

Основы программирования и баз данных



Модуль 4.

СТРУКТУРЫ ДАННЫХ. ОСНОВЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ

- Базовые структуры данных – массивы и записи
- Основные операции над структурами данных
- Динамические структуры данных. Списки. Стеки. Деревья
- Информационная система. Понятие базы данных
- Требования пользователей к базам данных
- Проектирование баз данных; Цели и этапы проектирования

Модуль 4.

СТРУКТУРЫ ДАННЫХ.

ОСНОВЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ

(продолжение)

- Инфологический аспект. Модель «сущность-связь»
- Даталогический аспект. Модели данных (иерархическая, сетевая, реляционная), их достоинства и недостатки
- Реляционные базы данных. Понятие отношения. Нормализация
- Системы управления базами данных.
- Базы данных и компьютерные сети. Сетевые и распределённые базы данных.

Базовые структуры данных – массивы

- **Массив (индексный)** — простая статическая структура данных, предназначенная для хранения набора единиц данных, каждая из которых идентифицируется индексом или набором индексов.
- **Индекс** — обычно целое число, либо значение типа, приводимого к целому, указывающее на конкретный элемент массива.

12	25	99	20	...	37
0	1	2	3	...	N-1

- Количество используемых индексов массива может быть различным. Массивы с одним индексом называют *одномерными*, с двумя — *двумерными* и т. д.
- Одномерный массив соответствует вектору в математике, двумерный -- матрице.

Материал из Википедии — свободной энциклопедии

Базовые структуры данных – массивы (продолжение)

- **Специфические типы массивов**

- **Статические массивы.**

- Статическим называется массив, размер которого не может изменяться во время исполнения программы

- **Динамические массивы.**

- *Динамическим* называется массив, размер которого может меняться во время исполнения программы.
 - Для реализации динамики язык программирования должен предоставлять встроенную поддержку.

- **Гетерогенные массивы.**

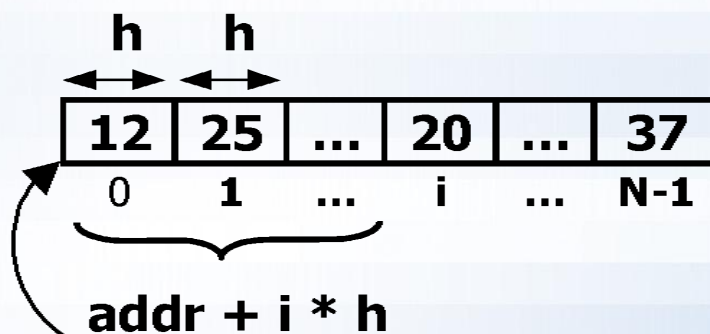
- Гетерогенным называется массив, элементы которого могут содержать значения, относящиеся к различным типам данных.
 - Гетерогенные массивы удобны как универсальная структура для хранения наборов данных произвольных типов, но требуют усложнения механизма поддержки массивов в трансляторе языка.

Материал из Википедии — свободной энциклопедии

Базовые структуры данных – массивы (продолжение)

• Достоинства массивов

- легкость вычисления адреса элемента по его индексу (поскольку элементы массива располагаются один за другим)



- одинаковое время доступа ко всем элементам
- малый размер элементов: они состоят только из информационного поля

Базовые структуры данных – массивы (продолжение)

● Недостатки массивов

- для статического массива
 - отсутствие динамики — невозможность удаления или добавления элемента без сдвига других
- для динамического и/или гетерогенного массива
 - более низкое (по сравнению с обычным статическим) быстродействие
 - дополнительные накладные расходы на поддержку динамических свойств и/или гетерогенности

Материал из Википедии — свободной энциклопедии

Базовые структуры данных – записи

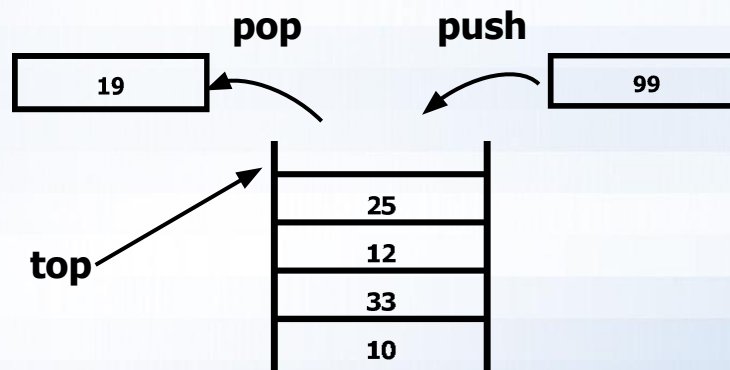
- **Записью (структурой)** называется структура хранения данных, элементы которой могут содержать значения, относящиеся к различным типам данных.
- Записи удобны для хранения наборов данных произвольных типов
 - например, анкета сотрудника, квитанция и т.п.

Фамилия И.О.	Name	Иванов И.И.
Дата рождения	Birthday	1.04.1980
Должность	Position	Менеджер
Оклад	Salary	25000

- Элементы записи называются **ПОЛЯМИ** и им присваиваются уникальные (в пределах записи) имена
- Имена используются в операциях доступа к значениям полей
- Записи и их наборы (например, массивы) в языках программирования предоставляют удобный способ работы с реляционными базами данных

Динамические структуры данных. Стек.

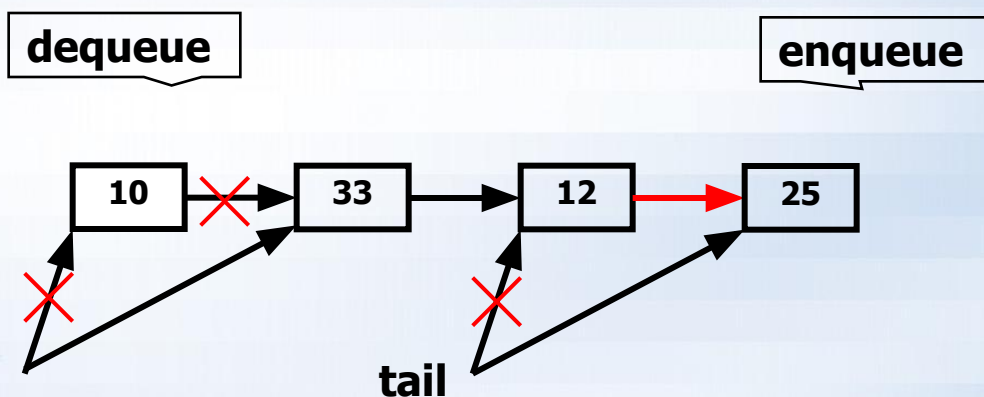
- **Стек** (англ. *stack* = стопка) — структура хранения данных с дисциплиной доступа к элементам **LIFO** (*Last In — First Out*, «последним пришёл — первым вышел»).
- Операцию добавления элемента на вершину (**top**) стека называют **push**, извлечения — **pop**.



- Стек широко используется в программировании и является неотъемлемой частью архитектуры современных процессоров.
- Компиляторы языков программирования высокого уровня используют стек для передачи параметров при вызове подпрограмм, процессоры — для хранения адреса возврата из подпрограмм.

Динамические структуры данных. Очередь.

- **Очередь** (англ. *queue*) — структура хранения данных с дисциплиной доступа к элементам **FIFO** (*First In — First Out*, "первый пришел - первый вышел").
 - Добавление элемента возможно лишь в конец (**tail**) очереди, выборка - только из начала (**head**) очереди, при этом выбранный элемент из очереди удаляется.
 - Операцию добавление элемента называют **enqueue**, выборки - **dequeue**.



- Очередь широко используется в программировании для синхронизации процессов обработки (например, сообщений), моделирования систем массового обслуживания и т.д.

Динамические структуры данных. Список.

- **Связанный список** — динамическая структура хранения данных, каждый элемент которой состоит из:
 - информационного поля (содержит значение элемента)
 - одного (*односвязный*) или двух (*двусвязный*) указателей на соседние элементы.

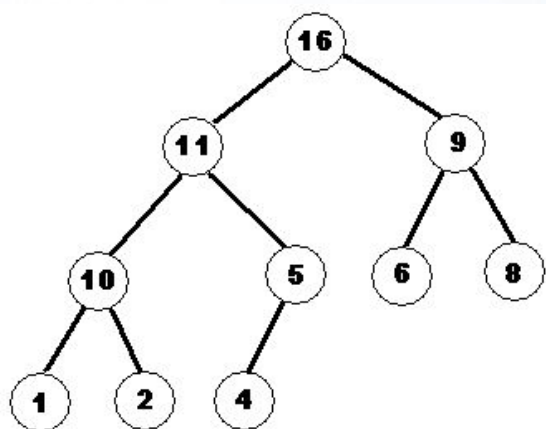


- Список может быть *сортированным* или *несортированным*
- **Достоинства**
 - лёгкость добавления и удаления элементов
 - размер списка ограничен только объемом доступной памяти
- **Недостатки**
 - сложно определить адрес элемента по его индексу (номеру) в списке
 - на поле указателей расходуется дополнительная память (в массивах указатели не нужны)

Материал из Википедии — свободной энциклопедии

Динамические структуры данных. Деревья

- **Двоичное дерево** — абстрактная структура данных, являющееся программной реализацией двоичного дерева (графа).
- Дерево состоит из узлов (записей) вида $(data, l, r)$,
 - где **data** — некоторые данные привязанные к узлу,
 - **l, r** — ссылки на узлы, являющиеся потомками данного узла. Узел **l** называется левым потомком, а узел **r** — правым.



Материал из Википедии — свободной энциклопедии

Основные операции над структурами данных

- Создать (пустую) структуру данных
- Добавить новый элемент
- Удалить заданный элемент
- Проверить структуру на наличие в ней элементов
- Найти элемент(ы) с заданными свойствами
- Извлечь значение заданного элемента
- Получить элемент, следующий (в некотором порядке) за текущим
- Перебрать (в некотором порядке) все элементы структуры данных
- Сортировать (в некотором порядке) все элементы структуры данных
- Удалить всю структуру данных

Информационная система. Понятие базы данных

- **Информационная система (ИС)** — это система, в которой присутствуют *информационные процессы* :
 - хранение,
 - передача,
 - преобразование информации.
- **Некоторые примеры информационных систем**
 - АСУ — Автоматизированные системы управления
 - АСУ П — Автоматизированные системы управления предприятия
 - АСУ ТП — Автоматизированные системы управления технологическими процессами
 - ИИС — Информационно-измерительные системы
 - ИПС — Информационно-поисковые системы
 - САПР — Системы автоматизации проектной деятельности
 - СИИ — Системы искусственного интеллекта

Материал из Википедии — свободной энциклопедии

Информационная система. Понятие базы данных (продолжение)

- **База данных (БД)** — централизованное хранилище данных, обеспечивающее хранение, доступ, первичную обработку и поиск информации.
- Базы данных разделяются на:
 - Иерархические
 - Сетевые
 - Реляционные
 - Объектно-ориентированные
- В настоящее время наибольшее распространение получили реляционные базы данных
- Сетевые и иерархические базы данных считаются устаревшими
- Объектно-ориентированные пока никак не стандартизированы и не получили широкого распространения.

Материал из Википедии — свободной энциклопедии

Требования пользователей к базам данных

- Полнота информации
- Актуальность (непротиворечивость)
- Целостность
- Сохранность (восстановимость после сбоев)
- Производительность (время отклика)
- Удобство в работе

Материал из Википедии — свободной энциклопедии

Проектирование баз данных. Цели и этапы проектирования

- **Концептуальное проектирование**
смысловое содержание базы данных, исходя из целей ее использования
- **Логическое проектирование**
представление логической организации информации средствами выбранной модели данных
- **Физическое проектирование**
разработка физического размещения данных на внешних носителях с целью оптимизации производительности БД

Материал из Википедии — свободной энциклопедии

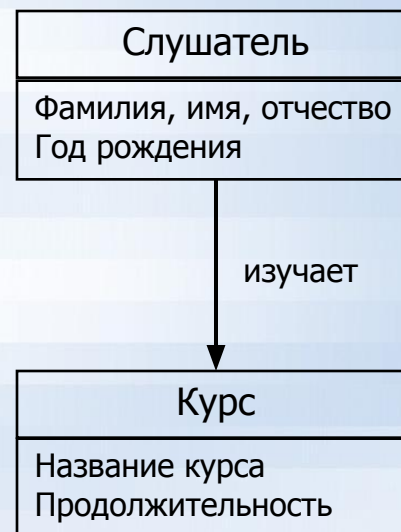
Инфологический аспект. Модель «сущность-связь»

- Инфологическое описание «информация об информации» - характеристики информационных единиц
 - типы данных и области возможных значений (домены)
 - обязательность присутствия и значения по умолчанию
 - и т.п.
- Модель «сущность-связь» (ER-модель):
 - Основные понятия:
 - **Сущности** - объекты
 - **Атрибуты** - характеристики экземпляров сущностей
 - **Связи** между сущностями:
 - Взаимоотношения – целое-часть (фирма – отдел)
 - Взаимодействия (менеджер консультирует клиента)
 - Роли (начальник руководит подчиненным)
 - и т.п.

Материал из Википедии — свободной энциклопедии

Инфологический аспект. Модель «сущность-связь» (продолжение)

- Пример ER-модели:
 - Сущность (объект) – *слушатель*
 - Атрибуты:
 - *Фамилия, имя, отчество*
 - *Год рождения*
 - Сущность – *курс*
 - Атрибуты:
 - *Название курса*
 - *Продолжительность*
 - Связь – слушатель *изучает* курс



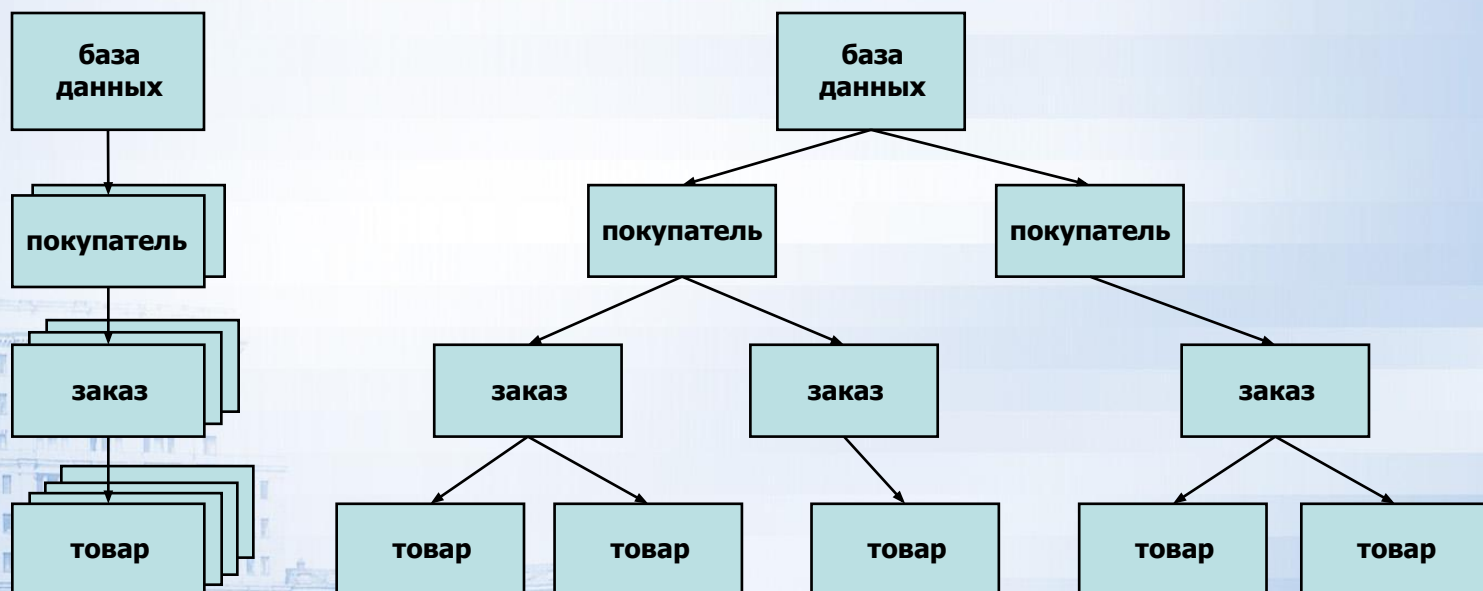
Даталогический аспект. Модели данных (иерархическая, сетевая, реляционная), их достоинства и недостатки

- *Иерархическая модель* базы данных состоит из файлов с указателями от родительских файлов к потомкам, соединяя вместе связанную информацию.
- На схеме иерархического дерева узлы представляются вершинами графа. Каждый узел на более низком уровне связан только с одним узлом, находящимся на более высоком уровне.
- К каждой записи базы данных существует только один (иерархический) путь от корневой записи.
- Например, если иерархическая база данных содержит информацию о покупателях и их заказах, то будет существовать файл **«покупатель»** (родитель) и файл **«заказ»** (дочерний).
- Файл **«покупатель»** будет иметь указатели от каждого заказчика к физическому расположению заказов покупателя в файле **«заказ»**.

Материал из Википедии — свободной энциклопедии

Даталогический аспект. Модели данных (иерархическая, сетевая, реляционная), их достоинства и недостатки (продолжение)

- В этой модели запрос, направленный вниз по иерархии, прост
 - например: какие заказы принадлежат этому покупателю;
- Однако запрос, направленный вверх по иерархии, более сложен
 - например, какой покупатель поместил этот заказ или заказал данный товар.



- Также, трудно представить не-иерархические данные при использовании этой модели.

Даталогический аспект. Модели данных (иерархическая, сетевая, реляционная), их достоинства и недостатки (продолжение)

- Сетевые базы данных подобны иерархическим, за исключением того, что в них имеются указатели в обоих направлениях, которые соединяют родственную информацию.



- Хотя эта модель решает некоторые проблемы, связанные с иерархической моделью, выполнение запросов остается достаточно сложным процессом.
- Так как логика процедуры выборки данных зависит от физической организации данных, то эта модель не является полностью независимой от приложения. Другими словами, если необходимо изменить структуру данных, то нужно изменить и приложение

Материал из Википедии — свободной энциклопедии

Даталогический аспект. Модели данных (иерархическая, сетевая, реляционная), их достоинства и недостатки (продолжение)

- В реляционных базах данных все данные представлены в виде простых *таблиц*, разбитых на строки и столбцы, на пересечении которых расположены данные.
- У каждого *столбца* есть своё имя, которое служит его названием, и все значения в одном столбце имеют один тип.
- *Строки* в реляционной базе данных неупорядочены - упорядочивание производится в момент формирования ответа на запрос.
- *Запросы* к таким таблицам возвращают таблицы, которые сами могут становиться предметом дальнейших запросов.
- Каждая база данных может включать несколько таблиц, которые, как правило, связаны с друг с другом, откуда и произошло название *реляционные*
- Общепринятым стандартом языка работы с реляционными базами данных является язык **SQL**

Материал из Википедии — свободной энциклопедии

Реляционные базы данных. Понятие отношения. Нормализация

- Понятие *нормальной формы* было введено Эдгаром Коддом при создании реляционной модели БД.
- Основное назначение нормальных форм — приведение структуры базы данных к виду, обеспечивающему минимальную *избыточность*.
- Устранение избыточности производится за счёт *декомпозиции* отношений (таблиц) таким образом, чтобы свести к минимуму функциональные зависимости между их атрибутами (полями).

Материал из Википедии — свободной энциклопедии

Реляционные базы данных. Понятие отношения. Нормализация (продолжение)

- Заказы (пример)
 - Фирма принимает от клиентов заказы на поставку товаров.
 - Для обслуживания заказов к каждому заказу прикрепляется торговый агент.
 - Учет заказов ведется в журнале в следующей таблице (бумажной форме)

№ заказа	Дата	Клиент	Адрес	Товар	Цена	Количество	Агент	Телефон
1	1.04.06	РиК	Черноморск	Пила	800	5	Ляпкин	1112233
				Топор	450	8		
2	3.04.06	Нимфа	Энск	Доска	200	50	Тяпкин	1112323

Реляционные базы данных. Понятие отношения. Нормализация (продолжение)

- Заказы
 - Для ввода в компьютер в ячейке таблицы не должно содержаться более одного значения
 - Но, при наличии пустых ячеек, к такой таблице нельзя применять операции сортировки и выборки строк.

№ заказа	Дата	Клиент	Адрес	Товар	Цена	Количество	Агент	Телефон
1	1.04.06	РиК	Черноморск	Пила	800	5	Ляпкин	1112233
				Топор	450	8		
2	3.04.06	Нимфа	Энск	Доска	200	50	Тяпкин	1112323

Реляционные базы данных. Понятие отношения. Нормализация (продолжение)

- Заказы

№ заказа	Дата	Клиент	Адрес	Товар	Цена	Количество	Агент	Телефон
1	1.04.06	РиК	Черноморск	Пила	800	5	Ляпкин	1112233
1	1.04.06	РиК	Черноморск	Топор	450	8	Ляпкин	1112233
2	3.04.06	Нимфа	Энск	Доска	200	50	Тяпкин	1112323

- 1-я нормальная форма

1НФ: Без множественных полей (плоская таблица) + первичный ключ

Заказы (№ заказа, Дата, Клиент, Адрес, Товар, Цена, Количество, Агент, Телефон)

Реляционные базы данных. Понятие отношения. Нормализация (продолжение)

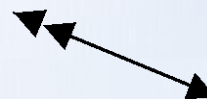
- 2-я нормальная форма

2НФ: 1 НФ + Неключевые реквизиты зависят от всего первичного ключа (а не от его части).

Заказы (№ заказа, Дата, Клиент, Адрес, Агент, Телефон)



Перечень (№ заказа, Товар, Количество)



Товары (Товар, Цена)

Реляционные базы данных. Понятие отношения. Нормализация (продолжение)

- 3-я нормальная форма

3 НФ: 2 НФ + Нет зависимостей между неключевыми атрибутами.

Клиенты (Клиент, Адрес)

Агенты (Агент, Телефон)

Заказы (№ заказа, Дата, Клиент, Агент)

Перечень (№ заказа, Товар, Количество)

Товары (Товар, Цена)

Системы управления базами данных

- **Система управления базами данных (СУБД)** — специализированная программа (чаще комплекс программ), предназначенная для манипулирования базой данных.

Для создания и управления информационной системой СУБД необходима в той же степени, как для разработки программы на алгоритмическом языке необходим транслятор.

- Основные функции СУБД:
 - управление данными во внешней памяти (на дисках);
 - управление данными в оперативной памяти;
 - журнализация изменений и восстановление базы данных после сбоев;
 - поддержание языков БД (язык определения данных, язык манипулирования данными).

Материал из Википедии — свободной энциклопедии

Базы данных и компьютерные сети. Сетевые и распределённые базы данных

- Сетевые БД централизованно хранят данные на одном из узлов сети – **сервере данных** - поддерживают сетевые соединения с **клиентами**
 - такая архитектура называется архитектурой «клиент - сервер»
- Распределенные БД хранят данные на нескольких узлах сети
 - в этом случае важную роль приобретает вопрос рационального дублирования данных по узлам сети - **репликация данных**

Материал из Википедии — свободной энциклопедии