top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Данные, виды данных

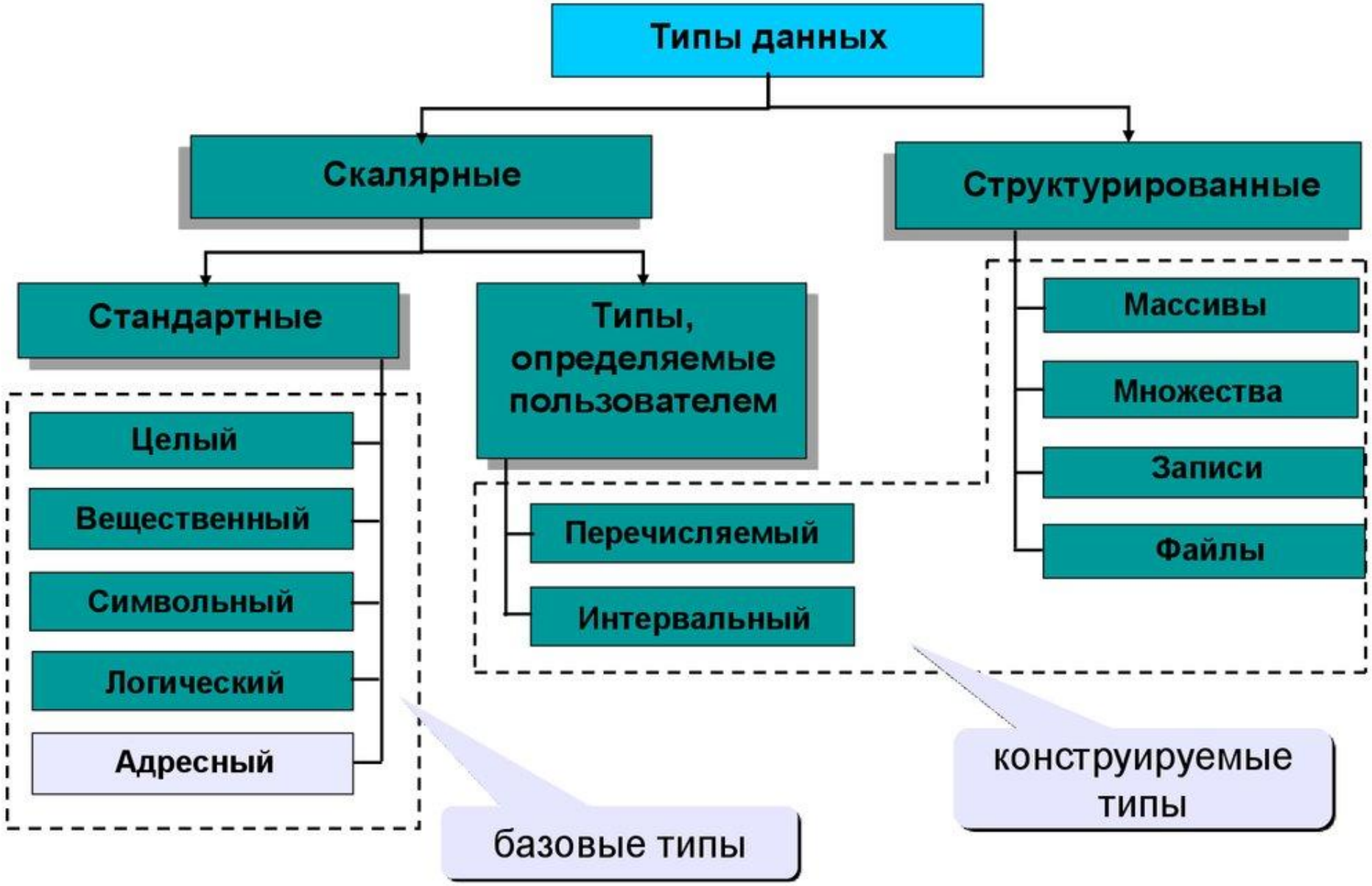
top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Виды данных в программировании – основополагающее понятие. Классификация данных позволяет определить, где они хранятся, что собой представляют и для каких операций могут применяться.

Виды данных в программировании – это варианты представления переменных конкретного объекта. Чтобы генерировать нужный код программирования, информация о типах данных должна быть доступна программисту и транслятору.

Виды данных = формат + размерные характеристики, диапазон показателей + операции.

Классификация типов данных



top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Различают базовые и производные виды данных в программировании:

1. **Первый вид (БВД)** включает данные, которые изначально заложены в транслятор. Как правило, они зависят от сферы применения языка программирования. В языке Си, который отличается зависимостью от архитектуры, базовые виды данных не отличаются от основных форматов, принятых в IT. Другими словами, при преобразовании операций с данными в различные машинные команды они (данные) совершенно не меняются.

2. **Второй вид – производные (ПВД),** программист преобразует из базовых. Это приводит к формированию иерархии видов данных. При этом для обозначения некоторых из них могут применяться дополнительные имена, которые также можно применять наряду с базовыми. Например, в языке Си++, который является объектно-ориентированным, это имена классов.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

РАЗНОВИДНОСТИ БАЗОВЫХ ТИПОВ ДАННЫХ В ПРОГРАММИРОВАНИИ

- **Числовые виды данных в программировании**

Целочисленные.

Виды данных в программировании делятся на знаковые и беззнаковые. Уже понятно из наименования: в знаковых могут храниться все действительные числа, а также ноль, а в беззнаковых – только положительные (больше нуля).

У беззнаковых данных диапазон больше в 2 раза, чем у знаковых. Это – из-за компьютерного восприятия: в знаковых типах бит отражает знак числа, где 0 является положительным значением, а 1 – отрицательным.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

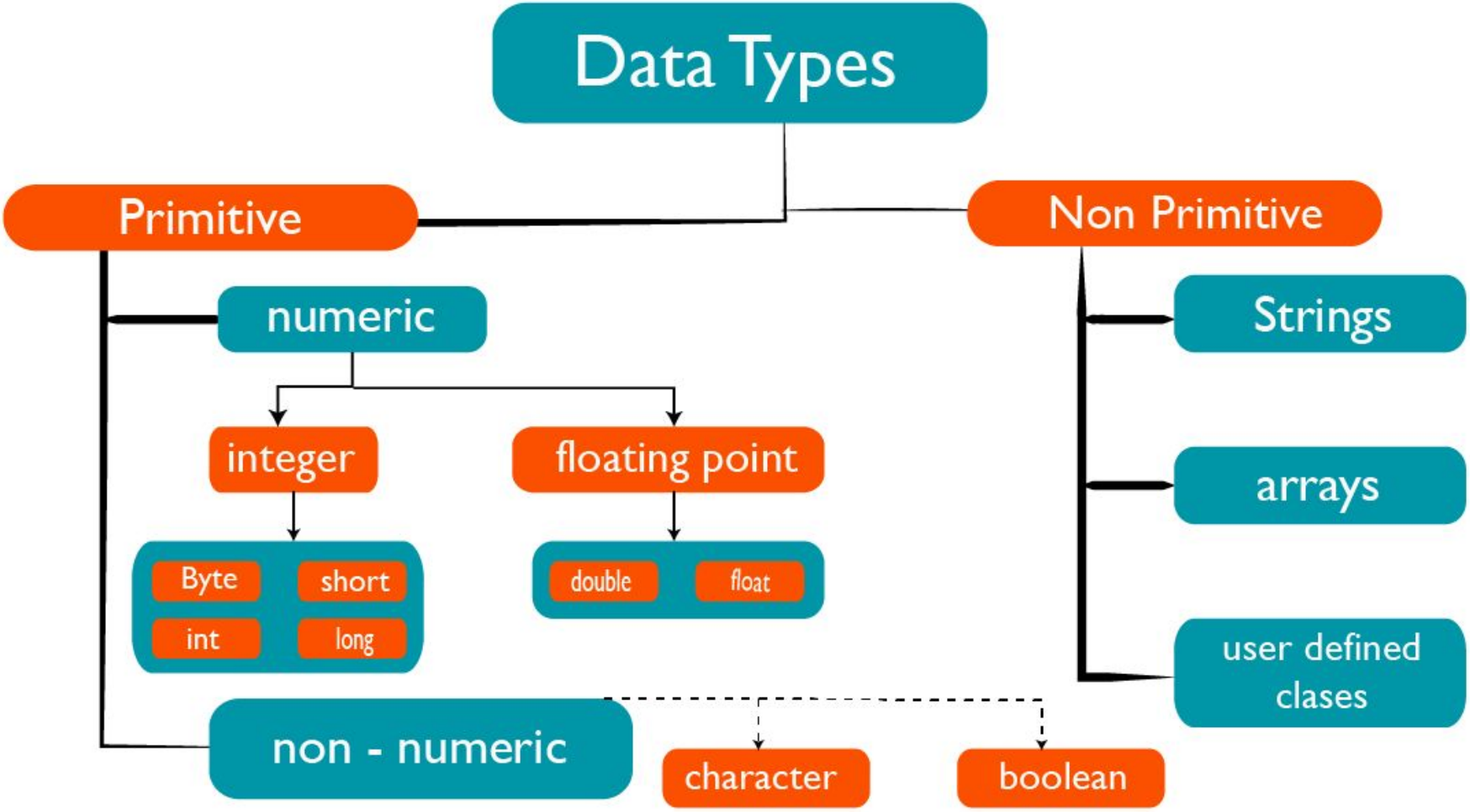
Учитывая восприятие компьютерными устройствами целого значения, в ячейке памяти из n бит может храниться и 2^{n-1} для знаковых типов, и 2^n – для беззнаковых:

- **Тип short (короткий целый.)** Для него в памяти отведено 16 бит, то есть 2 байта (216 = 65 536). Диапазон значений, который может принять тип short со знаком – это [-32 768; 32 767].

- **Переменный тип long (длинный целый).** Этому типу выделено 64 бита, то есть 8 байт. (264 = 1,8 446 744 * 1 019). Он имеет внушительный диапазон: в случае знакового типа это [-9 223 372 036 854 775 808 9 223 372 036 854 775 807]. Также модификатор long может использоваться в связке с другими типами (long будет указан перед наименованием типа, допустим, long double). Благодаря этому увеличивается диапазон возможных значений.

Тип	Количество байтов	Диапазон значений
short	2	–32 768 ÷ 32 767
unsigned short	2	0 ÷ 65 535
int	4	–2 147 483 648 ÷ 2 147 483 647
unsigned int	4	0 ÷ 4 294 967 295
long	4	–2 147 483 648 ÷ 2 147 483 647
unsigned long	4	0 ÷ 4 294 967 295
long long	8	–9 223 372 036 854 775 808 ÷ 9 223 372 036 854 775 807
unsigned long long	8	0 ÷ 18 446 744 073 709 551 615
signed char	1	–128 ÷ 127
unsigned char	1	0 ÷ 255
bool	1	false или true
float	4	3.4E +/– 38 (7 знаков после запятой)
double	8	1.7E +/– 308 (15 знаков после запятой)

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ



top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

ВЕЩЕСТВЕННЫЕ.

Значения этого типа имеют плавающую запятую.
Плавающая запятая — форма представления действительных чисел, в которой число хранится в форме мантиссы и показателя степени. Если говорить на языке программирования, то каждое число может быть представлено в следующей форме:

$N = M * 10^p$,
где N — записываемое число;
M — мантисса;
p (целое) — порядок.

Тип	Значение	Число значащих цифр
REAL	$2,9 * 10^{-39} \ldots 1,7 * 10^{38}$	11-12
SINGLE	$1,5 * 10^{-45} \ldots 3,4 * 10^{38}$	7-8
DOUBLE	$5,0 * 10^{-324} \ldots 1,7 * 10^{308}$	15-16
EXTENDED	$3,4 * 10^{-4932} \ldots 1,1 * 10^{4932}$	19-20
COMP	-263+1.....263-1	19-20

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Например: 14 441 544 = 1,4 441 544 $*10^{7}$; 0,0 004 785 = 4,785 $*10^{-4}$. На мониторе компьютера вы увидите следующие значения:

1,4441544E+7; 4,785E–4.

Таким образом, в предназначенном месте памяти хранится целое число фиксированной длины и последовательность вносимого значения. Разберем пример типа данных, хранящихся в 64 битах. Мантисса будет равна 53 битам: 1 для знака числа и 52 для её значения; порядок 10 битов: 1 бит для знака и 10 для значения. Здесь можно порассуждать о диапазоне точности, а именно какое самое большое и самое маленькое число может хранить рассматриваемый тип данных: от 4,94 * 10^{-324} до 1,79 * 10^{308}.

Но мы помним, что компьютерная память не резиновая, поэтому сохраняются первые разряды мантиссы. По-другому их ещё именуют значащими.
Итог: вещественные виды данных в программировании, примеры которых мы привели, характеризуются количеством значащих разрядов и диапазоном точности, что отличает их от целочисленных.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

СИМВОЛЬНЫЙ ТИП ДАННЫХ В ПРОГРАММИРОВАНИИ.

В символьном типе переменная имеет только один символ, целое число.
В соответствии с кодировкой, он преобразуется в некий символ.
Символьному виду данных в программировании присущ только размер выделяемой под них памяти.

- Для обозначения этого типа используется ключевое слово **char**.
- Тип **char** может быть со знаком или без знака. В величинах со знаком можно хранить значения в диапазоне от **-128** до **127**. По умолчанию тип **char** являемся знаковым, то есть спецификатор **signed** использовать не обязательно. При использовании спецификатора **unsigned** значения могут находиться в пределах от **0** до **255**.

ЛОГИЧЕСКИЙ ТИП ДАННЫХ В ПРОГРАММИРОВАНИИ.

У этого типа данных могут быть следующие значения: false (ложь) или true (правда). Ему соответствуют в языках C# и C++ тип bool, в Java – boolean.

Категория	Логические (булевские) значения
Синтаксис	**boolean**
Константы	**true**, **false**
Операции	**and**, **or**, **xor**, **not**
Реализация	байт (слово) : 0 – false, 1 - true

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

ПЕРЕЧИСЛИМЫЙ ВИД ДАННЫХ В ПРОГРАММИРОВАНИИ.

Для внутреннего представления этот вид аналогичен целочисленному, но в нем программист использует уже готовые конкретные строковые значения вместо чисел.

Для более точного представления разберем пример на языке C++ (в C# и Java – аналогично):
enum Forms {shape, sphere, cylinder, polygon};

В данном случае переменные перечислимого вида данных Forms могут иметь только те значение, которые указаны в примере. Это удобно, поскольку мы работает не с числами, а определенными смысловыми значениями. Но компьютер считывает данные как целые числа.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

МАССИВЫ ДАННЫХ.

Теперь рассмотрим сложные виды данных в программировании. И на первом месте – массив. Массив – это последовательно выстроенная и имеющая общее имя структура данных, в которой хранятся элементы одного типа. Его можно представить как набор пронумерованных ячеек, в каждую из которых поместили какие-то данные (один элемент данных в конкретную ячейку).

Каждый массив определяется типом данных составляющих его элементов, а они могут быть абсолютно любыми. В программировании не допускается использование всего массива, работа осуществляется с определенной его частью. Для того, чтобы добраться до него, в трёх указанных выше языках применяют оператор «[]»:
array[0].

Индексом массива является целое число, ссылающее на определенную часть массива. Индекс, как правило, имеет вид int.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

СТРУКТУРА.

До этого мы разобрали встроенные виды данных в программировании. Далее рассмотрим пользовательский тип данных.
Структура хранит в себе комплект переменных различных видов. В программировании структуры нужны для того, чтобы объединить близкие по значению вещи.

КЛАСС.

Другим пользовательским видом данных в программировании является класс. Класс наделен теми же возможностями, что и структура, но, помимо параметров, он ещё имеет и методы. Также класс поддерживает большое число вещей, которые объединены объектно-ориентированным программированием.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Примеры видов данных в программировании

В языке Python используются следующие типы данных программирования:

- int — целочисленный;
- char — символьный;
- bool — логический;
- float —с плавающей запятой;
- double —с плавающей запятой двойной точности.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Язык программирования JavaScript содержит следующие типы данных:

srting — тип данных «строка»;

number — «число»;

object — тип данных, хранящих свойства и методы;

undefined — тип данных, значения которых не определены;

boolean — логический;

null —с «пустыми» значениями.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

ТИП ДАННЫХ — «ЧИСЛО».

В таком виде данных могут быть как дробные, так и целые числа.

int — целые, то есть числа без дробной части;

number — числовые данные (в JavaScript);

double — тип данных с плавающей запятой двойной точности;

float — вещественные, дробные числа с десятичной точкой.

Целые числа используют для подсчета, а вещественные — для измерения таких свойств, как вес.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

ЛОГИЧЕСКИЕ (БУЛЕВЫЕ) ТИПЫ ДАННЫХ В ПРОГРАММИРОВАНИИ — boolean.

Принимая решение, что необходимо выполнить далее, компьютер анализирует и задает вопрос – истина или ложь? «Да» или «Нет»? Вопросы, на которые существует только 2 варианта ответа, являются логическими (булевыми) выражениями.Логический тип может принимать либо значение «истина» (true), либо значение «ложь» (false). После этого компьютер анализирует, сравнивает данные (числа, строки и переменные) и принимает решение.

Существуют ещё такие виды данных в программировании, как null, undefined, object (объект) — в JavaScript или list (список), dict (словарь), tuple (кортеж) — в Python. Но чтобы познать азы программирования, достаточно будет знаний типов данных «строка», «число» и логическое значение.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Основой языка программирования Паскаль, как и любого другого языка, является **алфавит — набор допустимых символов, которые можно использовать для записи программы**. Это:

• латинские прописные буквы (A, B, C, ..., X, Y, Z);
• латинские строчные буквы (a, b, c, ..., x, y, z);
• арабские цифры (0, 1, 2, ..., 7, 8, 9);
• специальные символы (знак подчёркивания; знаки препинания; круглые, квадратные и фигурные скобки; знаки арифметических операций и др.).

В качестве неделимых элементов (составных символов) рассматриваются следующие последовательности символов:

:= (знак операции присваивания);
>= и <= (знаки ≥ и ≤);
(* и *) (начало и конец комментария).

В языке существует также некоторое количество различных цепочек символов, рассматриваемых как единые смысловые элементы с фиксированным значением. Такие цепочки символов называются служебными словами.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Для обозначения констант, переменных, программ и других объектов используются имена — любые отличные от служебных слов последовательности букв, цифр и символа подчёркивания, начинающиеся с буквы или символа подчёркивания.

Прописные и строчные буквы в именах не различаются.
Длина имени может быть любой. Для удобства мы будем пользоваться именами, длина которых не превышает 8 символов.

Служебное слово языка Паскаль	Значение служебного слова
and	и
array	массив
begin	начало
do	выполнить
else	иначе
for	для
if	если
of	из
or	или
procedure	процедура
program	программа
repeat	повторять
then	то
to	до (увеличивая до)
until	до (до тех пор, пока)
var	переменная
while	пока

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

ТИПЫ ДАННЫХ, ИСПОЛЬЗУЕМЫЕ В ЯЗЫКЕ ПАСКАЛЬ

В языке Паскаль используются различные типы данных. Мы будем пользоваться некоторыми из так называемых простых типов данных

Название	Обозначение	Допустимые значения	Область памяти
Целочисленный	integer[1]	-32 768 .. 32 767	2 байта со знаком
Вещественный	real	$\pm(2{,}9 \cdot 10^{-39} .. 1.7 \cdot 10^{+38})$	6 байтов
Символьный	char	Произвольный символ алфавита	1 байт
Строковый	string	Последовательность символов длиной меньше 255	1 байт на символ
Логический	boolean	true и false	1 байт

integer — основной, но не единственный тип для работы с целочисленными данными. Дополнительную информацию по этому вопросу вы можете найти в справочниках по программированию на языке Паскаль.

В вещественном числе целая часть от дробной отделяется точкой, при этом перед точкой и после неё должно быть, по крайней мере, по одной цифре. Пробелы внутри числа недопустимы.

СТРУКТУРА ПРОГРАММЫ НА ЯЗЫКЕ ПАСКАЛЬ

В программе, записанной на языке Паскаль, можно выделить:

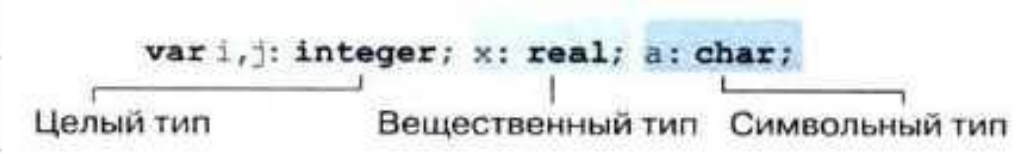
1) заголовок программы;
2) блок описания используемых данных;
3) блок описания действий по преобразованию данных (программный блок).

Заголовок программы состоит из служебного слова program и имени программы. После имени программы ставится точка с запятой.

Блок описания данных состоит из раздела описания констант (const), раздела описания переменных (var) и некоторых других разделов [2]. В разделе описания переменных указываются имена используемых в программе переменных и их типы.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Имена переменных одного типа перечисляются через запятую, затем после двоеточия указывается их тип; описание каждого типа заканчивается точкой с запятой. Ниже приведён пример раздела описания переменных:



Программа может не иметь заголовка; в ней может отсутствовать блок описания данных. Обязательной частью программы является программный блок. Он содержит команды, описывающие алгоритм решения задачи. Программный блок начинается со слова begin и заканчивается словом end с точкой.
Ниже приведён общий вид программы:

```
program <имя программы>;
  const <список постоянных значений>;
  var <описание используемых переменных>;
begin <начало программного блока>
  <оператор 1>;
  <оператор 2>;
  ...
  <оператор n>
end.
```

- Операторы — языковые конструкции, с помощью которых в программах записываются действия, выполняемые над данными в процессе решения задачи.
- Точка с запятой служит разделителем между операторами, а не является окончанием соответствующего оператора.
- Перед оператором end точку с запятой ставить не нужно.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

ОПЕРАТОР ПРИСВАИВАНИЯ

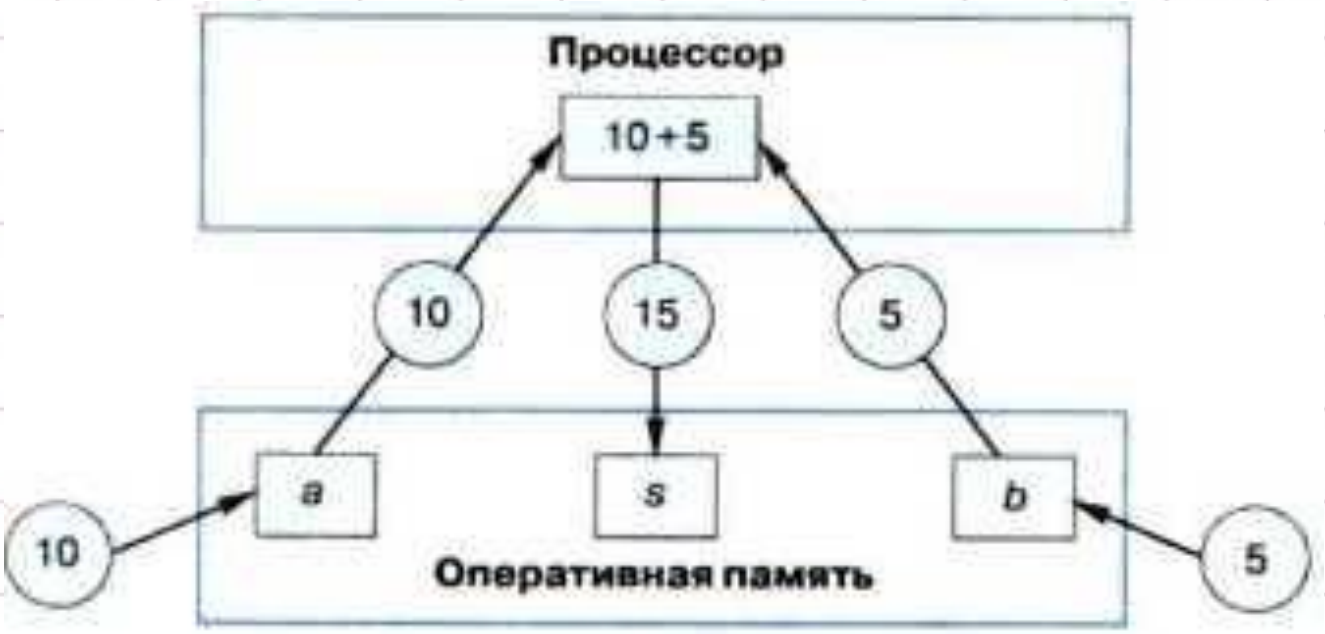
Основное преобразование данных, выполняемое компьютером, — **присваивание переменной нового значения**, что означает изменение содержимого области памяти; оно осуществляется оператором присваивания, аналогичным команде присваивания алгоритмического языка.

Общий вид оператора:
<имя переменной>:=<выражение>
Операция присваивания допустима для всех приведённых в табл. 3.2 типов данных. Выражения в языке Паскаль конструируются по рассмотренным ранее правилам для алгоритмического языка.

Рассмотрим процесс выполнения операторов присваивания на следующем примере:

```
a:=10;
b:=5;
s:=a+b
```

При выполнении оператора a:=10 в ячейку оперативной памяти компьютера с именем a заносится значение 10; при выполнении оператора b:=5 в ячейку оперативной памяти компьютера с именем b заносится значение 5.
При выполнении оператора s:=a+b значения ячеек оперативной памяти с именами a и b переносятся в процессор, где над ними выполняется операция сложения.
Полученный результат заносится в ячейку оперативной памяти с именем s



top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

САМОЕ ГЛАВНОЕ

Паскаль — универсальный язык программирования, получивший своё название в честь выдающегося учёного Блеза Паскаля.
В языке Паскаль используются различные типы данных: целочисленный (**integer**), вещественный (**real**), символьный (**char**), строковый (**string**), логический (**boolean**) и другие.

В программе, записанной на языке Паскаль, можно выделить:

1) заголовок программы;
2) описание используемых данных;
3) описание действий по преобразованию данных (программный блок).

Словарь языка

Служебное слово языка Паскаль	Значение служебного слова
and	и
array	массив
begin	начало
do	выполнить
else	иначе
for	для
if	если
of	из
or	или
procedure	процедура
program	программа
repeat	повторять
then	то
to	до (увеличивая до)
until	до (до тех пор, пока)
var	переменная
while	пока

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

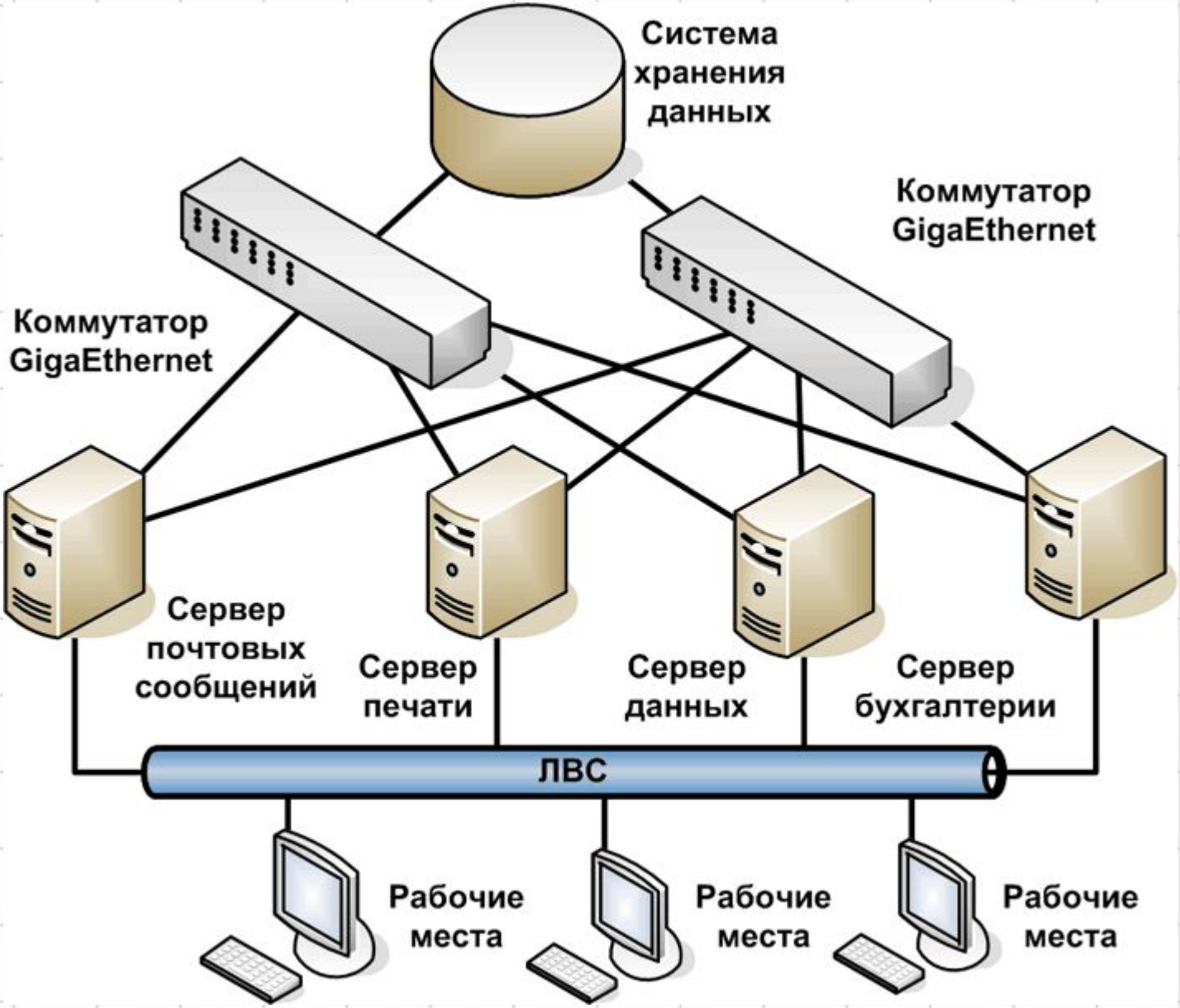
Общий вид программы:

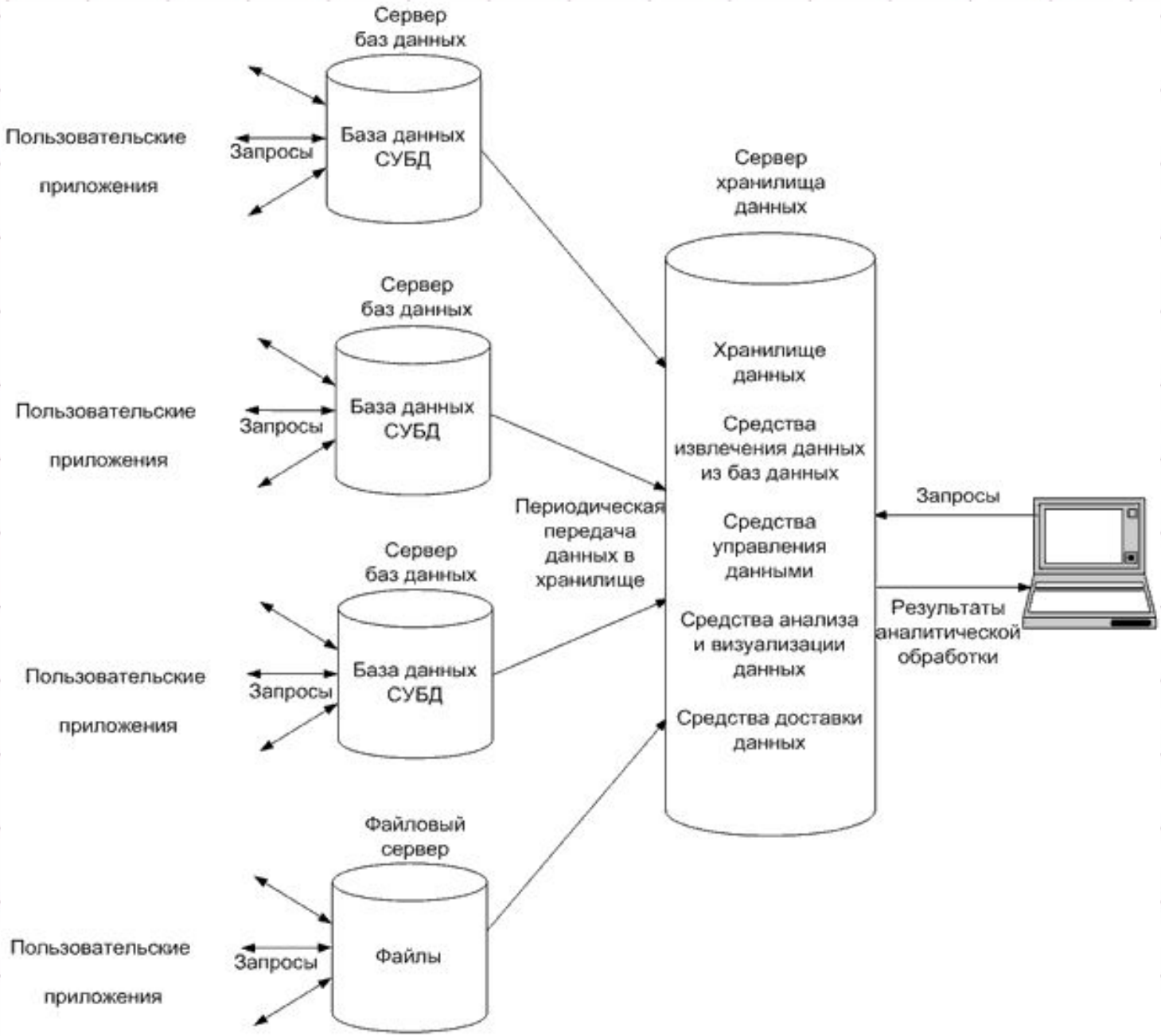
```
program <имя программы>;
  const <список постоянных значений>;
  var <описание используемых переменн
begin
   <оператор 1>;
   <оператор 2>;
   …
   <оператор N>
end.
```

Словарь языка

Служебное слово языка Паскаль	Значение служебного слова
and	и
array	массив
begin	начало
do	выполнить
else	иначе
for	для
if	если
of	из
or	или
procedure	процедура
program	программа
repeat	повторять
then	то
to	до (увеличивая до)
until	до (до тех пор, пока)
var	переменная
while	пока

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ




Сервер баз данных
Пользовательские приложения
Запросы
База данных СУБД
Сервер баз данных
Пользовательские приложения
Запросы
База данных СУБД
Сервер баз данных
Пользовательские приложения
Запросы
База данных СУБД
Файловый сервер
Пользовательские приложения
Запросы
Файлы
Периодическая передача данных в хранилище
Сервер хранилища данных
Хранилище данных
Средства извлечения данных из баз данных
Средства управления данными
Средства анализа и визуализации данных
Средства доставки данных
Запросы
Результаты аналитической обработки

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

ИНФОРМАЦИЯ И СУБД

Появление компьютеров не сразу привело к разработке информационных систем. На заре вычислительной техники компьютеры обладали ограниченными возможностями в части памяти. Оперативная память не обладает свойством долговременного хранения, магнитные ленты не обеспечивают прямого доступа, а магнитные барабаны имели ограниченный объем. Развитие информационных систем началось с появлением магнитных дисков и файловых систем. Информационные системы, главным образом, ориентированы на хранения и вывод и модификацию постоянно существующей информации. Структура данных в разных информационных системах может быть различна, но между ними больше общего.

Стремление выделить и обобщить общую часть информационных систем , ответственную за управление сложно-структурированными данными является одной из побудительных причин создания СУБД.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

СУБД – СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ.

Понятие согласованности данных является ключевым понятием баз данных. Если информационная система поддерживает согласованное хранение информации в нескольких файлах, то можно говорить о том, что она поддерживает базу данных. Если некоторая вспомогательная система позволяет работать с несколькими файлами, обеспечивая их согласованность, то можно назвать ее системой управления базами данных.

Для обеспечения параллельной работы с базой данных использование файлов не дает возможности одновременной модификации данных, т.к. первый же процесс накладывает блокировки на весь файл целиком, что приводит к генерации ошибки для параллельных процессов. СУБД обеспечивают более тонкую синхронизацию параллельного доступа к данным.

Если информационная система поддерживает два согласованных файла сотрудники и отделы и была добавлена запись файл сотрудники, а в файле отделы модификация записи не была выполнена из-за аварийного выключения питания, то при перезапуске системы данные будут находиться в рассогласованном состоянии. Согласование информации после сбоев берет на себя СУБД. То есть прикладная система не обязана отвечать за согласованность данных.

СТРУКТУРЫ ХРАНЕНИЯ ДАННЫХ И МЕТОДЫ ДОСТУПА К НИМ

Структурой хранения называется любое упорядочение данных на диске. Могут быть реализованы различные структуры хранения. Более того эти структуры могут меняться по мере изменения требований к производительности системы.

ДОСТУП К БАЗЕ ДАННЫХ

Поиск и предоставление данных пользователю осуществляется с помощью нескольких программ. При этом можно выделить следующие уровни доступа к данным: СУБДß(запрос данных)(возвращение данных)àДиспетчер файловß (запрос файла)(возвращение файла)àДиспетчер дисковß(дисковая операция ввода, вывода)(чтение данных с диска)àБД.

1) В СУБД определяется искомая запись и затем для ее извлечения запрашивается диспетчер файлов.
Запись – хранимая информация об объекте базы данных.
2) Диспетчер файлов определяет страницу, на которой находится искомая запись, а затем для извлечения этой страницы запрашивают диспетчер диска.
Страницей (блоком) устройства ввода/вывода называется количество данных, передаваемых из вешней памяти в оперативную за одно обращение. Размер страницы обычно кратен одному килобайту.
3) Диспетчер дисков определяет физическое положение искомой страницы на диске, и посылает соответствующий запрос на ввод/вывод данных. Если в результате предыдущих запросов искомая страница уже находится в оперативной памяти, то этот пункт не выполняется.

С точки зрения СУБД база данных представляет собой набор записей, которые могут просматриваться с помощью диспетчера файлов. Диспетчер файлов рассматривает базу данных как набор страниц, просматривая с помощью диспетчера дисков. Диспетчер дисков непосредственно работает с диском.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

ДИСПЕТЧЕР ДИСКОВ

Диспетчер дисков является компонентом ОС. При выполнении дисковых операций необходимо знать физические адреса на диске. Диспетчер файлов рассматривает диск как набор страниц фиксированного размера с уникальным идентификационным номером набора страниц. Каждая страница обладает уникальным внутри данного набора идентификационным номером страницы.

Соответствие физических номер на диске и номеров страниц достигается с помощью диспетчера дисков. Преимуществом такого подхода является аппаратная независимость программных компонентов высокого уровня.
Один из наборов страниц называется набором пустых страниц и содержит все имеющиеся свободные страницы.

Основные операции выполняемы диспетчером диска:
1) извлечь страницу P из набора страниц S
2) заменить страницу P из набора страниц S
3) добавить новую страницу в набор страниц S (то есть извлечь одну страницу из набора пустых страниц и возвратить новую страницу с номером P)
4) удалить страницу P из набора S (то есть возвратить страницу с номером P в набор пустых страниц)

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Файлом называется набор однотипных записей. Основными операциями выполняемыми диспетчером файлов являются:

1) извлечь запись R из файла F
2) заменить запись M из файла F
3) добавить новую запись R в файл F
4) удалить запись R из файла F
5) создать новый файл F
6) удалить F

В одних системах диспетчер файлов является компонентом ОС а в других поставляется в составе СУБД.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

КЛАСТЕРИЗАЦИЯ

Это процесс как можно более близкого физического размещения на диске, логически связанных между собой и часто используемых данных.

Сегментà экстент (непрерывная последовательность блоков). Различают внутрифайловую и межфайловую кластеризацию. Например, у нас есть два файла: файл поставщиков и файл товаров. Если нам нужно получить поставщиков с номерами идентификаторов от ИД1 до ИД9, то оптимальным будет выполнение кластеризации поставщиков в соответствии с возрастанием убыванием) идентификаторов.

Если требуется одновременно получать информацию о поставщике и его товарах, то записи из двух разных файлов должны располагаться рядом. Для эффективного доступа. Это пример межфайловой кластеризации.

В каждый момент времени кластеризацию можно осуществить только одним из способов поскольку это связано с физическим размещением данных на диске.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

ФИЗИЧЕСКОЕ РАЗМЕЩЕНИЕ ДАННЫХ НА ДИСКЕ

Логическая последовательность страниц задается с помощью указателей, то есть логически близко находящиеся данные физически могут отстоять друг от друга значительно.

С целью сохранения близкого расположения логически связанных страниц на диске диспетчер дисков обычно размещает или удаляет страницы в наборах не по одной, а экстентами.

Для получения информации о размещении различных наборов страниц, диспетчеру дисков достаточно знать расположение только первой страницы в группе, расположение остальных определяется с помощью указателей в заголовках страниц. Все имеющиеся наборы страниц вместе с указателями на первые страницы каждого из наборов перечислены в отдельном месте на диске. Это место, а именно страница часто называется таблицей содержания диска, каталогом набора страниц или просто страницей «нуль».

Записи в пределах страницы также можно разместить в соответствии с логическим порядком.

Записи идентифицируются с помощью идентификационного номера записи (RID). Этот RID состоит из двух частей – номера страницы, на которой данная запись находится и байта смещения слота от конца страницы. Этот байт смещения содержит байт смещения записи от начала страницы. Эта схема является компромиссом между быстротой и непосредственной адресацией и гибкостью косвенной. В результате записи внутри страницы могут сдвигаться вверх и вниз (для поддержания соответствия между логическим и физическим порядком) без изменения идентификационных номеров записей.

Согласно выше – изложенному, для каждого файла всегда можно осуществить последовательный доступ ко всем записям. Такая последовательность называется физической хотя она не обязательно соответствует физическому размещению данных на диске. Запись может содержать дополнительную информацию в так называемой приставке, в частности здесь может содержаться информация об идентификационном номере файла, которому принадлежит запись (в случае межфайловой кластеризации), длина записи (для записи переменной длины), флаг удаления, указатель при связывании записей в цепочку и тд.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Индексирование

Пример: Поставщики

Номер	Фамилия	Скидка	Город
	Иванов		Ростов
	Петров		Таганрог
	Сидоров		Таганрог
	Федоров		Ростов
	Агафонов		Азов

При наличии индекса, имеется несколько стратегий доступа к данным из файла поставщики. Пример: найти поставщиков из города Ростов. 1я стратегия: прочитать (scan) файл поставщиков и выбрать те строки, у которых город = Ростов. 2я стратегия: чтение файла городов (scan). Для строк, у которых город = Ростов извлекаются записи из файла поставщики согласно RID указателям.

Если доля поставщиков из Ростова, по отношению к их количеству достаточно мала, то 2я стратегия будет эффективней 1й. Это связано с тем, что в силу упорядоченности записей в файле города, поиск будет прекращен с следующего за ростовом названия города, во вторых, если придется просмотреть файл городов полностью, то при этом потребуется значительно меньше операций ввода вывода. Поскольку файл городов имеет размер меньший чем файл поставщиков.

В просматриваемом примере файл городов называется индексным файлом или просто индексом по отношению к файлу поставщиков. Индексный файл это файл, который содержит данные из другого файла, который называется индексированным (в примере файл поставщики) и вид указателя фактические данные (файл поставщики). RID указатель служит для связывания записи индексного файла с записью индексируемого файла.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Использование индексов на ряду с ускорением выборки имеет недостаток , который связан с замедлением процесса обновления данных. Так как при изменении данных в индексированном файле придется менять данные в индексных файлах. И чем больше индексных файлов тем медленней процесс.

Индексы так же могут использоваться для проверки наличия некоторого значения. Например для ответа на вопрос, если поставщики из города Батайск, достаточно будет просмотреть только индексный файл. Файл может иметь несколько индексов, которые могут использоваться совместно. Кроме того индексы могут быть составными, которые эффективно могут использоваться при поиске значений в первом столбце или в первом втором, третьем.

Индексы часто называют инвертированными списками. Файл с индексами по каждому полю называется полностью инвертированным.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

ПЛОТНОЕ И НЕПЛОТНОЕ ИНДЕКСИРОВАНИЕ

Плотный индекс имеет вход для каждой записи индексированного файла. В индексе используются RID указатели, хотя достаточно было бы только номеров страниц. Так как извлечение конкретной записи уже будет выполняться в ОП. То есть число дисковых операций ввода вывода не увеличится. Если физическая последовательность файл поставщиков соответствует логической последовательности (например по номер по номеру поставщика). Иначе говоря, в этом файле выполнена кластеризация по полю номера. Если проиндексировать файл по полю номер, то нет необходимости в таком индексе хранить указатели для каждой записи индексируемого файла. Достаточно хранить максимальный номер поставщика для каждой страницы и номер соответствующей страницы. Такой индекс называется неплотным.
Преимущество использования неплотных индексов заключается в их малом размере поскольку просматриваться такие файлы будут с большой скоростью. Однако, с помощью одного такого индекса нельзя ответить на вопрос о наличии некоторого значения.
Для данного файла может быть построено любое число плотных индексов и только 1 неплотный индекс.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

СТРУКТУРА ТИПА B-TREE

Дерево состоит из двух множеств: множество вершин и множество дуг. Корневой называется вершина не имеющая входящих в нее дуг. Дерево всегда имеет только 1 корень. Листом называется вершина, которая не имеется выходящих дуг. У листьев нет дочерних вершин, а у корня нет родителя. Каждая вершина кроме корня имеет только одну родительскую вершину. Дерево называют сбалансированным тогда и только тогда, когда каждый его лист имеет одинаковое число предков. B-tree является индексными последовательными файлами имеющими иерархические индексы. Подобные структуры используются практически во всех СУБД. Буква B означает сбалансированная, а не бинарная. В отличии от бинарных деревьев, которые юзаются в ОП и каждый узел имеет не более 2х потомков, узлы B дерева имеют множество потомков. Каждый узел сбалансированного дерева на диске фактически занимает полностью весь блок. В итоге в каждом блоке БЗ содержится сотня указателей на потомков. При поиске в B tree проверяется только 3 дисковых блока.

Необходимость создания структуры B tree заключается в желании заключается в желании в избежание просмотра всего содержимого индексного файла. Идея состоит в том, чтобы создать для индексного файла еще 1 индекс и так далее. При этом индекс на каждом из уровней будет не плотным по отношению к нижнему индексированному уровню. Такой многоуровневый индекс состоит из двух частей: набора последовательностей и набора индексов.
Набор последовательностей включает одноуровневый индекс для реальных данных, который обычно является, при этом записи внутри индекса сгруппированы по страницам, а страницы связаны в цепочку, что дает возможность организовать быстрый последовательный доступ к индексированным данным.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

НАБОР ИНДЕКСОВ

Обеспечивает быстрый непосредственный доступ к наборам последовательностей (последовательным данным). Фактически набор индексов является индексным файлом со структурой B-tree. Комбинация набора индексов и набора последовательностей называется структурой типа B+tree.

Одним из недостатков иерархических структур является несбалансированность их работы после удаления или вставки некоторых элементов. В результате элементы с реальными данными могут оказаться на разных расстояниях от корневого элемента.

Продолжительность поиска в несбалансированной структуре может оказаться непредсказуемой. Преимуществом структуры типа B tree является возможность сбалансированной вставки или удаление значений.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Для вставки нового значения V в структуру типа B-tree порядка n. Пусть V = 41.

Например, может быть использован следующий алгоритм:
1) на самом низком уровне набора индексов следует найти элемент содержащий 2m индексных записей. Нам нужно отложит элемент N содержащий 2m индексных записей. Если элемент N содержит свободное пространство, то значение вставляется в него и на этом процесс завершается.
2) в противном случае (если свободного пространства нет) элемент N разделяем на два элемента N1 и N2. Обозначим символом S множество 2m+1 значений, в котором 2m несходных значений и 42. Первые m значений этой логической последовательности и среднее между ними значение W необходимо поместить в элемент N1. А последние m в элемент N2. Значение W необходимо поместить в родительский элемент P на более высоком структурном уровне. В последствии при осуществлении поиска некоторого значения U, достижения элемента P, поиск будет перенаправлен в сторону элемента N1 (если U<=W), либо в сторону элемента N (если U> W).

Далее этот процесс следует повторить для вставки среднего значения W в родительский элемент P на более высоком структурном уровне.

В худшем случае процесс разделения элемента структуры может продлиться вплоть до корневого элемента с образованием нового иерархического уровня.

Для удаления элемента следует применить аналогичный алгоритм в обратном порядке.

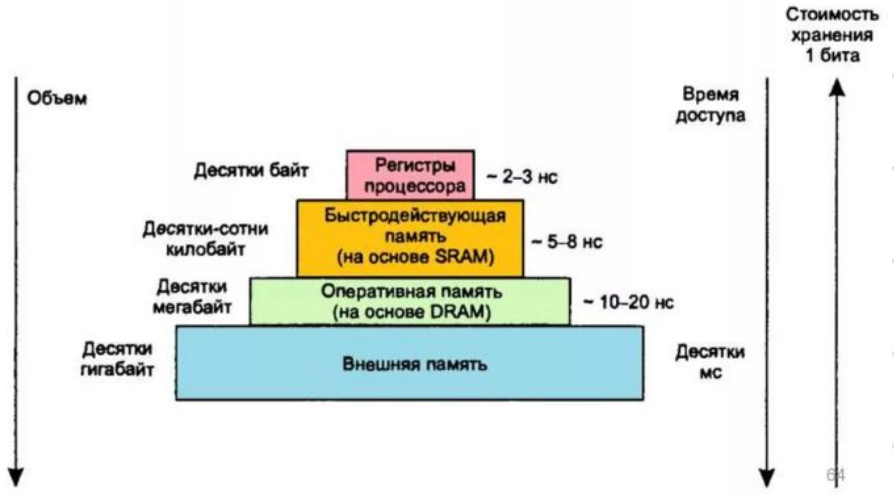
top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

КЭШИРОВАНИЕ

Стилем и порядком дерева называется максимально допустимое количество дочерних узлов для каждого родительского узла. Большие степени обычно используются для создания более широких и менее глубоких деревьев. На практике каждый узел в дереве является страницей и потому в нем может храниться значительно больше 3х указателей и 2х ключевых значений как в рассмотренном примере. Пример: страница имеет размер 4096 байт, указатель 4 байта, размер индексируемого поля тоже 4 байта, указатель на узел того же уровня тоже 4 байта, и того: на одной странице можно хранить (4096-4)/(4+4)=511 записей. Таким образом порядок B-tree+ равен 512. То есть корень дерева должен содержать до 511 записей и иметь до 512 дочерних узлов. Каждый дочерний узел также может содержать до 511 записей и 512 для его дочерних узлов. Кэшированием, кэш адресацией или хэш - индексированием называется технология быстрого прямого доступа к записи на основе заданного значения некоторого поля. При этом: 1) каждая запись помещается по адресу (reed указатель или номер страницы), который вычисляется при помощи специальной хэш функции на основе значения некоторого поля данной записи, которая называется хэш полем иди хэш ключом. Вычисленный адрес называется хэш адресом. 2) для сохранения записи сначала вычисляется хэш адрес новой записи, а затем диспетчер файлов помещает эту запись по вычисленному адресу. 3) для извлечения записи по заданному значению хэш поля сначала вычисляется хэш адрес а затем запись извлекается по этому адресу.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Для выбора «хорошей» функции хэширования необходимо знать, во первых примерное число записей в файле и во вторых сколько записей может поместиться в блок. Эта информация нужна для вычисления количества блоков в файле и соответственно диапазона значений функции хэширования. Если для файла будет выделено слишком мало блоков, то эффективность доступа будет уменьшаться по мере их переполнения. Если наоборот, для размещения файла будет предусмотрено слишком много блоков, то доступ будет осуществляться быстро, но значительная часть памяти в этом случае останется незанятой данными. Обычно считается нормальным, когда занято данными две трети блока.



top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Функция хэширования выбирается таким образом чтобы максимально удовлетворить следующим условиям:

1)каждое ее значение должно быть целым из интервала от 0 до (число блоков -1),
2) она должна быть удобной для вычислений, то есть вычисления значения функции должно происходить быстро,
3) для множества реальных значения ключа значения функции хэширования должны быть явно вероятными.

Если хэш функция удовлетворяет 3му условию, то говорят, что она перемешивает значения ключа. Если это условие нарушено, то может оказаться так, что часть блоков окажется переполненной, в то время как остальные будут заполнены незначительно, что отрицательно скажется на эффективности доступа.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

НЕПРИГОДНОСТЬ ФАЙЛОВ С ХЭШ АДРЕСАЦИЕЙ К ГРУППОВОЙ ОБРАБОТКЕ

Прямой доступ имеет место, когда требуется найти одно значение ключа. В процессе групповой обработки требуется получить несколько значений (или говорят ответов).

Назовем коэффициентов активности отношение числа ответов к числу записей в файле. Коэффициент активности близок к единице, когда требуется получить почти все записи из файла (например платежная ведомость).

Прямой доступ имеет место, когда коэффициент активности близок к нулю. То есть нужно найти одну запись. При групповой обработке использовать последовательный файл бывает выгодней чем файл с хэш адресацией.

Для того чтобы правильно выбрать организацию файла, необходимо хотя бы приблизительно знать коэффициент активности файла.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

ЦЕПОЧКИ УКАЗАТЕЛЕЙ

Для выполнения запроса типа “найти поставщиков из города N” можно применить способ хранения данных с использованием цепочек указателей (смотри рисунок). В данном случае этот способ будет более эффективным чем индексирование.

Из файла поставщиков извлекается атрибут город, который формирует родительскую структуру городов. Файл поставщиков по отношению к этому родительскому файлу называется дочерним. При выполнении запроса в такой дочерней структуре в файле городов находится нужный город, а затем по цепочке указателей извлекаются записи поставщиков из данного города.

При таком способе не будет выполняться чтение записей поставщиков из других городов. Другим преимуществом данной структуры является более простое выполнение вставки и удаления записей по сравнению с индексной структурой. Кроме того эта структура занимает меньше места на диске, поскольку нет повторения названия городов.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

НЕДОСТАТКИ СТРУКТУРЫ

1. Для доступа к Nму поставщику требуется перебрать все предыдущие записи поставщиков. В худшем случае, для каждого доступа может потребоваться отдельная дисковая операция чтения.

2. Структура весьма неэффективна для обратного запроса: «найти город данного поставщика». Дело в том, что для поиска названия города приходится просматривать всю цепочку. В итоге часто совместно с цепочкой указателей приходится использовать другие структуры.

3. Создать родительско – дочернюю структуру в уже имеющемся наборе записей не так просто, поскольку цепочки указателей пронизывают все записи. А значения соответствующих полей необходимо извлечь и выстроить родительский файл. Выполнить же индексирование имеющегося набора гораздо проще.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

МЕРЫ ПО УЛУЧШЕНИЮ СТРУКТУРЫ

1) Указатели можно задать также и в обратном направлении. При этом существенно упрощается процесс согласования указателей при удалении дочерней записи.
2) В записи дочерней структуры можно добавить указатель на родительскую запись.
3) Можно не удалять атрибут город из дочерней структуры.

В данном файле можно задать как любое количество индексов так и любое количество цепочек указателей.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

УПРАВЛЕНИЕ ДАННЫМИ РАСПОЛОЖЕННЫМИ В ОПЕРАТИВНОЙ ПАМЯТИ

Возможность размещать базу данных целиком в оперативной памяти появилась в связи с резким снижением цен на модули памяти и широким распространением 64х разрядных операционных систем.

При простом перемещении данных «дисковой» СУБД в оперативную память, ожидаемого повышения производительности не произойдет. Дело в том, что реляционная база данных функционирует исходя из предположения, что обрабатываемые данные в основном находятся на диске. Алгоритмы оптимизации, управление буферным пулом и технология извлечения данных на основе индексов – все это существенным образом ориентировано на дисковое хранение данных.

Оптимизация запросов также выполняется исходя из предположения о хранении данных на диске. При работе с данными резидентно находящимися в ОП, надобность в буферном пуле отпадает.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

В традиционной СУБД узел B-tree представляет собой страницу диска и содержит максимально возможное число указателей на дочерние узлы. В результате дерево имеет малую глубину и большую ширину. Такая структура позволяет уменьшить число операций ввода-вывода. Если данные находятся в ОП, то эта структура будет работать хуже бинарных деревьев, так как вычислительная мощь процессора растрачивается на сравнение значений индекса в B-tree, а так же на управление буферами, которые содержат данные и индексы.

В Times Ten используется индекс типа T-tree оптимизированный для доступа к ОП. Для навигации по дереву используется указатели <= и >, представляющие собой непосредственно ссылки на адрес памяти, а не на дисковую страницу.

В Times Ten используются технологии индексирования двух типов: Хэш-индексы и T-tree. Хэш индексы используются, когда требуется найти точные совпадения с образцом, а T-tree более подходит для поиска значений в диапазоне. Хэшированные индексы создаются автоматически, когда в данных задается первичный ключ, а T-tree создаются при помощи команды.

top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ