, то это название следует добавить в имеющуюся совокупность стройматериалов. Над каждой линией связи, которая указывает на сам факт связи, укажем числовую характеристику связи — сколько единиц стройматериала производится и потребляется.

Пусть задан вариант производства заводами строительных материалов:

дер_з — пар, о_бл; з_жби — ф_бл; кер_з — кр_к, кер_п, с_кир, каф_п; сил_з — с_кир

и вариант потребления стройками строительных материалов:

ф_бл — жил_д; с_кир — жил_д, д_сад; каф_п — плк; о_бл — шк.

Тогда, в соответствии с вышеизложенным, получится ER-диаграмма экземпляров, показанная на рис. 1.1.

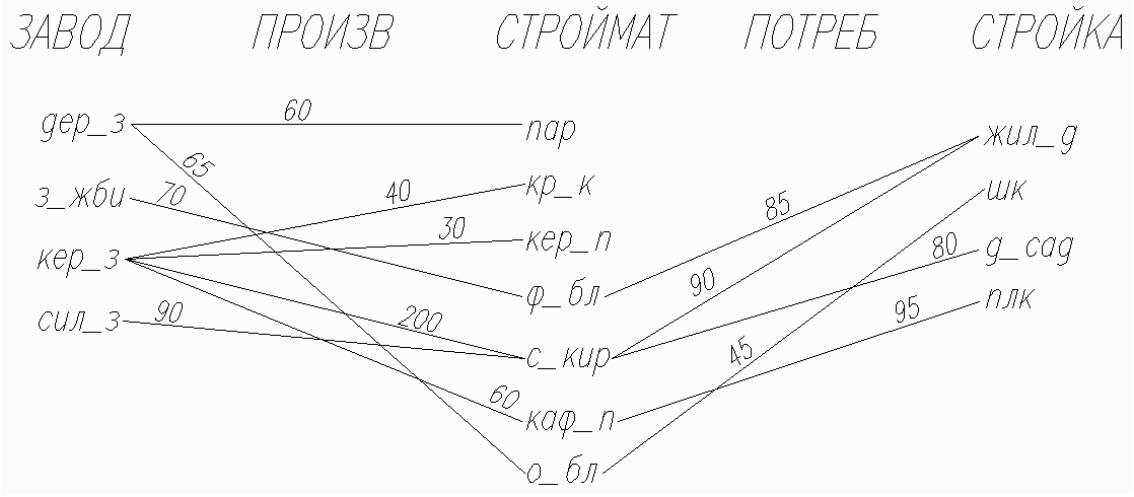


Рис. 12

Из диаграммы видно, что степень бинарной связи ПРОИЗВ M:N. Классы принадлежности сущностей ЗАВОД и СТРОЙМАТ обязательные по отношению к типу связи ПРОИЗВ.

Степень бинарной связи ПОТРЕБ N:M. Класс принадлежности сущности СТРОЙКА — обязательный. Класс принадлежности сущности СТРОЙМАТ по отношению к типу связи ПОТРЕБ — необязательный. Построим концептуальную схему (ER-диаграмму типов), представленную на рис. 1.2.

Рис. 12

Датологическое проектирование

В соответствии с правилом 6 для связи ПРОИЗВ и правилом 6 для связи ПОТРЕБ, концептуальная схема отображается на следующие 5 отношений, приведенные на рис. 1.3–1.7. Таким образом, три отношения ЗАВОД, СТРОЙМАТ и СТРОЙКА, представляющие одноименные сущности и два отношения ПРОИЗВ и ПОТРЕБ представляющие собой двусторонние (бинарные) связи образуют базу данных с обезличенным хранением строительных материалов.

СТРОЙМАТ

Н_СТРМ	ВЕС	ГАБАРИТ	ЦЕНА
каф_п	0,1	1-2-3	3
кер_п	0,15	80-250	5
кр_к	6	250×100×80	4
о_бл	70	1000×2000×150	150
пар	0,1	40×200	3
с_кир	5	150×200×100	7
ф_бл	7000	500×500×2000	3000

Рис. 13

ЗАВОД

Н_ЗАВ	ТЛФ_Д	А_ЗАВ
дер_з	36-68-13	Ул.Мира, 29
з_жби	35-56-21	Ул.Мира, 45
кер_з	56-57-78	Ул.Ленина, 5
сил_з	65-47-54	Ул.КИМ,76

Рис. 14

СТРОЙКА

Н_СТР	ТЛФ_П	А_СТР
д_сад	44-22	Ул.Козлова,6
жил_д	98-89	Ул.Елецкая,4
плк	33-88	Ул.Мира,8
шк	77-16	Ул.Огарева,10

Рис. 15

ПРОИЗВ

Н_ЗАВ	Н_СТРМ	ПРОИЗ
дер_з	пар	60
дер_з	о_бл	65
з_жби	ф_бл	70
кер_з	кр_к	40
кер_з	кер_п	30
кер_з	с_кир	200
кер_з	каф_п	60
сил_з	с_кир	90

Рис. 16

ПОТРЕБ

Н_СТРМ	Н_СТР	ПОТР
ф_бл	жил_д	85
с_кир	жил_д	90
с_кир	д_сад	80
каф_п	плк	95
о_бл	шк	45

Рис. 17

В нашем примере обезличка появляется потому, что силикатный кирпич поступает с заводов *сил_з* и *кер_з* на склад и выгружаются на одной площадке (вперемежку). Стройки *шк* и *д_сад* не знают, с какого завода поступил силикатный кирпич и кому в случае необходимости предъявить претензии по качеству. В обезличенной базе данных отсутствует информация о связях между тремя сущностями с тех случаях, когда хотя бы один экземпляр сущности (у нас это СТРОЙМАТ) связан не менее чем с одним экземпляром другой сущности и более чем с одним экземпляром третьей сущности.

Студенты составляют отчет по проектированию базы данных. Отчет должен иметь заголовок, состоящий из слова СКЛАД с последующими номерами вариантов производства и потребления, указанными в табл. 1. Например, для студента с номером по журналу 1 имя базы данных будет СКЛАД-8-5. Далее, в соответствии со своими вариантами производства и потребления, руководствуясь образцами, показанными рис. 11 – 17. соответственно студенты чертят ER-диаграмму экземпляров, ER-диаграмму типов, отношения ЗАВОД, СТРОЙМАТ, СТРОЙКА, ПРОИЗВ и ПОТРЕБ.

2.6. Правила датологического проектирования

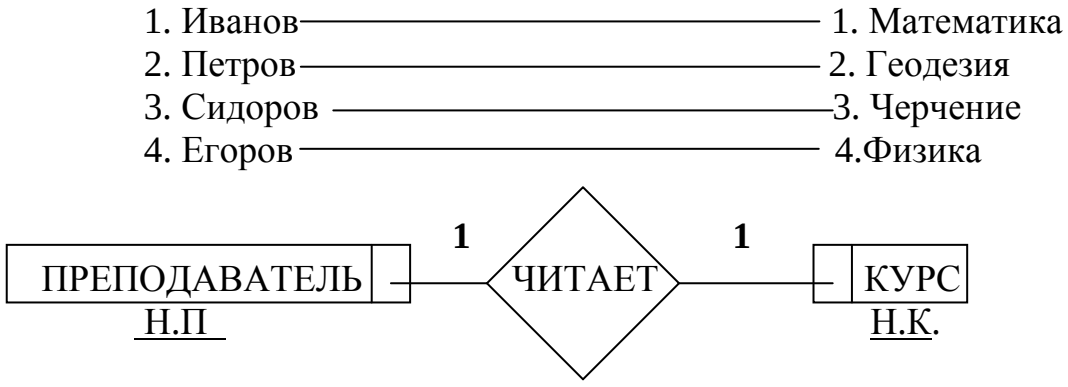
Задача датологического этапа проектирования – преобразование концептуальной схемы в совокупность отношений.

Существует два способа преобразования концептуальной схемы в совокупность отношений. *Первый способ* состоит в том, что каждая сущность заменяется *отношением*. Имя отношения совпадает с именем сущности. Каждый атрибут сущности представлен в отношении *доменом* (столбцом таблицы). Имя домена совпадает с именем соответствующего атрибута. Ключевому атрибуту сущности соответствует ключевой домен отношения. Каждый экземпляр сущности представлен в отношении соответствующим *кортежем* (строкой таблицы). Каждый тип связи также заменяется отношением связи. Отношение связи имеет два домена,

состоящие из значений ключевых атрибутов связываемых сущностей. Имена этих доменов совпадают с соответствующими именами ключевых атрибутов связываемых сущностей. Поэтому кортежи представляют собой экземпляры связи. Недостаток такого способа - большое количество отношений.

С целью сокращения количества отношений разработан *второй способ* преобразования концептуальной схемы в совокупность отношений. В дальнейшем будем пользоваться только вторым способом. При этом остаются в силе соглашения о соответствии имен отношений соответствующим именам сущностей и имен атрибутов именам доменов. При втором способе правила преобразования концептуальной схемы в совокупность отношений определяются в соответствии с классами принадлежности связываемых сущностей и степенью связи. Ниже приводятся эти правила. Для каждого правила на рис. 5 - 10 приведены соответствующие диаграммы экземпляров сущностей и экземпляров связей, концептуальные схемы и полученные совокупности отношений. Ключевыми атрибутами сущностей являются номер преподавателя и номер курса сокращенно обозначенные как Н.П. и Н.К.

ПРАВИЛО 1. Если степень связи равна 1:1 и класс принадлежности обеих сущностей является обязательным, то требуется только одно отношение с именами доменов, совпадающими с именами атрибутов обеих сущностей. Ключевым доменом этого отношения может быть ключевой атрибут любой из сущностей. (см. рис. 1.5).

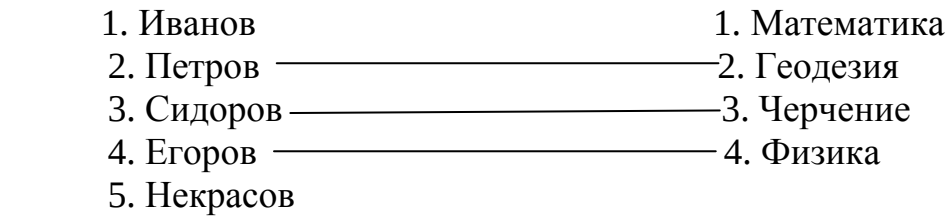


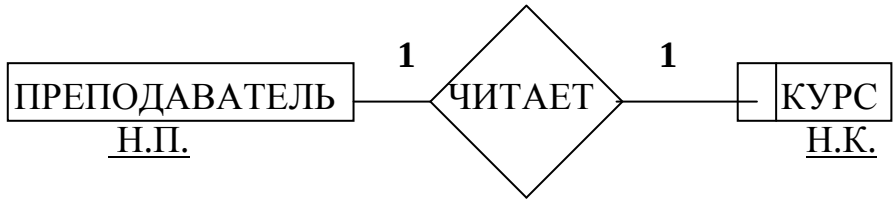
ПРЕПОДАВАТЕЛЬ_КУРС

Н.П	ФАМИЛИЯ	Н.К	НАЗВАНИЕ
1	Иванов	1	Математика
2	Петров	2	Геодезия
3	Сидоров	3	Черчение
4	Егоров	4	Физика

Рис.18

ПРАВИЛО 2. Если степень связи равна 1:1 и класс принадлежности одной сущности является обязательным, а другой необязательным, то необходимо построение двух отношений по одному от каждой сущности. Для связывания этих отношений ключевой атрибут сущности, для которой класс принадлежности является необязательным, добавляется в качестве домена (в качестве внешнего ключа) в отношение, выделенное для сущности с обязательным классом принадлежности. (См. рис. 6).





ПРЕПОДАВАТЕЛЬ

Н.П.	ФАМИЛИЯ
1	Иванов
2	Петров
3	Сидоров
4	Егоров
5	Некрасов

КУРС

Н.К	НАЗВАНИЕ	Н.П.
1	Математика	1
2	Геодезия	2
3	Черчение	3
4	Физика	4

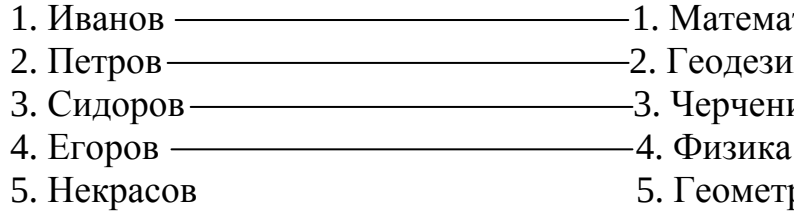
Рис.19

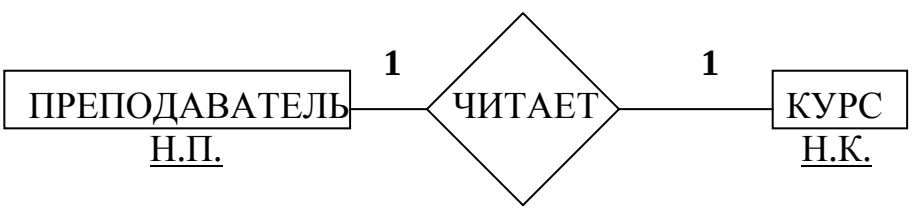
Внешний ключ служит для связывания кортежей двух отношений и располагается в том отношении, к которому осуществляется присоединение. Каждое значение внешнего ключа соответствует ER-диаграмме экземпляров и совпадает со значением первичного ключа того кортежа, который присоединяется.

ПРАВИЛО 3. Если степень связи равна 1:1 и класс принадлежности ни одной из сущностей не является обязательным, то необходимо использовать три отношения: по одному от каждой сущности, и одного (третьего) отношения для связи. Отношение, выделяемое для связи, должно иметь два домена (два внешних ключа), соответствующих ключевым атрибутам от каждой сущности. (См. рис. 20).

ПРАВИЛО 4. Если степень связи равна 1:N и класс принадлежности N-связной сущности является обязательным, то достаточным является использование двух отношений по одному от каждой сущности. Для их связывания ключевой атрибут односвязной сущности должен быть добавлен как домен (внешний ключ) в отношение, соответствующее N-связной сущности. (См. рис. 8).

Правило 4 сохраняет свою силу и в том случае, если степень связи не 1:N, а N:1. Правило 4 сохраняет свою силу независимо от класса принадлежности односвязной сущности.





ПРЕПОДАВАТЕЛЬ

Н.П.	ФАМИЛИЯ
1	Иванов
2	Петров
3	Сидоров
4	Егоров
5	Некрасов

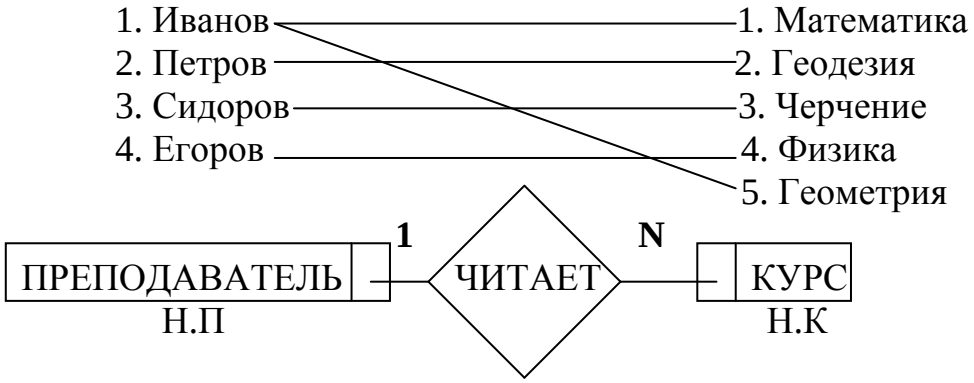
ЧИТАЕТ

Н.П.	Н.К.
1	1
2	2
3	3
4	4

КУРС

Н.К.	НАЗВАНИЕ
1	Математика
2	Геодезия
3	Черчение
4	Физика
5	Геометрия

Рис. 20



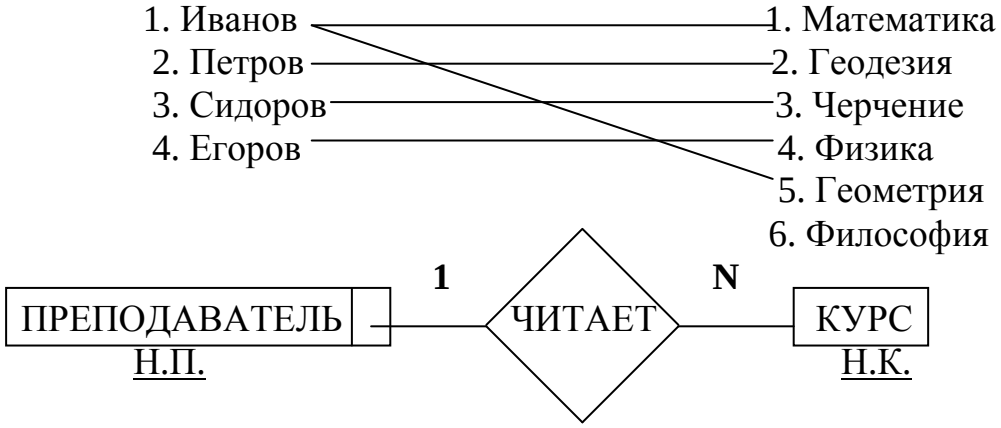
ПРЕПОДАВАТЕЛЬ

Н.П.	ФАМИЛИЯ
1	Иванов
2	Петров
3	Сидоров
4	Егоров

КУРС

Н.К.	НАЗВАНИЕ	Н.П.
1	Математика	1
2	Геодезия	2
3	Черчение	3
4	Физика	4
5	Геометрия	1

Рис. 21



ПРЕПОДАВАТЕЛЬ

Н.П.	ФАМИЛИЯ
1	Иванов
2	Петров
3	Сидоров
4	Егоров

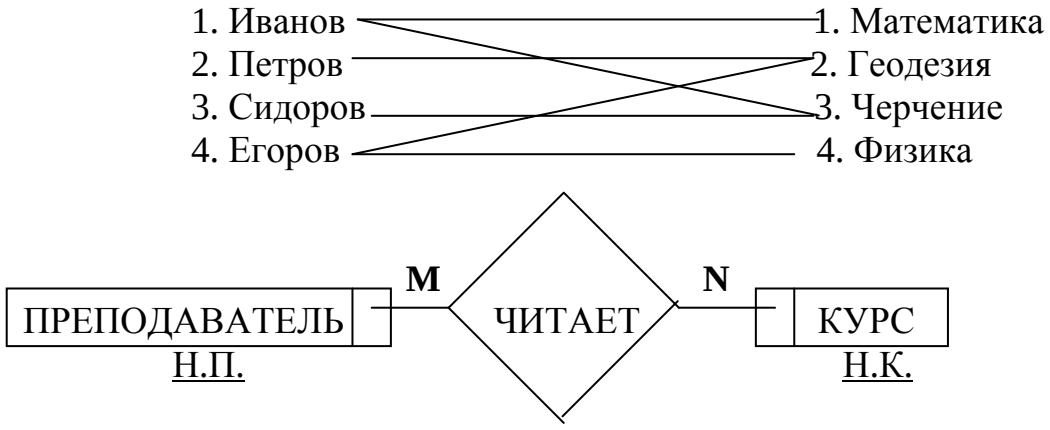
ЧИТАЕТ

Н.П.	Н.К.
1	1
2	2
3	3
4	4
1	5

КУРС

Н.К	НАЗВАНИЕ
1	Математика
2	Геодезия
3	Черчение
4	Физика
5	Геометрия
6	Философия

Рис. 22



ПРЕПОДАВАТЕЛЬ

Н.П.	ФАМИЛИЯ
1	Иванов
2	Петров
3	Сидоров
4	Егоров

Н.П.	Н.К.
1	1
2	2
3	3
4	4
1	3
4	2

КУРС

Н.К	НАЗВАНИЕ
1	Математика
2	Геодезия
3	Черчение
4	Физика

Рис. 23

ПРАВИЛО 5. Если степень связи равна 1:N и класс принадлежности N-связной сущности является необязательным, то необходимо формирование трех отношений: по одному от каждой сущности и одного отношения для связи. Отношение, выделяемое для связи, должно иметь два домена (два внешних ключа) соответствующих ключевым атрибутам от каждой сущности. (См. рис. 23).

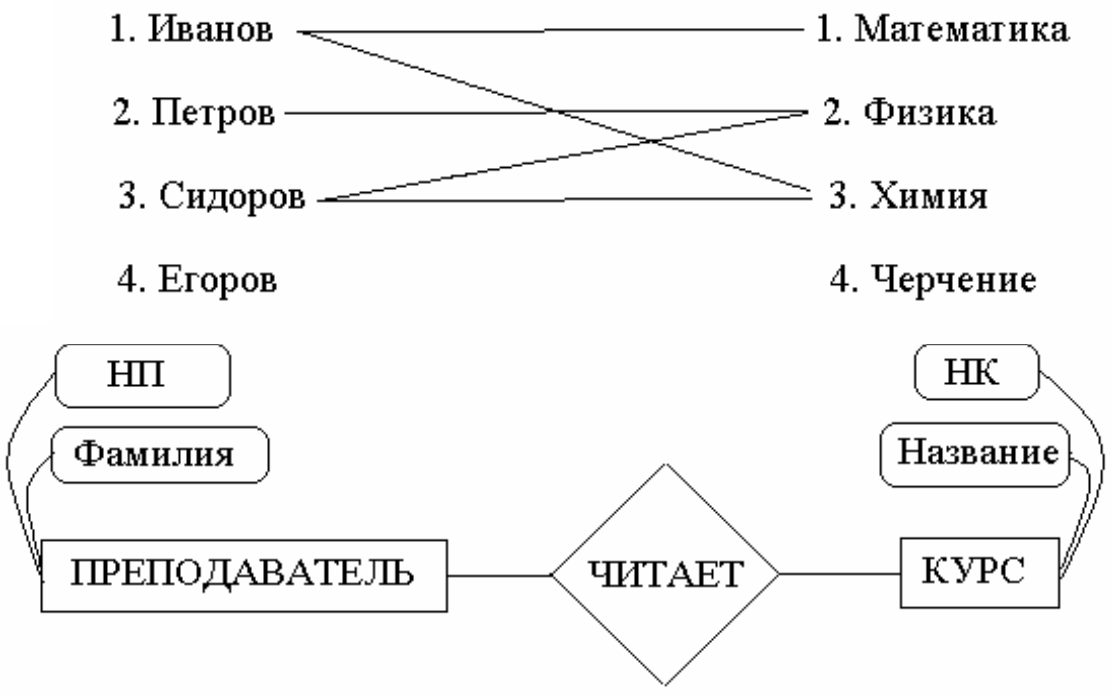
Правило 5 остается также в силе, если степень связи N:1. Правило 5 сохраняет свою силу независимо от класса принадлежности односвязной сущности.

ПРАВИЛО 6. Если степень связи равна M:N, то для хранения данных необходимо три отношения: по одному под каждую сущность и одного отношения для связи. Отношение, выделяемое для связи, должно иметь два домена (два внешних ключа) соответствующих ключевым атрибутам от каждой сущности. См. рис. 23.

Из правила 6 следует, что при степени связи M:N не имеет значения класс принадлежности каждой из сущностей.

2.7. Характеристики связей

Продолжим рассмотрение примера проектирования базы данных, предназначенной для хранения информации о преподавателях и читаемых ими курсах. Для определенности примем степень связи M:N и класс принадлежности обеих сущностей необязательный.



ПРЕПОДАВАТЕЛЬ

НП	Фамилия
1	Иванов
2	Петров
3	Сидоров
4	Егоров

ЧИТАЕТ

НП	НК
1	1
1	3
2	2

КУРС

НК	Название
1	Математика
2	Физика
3	Химия
4	Черчение

На рис. 11 представлены диаграмма экземпляров сущностей и связей, концептуальная схема и соответствующая данному примеру совокупность из трех отношений. Пусть требуется добавить в базу данных, представленную на рис. 11, информацию о количестве лекционных часов, которое каждый преподаватель затрачивает на чтение своего курса. В соответствии с вышеизложенным информация должна быть атрибутом какой-либо из сущностей.

Но при таком подходе хранение указанной информации в виде домена ЧАСЫ в составе отношений ПРЕПОДАВАТЕЛЬ или КУРС, представленных на рис. 11 невозможно.

Например, если вставить домен ЧАСЫ в состав отношения КУРС, то в строках физика и химия потребуется размещения двух чисел, а строке черчение будет пусто, что недопустимо.

Невозможно добавить домен ЧАСЫ в состав отношений ПРЕПОДАВАТЕЛЬ или КУРС, если степень связи M:N при любых классах принадлежности обеих сущностей, а также при необязательных классах принадлежности обеих сущностей и любых степенях связи.

Домен ЧАСЫ можно разместить в составе одного из отношений ПРЕПОДАВАТЕЛЬ или КУРС, если степень связи не равна M:N и одновременно класс принадлежности какой-либо сущности обязательный, подбирая сущность, в которую помещается домен ЧАСЫ. Однако, в этих случаях невозможно объяснить, почему одна и та же информация является атрибутом или одной или другой сущности.

Указанные трудности объясняются тем, что информация о количестве часов, затрачиваемых каждым преподавателем на чтение своего курса, не является характеристикой (атрибутом) какой либо из сущностей ПРЕПОДАВАТЕЛЬ или КУРС. Эта информация является характеристикой связи. В нашем примере связь характеризуется не только самим своим фактом (преподаватель ЧИТАЕТ), но также и тем, СКОЛЬКО ЧАСОВ читает.

Информация о связях в общем случае не сводится только к указанию на то, какой экземпляр одной сущности связан с каким экземпляром другой сущности (факт наличия связи). Эта информация может иметь и другие характеристики: *сколько* часов каждый преподаватель читает на каждом курсе (как в нашей задаче), а также, например, номер аудитории, *где* читаются лекции данного курса данным преподавателем, *время* проведения лекции в соответствии с расписанием и т.д.В частности, в приведенном, выше примере информация о факте наличия связи хранится в отношении ЧИТАЕТ. Поэтому и характеристику связи (домен ЧАСЫ) нужно также поместить в отношение ЧИТАЕТ.

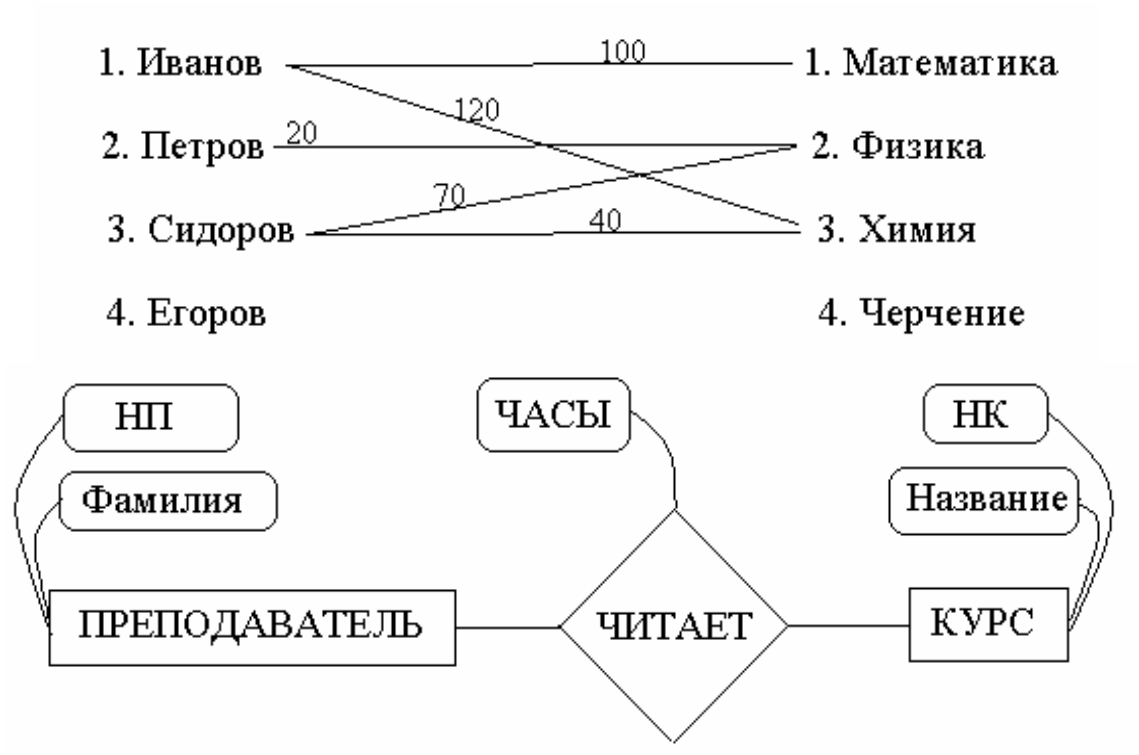


Рис. 24

ПРЕПОДАВАТЕЛЬ

НП	Фамилия
1	Иванов
2	Петров
3	Сидоров
4	Егоров

ЧИТАЕТ

НП	НК	ЧАСЫ
1	1	100
1	3	120
2	2	20
3	3	40
3	2	70

КУРС

НК	Название
1	Математика
2	Физика
3	Химия
4	Черчение

Рис. 25

На рис. 12 над каждой линией связи (экземпляром связи) указаны значения ее характеристики ЧАСЫ, которая отображена на концептуальной схеме как атрибут типа связи. Из рис. 12 видно, что характеристика типа связи ЧАСЫ составляет содержимое домена ЧАСЫ отношения ЧИТАЕТ.

Можно показать, что при использовании указанных выше соглашений, при всех возможных сочетаниях степени связи с классами принадлежности домен с характеристиками связи должен находится в составе того отношения, в котором располагаются два ключевых домена связываемых отношений, определяющих факт наличия связи. Независимость данных сохраняется всегда.

В литературе понятие «Тип связи» заменяется на понятие сущности особого вида, которая называется «Ассоциация», Этот вид сущности, также как и рассмотренный выше, может иметь неограниченное количество атрибутов.

КОНТРОЛЬНЫЕ ВОПРОСЫ:

1.Дать определение базы данных

2. Понятие СУБД

3. Понятие информационного фонда

4. Определение: сущность, атрибут сущности, ключевое поле.

5. Определение класса принадлежности

План выпуска учеб.-метод. документ. 2014 г., поз. 30

Публикуется в авторской редакции

Минимальные систем. требования:
PC 486 DX-33; Microsoft Windows XP; Internet Explorer 6.0; Adobe Reader 6.0.

Подписано в свет 17.04.2014.
Гарнитура «Таймс». Уч.-изд. л. 1,3. Объем данных 359 Кбайт.

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Волгоградский государственный архитектурно-строительный университет»
400074, Волгоград, ул. Академическая, 1
http://www.vgasu.ru, info@vgasu.ru