АЛГОРИТИЗАЦИЯ

***1.*** Алгоритм - это конечная последовательность инструкций, предназначенных для выполнения определенной задачи. Основные свойства алгоритма: корректность (правильность решения задачи), определенность (каждый шаг алгоритма должен быть однозначно определен), конечность (алгоритм должен завершаться за конечное количество шагов), эффективность (алгоритм должен быть выполнен за разумное время).

Линейный алгоритм - это алгоритм, который выполняется последовательно, каждый следующий шаг начинается после завершения предыдущего. Пример линейного алгоритма: вычисление суммы двух чисел.

Разветвляющийся алгоритм - это алгоритм, в котором на определенном этапе выбирается один из нескольких вариантов действий в зависимости от условий. Полная ветка - это ветвь, в которой предусмотрено выполнение действий при любых условиях. Неполная ветка - это ветвь, в которой предусмотрены действия только при выполнении определенных условий.

Циклический алгоритм - это алгоритм, который выполняется повторно до тех пор, пока не будет выполнено определенное условие. Цикл - это конструкция, которая позволяет многократно выполнять определенный блок кода. Вложенный цикл - это цикл, который находится внутри другого цикла. Тело цикла - это блок кода, который выполняется в каждой итерации цикла.

Языки программирования могут быть классифицированы по различным признакам, например, по типу системы счисления (двоичные, десятичные и т.д.), по типу применяемых алгоритмов (процедурные, объектно-ориентированные и т.д.), по типу применяемых парадигм (структурное, функциональное и т.д.), по типу сферы применения (научные, бизнес-ориентированные и т.д.).

Компилятор - это программа, которая преобразует исходный код на языке программирования в машинный код. Интерпретатор - это программа, которая выполняет исходный код на языке программирования построчно. Программа - это набор инструкций, написанных на языке программирования, который выполняется компьютером. Программный продукт - это совокупность программных компонентов, которые решают определ

***2.*** Циклический алгоритм - это алгоритм, который выполняется повторно до тех пор, пока не будет выполнено определенное условие. Цикл - это конструкция, которая позволяет многократно выполнять определенный блок кода. Вложенный цикл - это цикл, который находится внутри другого цикла. Тело цикла - это блок кода, который выполняется в каждой итерации цикла.

***3.***Языки программирования могут быть классифицированы по различным признакам, например, по типу системы счисления (двоичные, десятичные и т.д.), по типу применяемых алгоритмов (процедурные, объектно-ориентированные и т.д.), по типу применяемых парадигм (структурное, функциональное и т.д.), по типу сферы применения (научные, бизнес-ориентированные и т.д.).

***4.*** Компилятор - это программа, которая преобразует исходный код на языке программирования в машинный код. Интерпретатор - это программа, которая выполняет исходный код на языке программирования построчно. Программа - это набор инструкций, написанных на языке программирования, который выполняется компьютером. Программный продукт - это совокупность программных компонентов, которые решают определ

***5.*** Типы данных - это различные категории значений, которые могут быть обработаны компьютером. Некоторые из наиболее распространенных типов данных включают целые числа, числа с плавающей точкой, логические значения, символы и строки. Объявление объектов данных - это процесс создания переменных, которые будут использоваться в программе для хранения значений. Внутреннее представление данных в памяти компьютера зависит от типа данных и может различаться в разных языках программирования.

***6.*** Начало

Ввод числа a

Ввод числа b

Если a > b, то

Вывод "a больше b"

Иначе, если a < b, то

Вывод "a меньше b"

Иначе

Вывод "a равно b"

Конец

***7.***Структурированные типы данных - это типы данных, которые могут содержать более одного значения, такие как массивы, структуры и списки. Операторы языка программирования - это инструкции, которые используются для выполнения операций, таких как арифметические вычисления, ввод и вывод данных, управление потоком выполнения программы и т.д. Синтаксис языка - это правила, которые определяют, какие команды и выражения могут быть использованы в языке программирования. Арифметические выражения - это выражения, которые содержат математические операции, такие как сложение, вычитание, умножение и деление. Стандартные функции - это функции, которые предоставляются языком программирования и могут быть использованы для выполнения определенных операций. Структура программы - это организация кода программы в блоки, которые выполняют определенные задачи. Ввод и вывод данных - это операции, которые используются для получения данных от пользователя и вывода данных на экран или в файл. Оператор присваивания - это оператор, который используется для присвоения значения переменной. Условный оператор - это оператор, который выполняет определенный блок кода в зависимости от выполнения определенного условия. Составной оператор - это оператор, который позволяет объединить несколько команд в один блок кода.

***8.* Условный** **оператор** или **оператор** ветвления - **это** **оператор**, конструкция языка программирования, обеспечивающая выполнение определённой команды (набора команд) только при условии истинности некоторого логического выражения, либо выполнение одной из нескольких команд (наборов команд) в зависимости от значения некоторого выражения.

***9.*** Операторы цикла - это инструкции, которые используются для повторения блока кода определенное количество раз или до выполнения определенного условия. Цикл с постусловием - это цикл, который сначала выполняет блок кода, а затем проверяет условие для продолжения цикла. Цикл с предусловием - это цикл, который сначала проверяет условие для продолжения цикла, а затем выполняет блок кода. Цикл с параметром - это цикл, который использует счетчик для определения количества повторений цикла. Вложенные циклы - это циклы, которые находятся внутри других циклов.

***10***.  
1. Цикл с постусловием (do-while) - в Python этот тип цикла не используется.  
2. Цикл с предусловием (while) - в Python можно использовать цикл while, в котором проверка условия происходит перед выполнением тела цикла. Пример:  
while условие:  
[#выполняем](https://vk.com/im?sel=184661574&st=%23%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D1%8F%D0%B5%D0%BC) тело цикла  
```  
3. Цикл с параметром (for) - в Python также присутствует цикл for, который позволяет перебрать все элементы из коллекции, используя индекс или без него. Пример:  
for i in коллекция:  
[#выполняем](https://vk.com/im?sel=184661574&st=%23%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D1%8F%D0%B5%D0%BC) тело цикла  
```  
Использование цикла с параметром без использования индекса:  
for элемент in коллекция:  
[#выполняем](https://vk.com/im?sel=184661574&st=%23%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D1%8F%D0%B5%D0%BC) тело цикла  
```  
4. Вложенные циклы - в Python также можно использовать вложенные циклы, один или несколько циклов находятся внутри других циклов. Пример:  
for i in range(5):  
for j in range(5):  
[#выполняем](https://vk.com/im?sel=184661574&st=%23%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D1%8F%D0%B5%D0%BC) тело вложенного цикла  
Также в Python есть цикл for с использованием индекса:  
for i in range(начало, конец, шаг):  
[#выполняем](https://vk.com/im?sel=184661574&st=%23%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D1%8F%D0%B5%D0%BC) тело цикла  
```

***11.***

- Python подпрограммы реализуются в виде функций и процедур. Функция - это блок кода, который может принимать входные параметры и возвращать результат своей работы. Процедура - это блок кода, который может принимать входные параметры, но не возвращает результата своей работы. В Python, функции и процедуры определяются с помощью ключевых слов def.  
  
-Для вызова функции или процедуры в Python используется имя функции, после которого в круглых скобках перечисляются входные параметры.   
  
-Стандартный механизм передачи параметров в Python осуществляется по значению (pass by value). Это означает, что при передаче параметра функции создается новая локальная переменная, которая копирует значение из переданного аргумента. Механизм передачи параметров включает позиционные параметры, именованные параметры и параметры по умолчанию.  
-Область видимости переменной в Python - это место, где переменная создается и используется. В Python существует три уровня области видимости: локальная область видимости (создается внутри функции или процедуры), глобальная область видимости (создается вне функции или процедуры, и доступна во всем коде), а также область видимости встроенных функций (встроенные функции Python).  
  
-Время жизни переменной в Python зависит от ее места создания. Локальная переменная существует только во время выполнения функции, в то время как глобальная переменная существует до завершения выполнения программы.

***12.***

* Структуризация в программировании - это процесс разделения программы на логические блоки и подпрограммы с целью упростить ее понимание и обеспечить удобное сопровождение и модификацию.
* Основы структурного программирования в Python - это подход к написанию программного кода, который предполагает разбиение задачи на более мелкие блоки, логическое управление выполнением программы и использование логических структур, таких как условные операторы и циклы.
* Основные методы структурного программирования включают:  
    
  1. «Разделение на блоки» - декомпозиция главной задачи на более мелкие подзадачи. Это помогает сделать программу более логически понятной, а также поддерживаемой и модифицируемой в будущем.  
    
  2. «Абстракция» - выделение существенных свойств и параметров системы, исключая несущественные детали.  
    
  3. «Структурирование данных» - это принцип, по которому переменные объединяются в определенную структуру данных, такую как массив, список, стек или очередь.  
    
  4. «Логическое управление» - это принцип, по которому контроль за выполнением программы осуществляется через использование логических структур, таких как условные операторы, циклы и подпрограммы.  
    
  5. «Дефенсивное программирование и декомпозиция на простые значения» - это методы, призванные уменьшить количество ошибок в процессе написания программ. Они включают щедрое резервирование памяти, контроль граничных значений и прочие.  
    
  6. «Модульное программирование» - это методика разбиения программы на логические блоки, минимизируя исходный код отслеживаемых багов, и отдельной возможности менять эти блоки.  
    
  7. «Стандартизация кода» - это принцип, по которому использование однотипных имен функций, переменных и констант не только упрощает понимание программы, но и облегчает ее поддержку.  
    
  Все эти методы структурного программирования помогают создать программное обеспечение со структурой и логикой, что упрощает его понимание, поддержку и модификацию.

***13.***

Модульное программирование - это подход к программированию, основанный на создании небольших независимых единиц (модулей) и их последующем объединении в единую программу.  
  
  
Модуль - это некоторая логически завершенная часть программы, реализующая определенную функциональность. Он может включать в себя объявления переменных, функций, классов и других элементов программы, а также спецификации импортируемых модулей.

Структура модуля может варьироваться в зависимости от используемого языка программирования. Однако, в целом, модуль обычно включает в себя:  
- Заголовочную часть (например, с комментарием, автором, датой создания и т.д.)  
- Объявления импортируемых модулей  
- Определения переменных и функций  
- Логику работы программы  
  
Компиляция - это процесс преобразования исходного кода программы в машинный код, который может быть выполнен на компьютере.  
  
Компоновка - это процесс объединения скомпилированных модулей в единую программу.  
Стандартные модули - это набор модулей, которые поставляются с языком программирования и предоставляют программисту ряд полезных функций для работы с файловой системой, сетью, форматирования вывода и др. В многих случаях, использование стандартных модулей позволяет значительно ускорить процесс разработки программы.

***14.***

Структуры данных - это способ организации и хранения данных в программе. Они могут использоваться для эффективного хранения, изменения и поиска данных в программе.  
  
В языках программирования существует множество структур данных, некоторые из которых:  
- Массивы  
- Строки  
- Множества  
- Записи  
- Файлы  
  
Массив - это упорядоченная коллекция элементов одного типа. Каждый элемент массива может быть обращен по индексу, который указывает на его позицию в массиве. Индекс является целым числом и начинается с 0.  
  
Особенности программирования с массивами  
  
Написание программы с массивами включает в себя следующие шаги:  
  
1. Объявление массива - указание типа элементов массива и его размерности  
2. Инициализация массива - присвоение начальных значений элементам массива  
3. Обращение к элементам массива - получение значения элемента по его индексу  
4. Изменение элементов массива - присваивание новых значений элементам массива  
5. Проход по массиву - обработка всех элементов массива  
  
Основным преимуществом использования массивов является простота и быстрота доступа к элементам массива по индексу. Однако, следует учитывать, что размерность массива должна быть известна заранее, что может ограничить возможности программы.  
  
Пример написания программы с массивами на языке Python:  
  
# Объявление и инициализация массива  
my\_array = [1, 2, 3, 4, 5]  
  
# Обращение к элементам массива  
print(my\_array[0]) # выведет 1  
print(my\_array[2]) # выведет 3  
  
# Изменение элементов массива  
my\_array[0] = 10  
print(my\_array[0]) # выведет 10  
  
# Проход по массиву  
for element in my\_array:  
print(element) # выведет каждый элемент массива по одному

***15.*** Символьный и строковый типы - это типы данных, которые используются для хранения символов и строк. Объявление типов - это определение типа переменной при ее создании. Поиск, удаление, замена и добавление символов в строке - это операции, которые выполняются над строками для изменения их содержимого. Операции со строками - это операции, которые выполняются над строками, такие как конкатенация (соединение) строк, извлечение подстроки и т.д. Стандартные функции и процедуры для работы со строками - это функции и процедуры, которые предоставляются языком программирования и могут быть использованы для выполнения операций со строками, таких как поиск, замена и т.д.

***16.*** Понятие множества - это коллекция уникальных элементов без определенного порядка. Объявление множества - это создание переменной типа множество. Операции над множествами - это операции, которые выполняются над множествами, такие как объединение, пересечение и т.д.

***17.*** Типы файлов - это способ организации данных на диске компьютера. Организация доступа к файлам - это определение прав доступа к файлам на уровне операционной системы. Открытие и закрытие файла - это операции, которые выполняются для установления и завершения связи между программой и файлом. Запись в файл и чтение из файла - это операции, которые выполняются для записи данных в файл или чтения данных из файла.

***18.*** Объективно-ориентированное программирование - это методология программирования, которая использует понятие объекта для организации кода. Базовые понятия ООП: объект - это экземпляр класса, его свойства - это атрибуты объекта, а методы - это процедуры, которые могут быть вызваны для выполнения определенных действий. Класс - это шаблон, на основе которого создаются объекты.

***19.*** Основные принципы ООП - это инкапсуляция, наследование и полиморфизм. Инкапсуляция - это механизм, который позволяет скрыть детали реализации объекта и предоставить только интерфейс для работы с ним. Наследование - это механизм, который позволяет создавать новый класс на основе существующего класса. Полиморфизм - это механизм, который позволяет объектам одного класса использовать методы другого класса.

***20.*** Событийно-управляемая модель программирования - это методика программирования, которая использует события для определения поведения программы. Компонентно-ориентированный подход - это методология программирования, которая использует набор компонентов для создания программы. Каждый компонент выполняет определенную функцию и может быть использован в разных программах.

***21***.Интегрированная среда разработки (IDE) - это программное обеспечение, которое предоставляет программистам инструменты для разработки программ. Интерфейс среды разработки - это пользовательский интерфейс, который предоставляет доступ к инструментам IDE. Основные окна IDE - это окно кода проекта, окно свойств проекта, окно консоли, окно отладки и т.д. Инструменты IDE - это инструменты, которые используются для написания, отладки и тестирования кода. Объекты IDE - это объекты, которые представляют элементы программы, такие как окна, кнопки, текстовые поля и т.д. Форма и размещение на ней управляющих элементов зависят от конкретной IDE.

***22.*** Проектирование объектно-ориентированного приложения - это процесс создания плана для разработки программы, используя принципы ООП. Создание интерфейса пользователя - это процесс создания пользовательского интерфейса, который позволяет пользователям взаимодействовать с программой. Программирование приложения - это процесс написания кода для реализации функциональности приложения. Тестирование, откладка приложения - это процесс проверки приложения на наличие ошибок и исправление их. Создание документации - это процесс создания документации, которая описывает функциональность и использование приложения

***23.*** Классы объективно-ориентированного языка программирования - это шаблоны, которые используются для создания объектов. Виды классов - это базовый класс, производный класс и абстрактный класс. Назначение класса - это определение поведения объекта. Свойства класса - это атрибуты объекта. Методы класса - это процедуры, которые могут быть вызваны для выполнения определенных действий. События класса - это сигналы, которые могут быть отправлены другим объектам. Объявление класса - это создание нового класса. Свойства и методы экземпляра класса - это атрибуты и методы, которые принадлежат конкретному объекту класса. Наследование - это механизм, который позволяет создавать новый класс на основе существующего класса.

***24.*** Основные компоненты (элементы управления) интегрированной среды разработки - это кнопки, поля ввода текста, списки, деревья, вкладки и т.д. Эти элементы управления используются для создания пользовательского интерфейса приложения. Дополнительные элементы управления - это элементы управления, которые не входят в стандартный набор элементов управления, но могут быть добавлены с помощью расширений или плагинов. Свойства компонентов (элементов управления) - это параметры, которые определяют внешний вид и поведение элементов управления. Виды свойств - это свойства, которые определяют внешний вид элемента управления, и свойства, которые определяют его поведение.

***25.*** Синтаксис определения свойств - это способ определения значения свойства для элемента управления. Категория свойств - это группировка свойств по их назначению. Назначение свойств - это определение внешнего вида и поведения элемента управления. Свойства могут влиять на результат работы программы, например, изменение свойства текстового поля может привести к изменению отображаемой в нем информации. Управление объектом через свойства - это способ изменения значения свойства элемента управления для достижения желаемого результата.

***26.*** Разработка оконного приложения - это процесс создания программы, которая имеет графический интерфейс пользователя. Разработка функционального интерфейса приложения - это процесс создания интерфейса, который позволяет пользователям взаимодействовать с программой. Создание интерфейса приложения - это процесс создания пользовательского интерфейса, который включает в себя различные элементы управления, такие как кнопки, поля ввода текста, меню и т.д. Цель создания интерфейса приложения - упрощение работы с программой для пользователя.

***27.*** Разработка функциональной схемы работы приложения - это процесс создания плана для реализации функциональности приложения, определения основных блоков их взаимосвязей. Создание процедур обработки событий - это процесс написания кода, который будет выполняться при возникновении определенных событий, например, нажатия кнопки или выбора элемента из списка. Компиляция - это процесс преобразования исходного кода в машинный код, который может быть исполнен на компьютере. Запуск приложения - это процесс запуска скомпилированного приложения на компьютере пользователя.

БАЗЫ ДАННЫХ

***1.*** *Система баз данных* - это компьютеризированная система хранения данных, основная цель,  которой содержать информацию и предоставлять её по требованию.

*Система управления базами данных (СУБД)*- программное обеспечение, предназначенное  для использования и (или) модификации этих данных одним или несколькими лицами

В системе баз данных выделяют четыре основных компонента:

* данные;
* аппаратное обеспечение;
* программное обеспечение;
* пользователи.

***2.*** В истории развития баз данных можно выделить следующие **этапы**:

1. Файлы и файловые системы.

2. Базы данных на больших ЭВМ. Первые СУБД.

3. Эпоха персональных компьютеров. Настольные СУБД.

4. Распределенные базы данных.

Согласно данным ряда аналитических отчетов, лидерами рынка СУБД, предназначенных для OLTP-систем, на данный момент являются компании **IBM, Oracle, Microsoft и Sybase**, а на рынке СУБД, предназначенных для создания хранилищ данных, — IBM, Oracle, Microsoft, Sybase и Teradata

***3.***Технология работы с базами данных включает в себя несколько этапов:

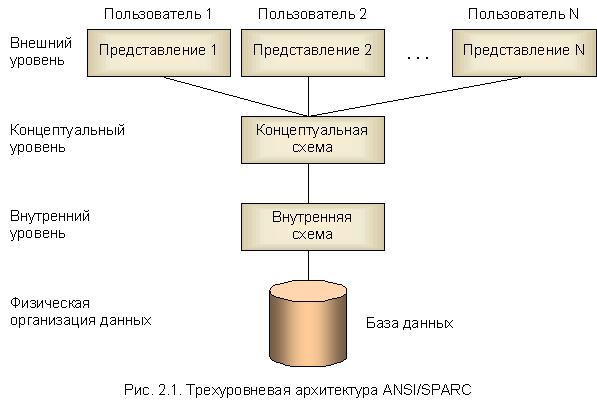
1) построение инфологической модели базы (определение назначения и цели создания БД);

2) создание структуры таблиц базы данных (определение необходимых таблиц и их связей, полей, задание ключа);

3) ввод, поиск и обработка данных, содержащихся в таблицах (создание форм, фильтров и запросов);

4) вывод информации из базы данных (создание отчетов).

***4.***Трехуровневая архитектура ANSI/SPARC имеет внешний, концептуальный и внутренний уровни, как показано на рис. 2.1. Цель трехуровневой архитектуры заключается в отделении пользовательского представления базы данных от ее физического представления, что обусловлено рядом причин:

* каждый пользователь должен иметь возможность обращаться к одним и тем же данным, используя свое представление о них, и изменять свое представление о данных, не оказывая при этом влияние на других пользователей;
* пользователи не должны знать особенности физического хранения данных в базе;
* АБД должен иметь возможность изменять структуру хранения данных в базе, не оказывая влияние на пользовательские представления;
* АБД должен иметь возможность изменять концептуальную или глобальную структуру базы данных без влияния на всех пользователей. 

***5.*Модель данных** - это совокупность структур данных и операций их обработки. Рассмотрим три основных типа моделей данных: иерархическую, сетевую и реляционную.

**Иерархическая модель** представляет собой совокупность элементов, расположенных в порядке их подчинения от общего к частному и образующих перевернутое по структуре дерево (граф).

К основным понятиям иерархической структуры относятся уровень, узел и связь. **Узел** - это совокупность атрибутов данных, описывающих некоторый объект.

**В сетевой структуре** при тех же основных понятиях (уровень, узел, связь) каждый элемент может быть связан с любым другим элементом.

**Реляционная модель** данных объекты и связи между ними представляет в виде таблиц, при этом связи тоже рассматриваются как объекты. Все строки, составляющие таблицу в реляционной базе данных, должны иметь первичный ключ. Все современные средства СУБД поддерживают реляционную модель данных.

Эта модель характеризуются простотой структуры данных, удобным для пользователя табличным представлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для обработки данных.

Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

1. Каждый элемент таблицы соответствует одному элементу данных.

2. Все столбцы в таблице однородные, т.е. все элементы в столбце имеют одинаковый тип и длину.

3. Каждый столбец имеет уникальное имя.

4. Одинаковые строки в таблице отсутствуют;

5. Порядок следования строк и столбцов может быть произвольным.

***6.***Реляционная модель представляет собой **совокупность данных, состоящую из набора двумерных таблиц**. В теории множеств таблице соответствует термин отношение (), физическим представлением которого является таблица, отсюда и название модели– реляционная. Реляционная модель является удобной и наиболее привычной формой представления данных. При табличной организации данных отсутствует иерархия элементов.

Основными понятиями, с помощью которых определяется реляционная модель, являются следующие:

1. ***реляционная БД*** – набор нормализованных отношений;
2. ***отношение*** – файл, плоская таблица, состоящая из столбцов и строк; таблица, в которой каждое поле является атомарным;
3. ***домен***– совокупность допустимых значений, из которой берется значение соответствующего атрибута определенного отношения. С точки зрения программирования, ***домен*** – это тип данных;
4. ***универсум***– совокупность значений всех полей или совокупность доменов;
5. ***кортеж***– запись, строка таблицы;
6. ***кардинальность -***количество строк в таблице;
7. ***атрибуты* –**поименованныеполя, столбцы таблицы;
8. ***степень отношения*** - количество полей (столбцов);
9. ***схема отношения***– упорядоченный список имен атрибутов;
10. ***схема реляционной БД***– совокупность схем отношений;
11. ***первичный ключ***– уникальный идентификатор с неповторяющимися записями – столбец или некоторое подмножество столбцов, которые единственным образом определяют строки.

***7.***Допустимые информационные структуры

1. Реляционная структура  
   2. Иерархическая структура  
   3. Сетевая структура  
   4. Объектно-ориентированная структура  
   5. XML-структура  
   6. NoSQL структура  
   7. Графовая структура  
   8. Текстовая структура (например, CSV-файлы)
2. Ограничение целостности данных - это набор правил, которые определяют допустимые значения и отношения в базе данных, чтобы поддержать ее целостность. Все РСУБД позволяют определить разнообразные ограничения целостности, которые позволяют ограничить допустимые значения для атрибутов и связей разных таблиц базы данных. Среди знаменитых ограничений целостности данных можно выделить следующие:  
     
   - Ограничение NOT NULL (не пустое) - определяет, что значение определенного атрибута не может быть NULL (т.е., отсутствовать или пустым).  
   - Ограничение UNIQUE (уникальное) - требуется, чтобы значение в определенном атрибуте было уникальным (то есть, не повторялось более одного раза в базе данных).  
   - Ограничение PRIMARY KEY (первичный ключ) - определяет, что определенный атрибут (или набор атрибутов) в таблице базы данных будет служить первичным ключом, т.е. гарантирует, что каждая запись имеет уникальный ключ.  
   - Ограничение FOREIGN KEY (внешний ключ) - определяет связи между двумя таблицами базы данных, связывая таблицы между собой по значениям её атрибутов.  
   - Ограничение CHECK (проверка) - ограничение CHECK определяет условия, или правила, которые должны присутствовать в каждой записи в таблице. Например, ограничение CHECK может определять, что определенный атрибут должен находиться в определенном диапазоне значений, чтобы быть допустимым.  
     
   Объединение всех ограничений целостности данных помогает обеспечить корректность и надежность данных в базе данных.

***8.*** Проектирование базы данных осуществляется в три этапа:

-концептуальное проектирование

-логическое проектирование

-физическое проектирование

***9.*** Основные этапы проектирования БД включают:

- Анализ бизнес-требований и определение целей БД.

- Создание концептуальной модели данных, описывающей предметную область и основные объекты данных.

- Разработка информационно-логической модели данных, которая описывает структуру БД и связи между таблицами.

- Создание физической модели данных, которая определяет способы хранения данных (типы таблиц, индексы, ключи, ограничения и т. д.).

- Разработка программного интерфейса БД и создание нужных объектов данных (таблиц, представлений, индексов и т. д.).

- Тестирование и оптимизация БД.

При проектировании базы данных необходимо также учитывать функциональные зависимости между реквизитами - отношения, когда значение одного реквизита зависит от значения другого реквизита, или группы реквизитов.

***10.*** Концептуальное проектирование БД - это первый этап проектирования базы данных, на котором определяются основные сущности, объекты и их отношения в предметной области, которые необходимы для будущих пользователей БД. На этом этапе не рассматриваются детали ее реализации в физической структуре, а представляется базовая структура, которая отражает цели и требования бизнеса. Результатом будет концептуальная модель данных.

***11.*** Инфологическое проектирование баз данных (ИЛП) - это процесс разработки модели данных с использованием формальных логических конструкций, описывающих свойства и ограничения данных в базе данных. ИЛП включает в себя разработку схемы данных (диаграммы сущность-связь, ER-диаграммы) и определение всех сущностей, атрибутов, связей и ограничений.

***12.*** Первая нормальная форма (1НФ) - это первый шаг в процессе нормализации реляционных баз данных (РБД). Для того, чтобы РБД находилась в первой нормальной форме, данные в таблицах должны быть атомарными (не разделяемыми) и каждый элемент данных должен находиться в своей собственной ячейке таблицы.

***13.*** Вторая нормальная форма (2НФ) - это второй шаг в процессе нормализации РБД, когда все данные в таблице достигли первой нормальной формы (1НФ), и каждый неключевой атрибут в таблице функционально зависит только от первичного ключа. Третья нормальная форма (3НФ) - это следующий шаг после 2НФ, который устраняет транзитивную функциональную зависимость между атрибутами.

***14.*** Создание таблиц - это процесс определения таблиц и их полей для хранения данных в базе данных. Связывание таблиц - это процесс установления связей между таблицами по ключевым полям, чтобы можно было объединять данные из нескольких таблиц в один запрос. Схема данных - это описание структуры БД, включающее все физические объекты (таблицы, представления, индексы и другие).

***15.*** Создание запросов в базе данных - это процесс извлечения нужных данных из таблиц с помощью SQL запросов. Ниже описаны три типа запросов на выборку данных:

- Запрос на выборку - это запрос, который извлекает данные из указанных таблиц по заданным критериям. Запросы могут содержать условия, сортировку, вычисляемые поля и другие опции.

Пример запроса на выборку:

sql

SELECT \* FROM users WHERE age > 30

Этот запрос выберет все строки из таблицы `users`, где значение в поле `age` больше 30.

- Запрос с параметром - это запрос, который содержит специальное значение, которое заменяется на конкретное значение или значение, введенное пользователем во время выполнения запроса.

Пример запроса с параметром:

sql

SELECT \* FROM users WHERE age > ?

В этом запросе значение вместо знака `?` может быть специальным значением или значением, заданным пользователем при выполнении запроса.

- Запрос с группировкой - это запрос, который группирует данные на основе заданных полей и вычисляет агрегатные функции для каждой группы, такие как `COUNT`, `SUM`, `AVG` и другие.

Пример запроса с группировкой:

sql

SELECT department, COUNT(\*) as number\_of\_employees

FROM employees

GROUP BY department

HAVING COUNT(\*) > 10

Этот запрос группирует данные по полю `department`, подсчитывает количество записей в каждой группе и выбирает только те группы, в которых количество записей больше 10.

В целом, при создании запросов в базе данных необходимо учитывать особенности структуры таблиц, данные которых необходимо извлекать, а также потребности пользователей, которые будут использовать их при работе с базой данных.

* ***16.***

Вот несколько примеров запросов на создание таблиц, удаление и добавление записей в MySQL:  
Создание таблицы:  
CREATE TABLE table\_name (  
column1 datatype1,  
column2 datatype2,  
column3 datatype3,  
.....  
);  
Пример:  
CREATE TABLE employees (  
id INT(11) NOT NULL AUTO\_INCREMENT,  
name VARCHAR(255) NOT NULL,  
age INT(11),  
salary DOUBLE,  
PRIMARY KEY (id)  
);  
Удаление таблицы:  
DROP TABLE table\_name;  
Пример:  
DROP TABLE employees;  
Добавление записи:  
INSERT INTO table\_name (column1, column2, column3,...) VALUES (value1, value2, value3, ...);  
Пример:  
INSERT INTO employees (name, age, salary) VALUES ('John Doe', 30, 50000);  
Обновление записи:  
UPDATE table\_name SET column1 = value1, column2 = value2, ... WHERE some\_column = some\_value;  
Пример:  
UPDATE employees SET salary = 60000 WHERE name = 'John Doe';

***17.***

Вот несколько примеров современных СУБД:  
1. MySQL: это одна из самых популярных СУБД в мире. Она открытая программа с отличной производительностью, быстрой скоростью выполнения запросов и широким спектром функций включая скалярные функции, процедуры, триггеры, встроенные функциитекстового поиска и др. MySQL обычно используется веб-сайтами, приложениями и хостинг-провайдерами.  
2. PostgreSQL: это олертая открытая программа с устойчивым и надежным хранением большого количества данных. PostgreSQL используется в большом количестве приложений, включая географические информационные системы, научно-исследовательские и телекоммуникационные приложения.  
3. Microsoft SQL Server: это коммерческий проект Microsoft, используемый на многих клиентских компьютерах. Microsoft SQL Server выполняется только на платформах Windows и предоставляет возможности по обработке данных от малых до огромных масштабов. Он предлагает масштабируемость, высокую доступность и технологии интеграции, такие как Reporting Services, Analysis Services и Integration Services.  
4. Oracle: это коммерческая программа, которую разработала компания Oracle Corporation. Oracle является одной из лучших СУБД для корпоративной среды, таких как финансовые компании и крупные конкурирующие с исполяющих базы данных. Он предоставляет высокую доступность и надежный контроль над данными.  
5. MongoDB: это нереляционная, открытая программа с высокой производительностью и масштабируемыми базами данных. MongoDB поддерживает хранение неструктурированных данных, таких как документы JSON, и очень гибкий в доступе к хранящейся информации. Его часто используют в веб-разработке и облачной среде.

* ***18.***

Модуль sqlite3 в Python предоставляет возможность работать с базами данных SQLite. SQLite является легковесной СУБД и хранит данные в единственном файле, что упрощает ее использование.  
import sqlite3  
# создание соединения с базой данных  
conn = sqlite3.connect('example.db')  
# создание курсора  
cursor = conn.cursor()  
# создание таблицы  
cursor.execute('''CREATE TABLE person  
(id INTEGER PRIMARY KEY, name TEXT, age INTEGER)''')  
# добавление записи в таблицу  
cursor.execute("INSERT INTO person (name, age) VALUES ('Alice', 25)")  
# добавление нескольких записей в таблицу  
persons = [('Bob', 26), ('Charlie', 27)]  
cursor.executemany("INSERT INTO person (name, age) VALUES (?, ?)", persons)  
# сохранение изменений  
conn.commit()  
# закрытие соединения  
conn.close()  
В этом примере мы создаем базу данных example.db, создаем таблицу person, добавляем записи с помощью метода execute() и метода executemany(), и сохраняем изменения при помощи метода commit().  
Чтобы изменить данные в таблице, мы можем использовать метод UPDATE. Вот пример обновления записи в таблице:  
# обновление записи  
cursor.execute("UPDATE person SET age = 28 WHERE name = 'Bob'")  
# сохранение изменений  
conn.commit()  
# закрытие соединения  
conn.close()  
Этот пример обновляет запись "Bob" в таблице person, изменяя значение поля age.  
Мы также можем использовать метод DELETE, чтобы удалить записи из таблицы. Вот пример удаления записи из таблицы:  
# удаление записи  
cursor.execute("DELETE from person WHERE name='Charlie'")  
# сохранение изменений  
conn.commit()  
# закрытие соединения  
conn.close()  
Этот пример удаляет запись с именем "Charlie" в таблице person..

***19.***

Вот несколько примеров выполнения запросов к базе данных SQLite с использованием модуля sqlite3:  
import sqlite3  
# создание соединения с базой данных  
conn = sqlite3.connect('example.db')  
# создание курсора  
cursor = conn.cursor()  
# выполнение SELECT-запроса  
cursor.execute("SELECT \* FROM person")  
rows = cursor.fetchall()  
# вывод записей на экран  
for row in rows:  
print(row)  
# выполнение INSERT-запроса  
cursor.execute("INSERT INTO person (name, age) VALUES ('David', 30)")  
# выполнение UPDATE-запроса  
cursor.execute("UPDATE person SET age = 29 WHERE name = 'Bob'")  
# выполнение DELETE-запроса  
cursor.execute("DELETE from person WHERE name='Charlie'")  
# сохранение изменений  
conn.commit()  
# закрытие соединения  
conn.close()

***20.***

SQL (Structured Query Language) - язык запросов используется в СУБД для управления данными.  
Вот основные понятия, которые нужно знать:  
- База данных (Database) - набор данных, хранящихся вместе. Она содержит таблицы, индексы, представления и другие объекты базы данных.  
- Таблица (Table) - это коллекция данных, хранящихся в строках и столбцах. Каждая строка в таблице представляет отдельную запись, а каждый столбец таблицы представляет собой отдельное поле или характеристику записи.  
- Столбец (Column) - это имя хранимого в таблице атрибута или поля. В столбцах содержатся фактические данные.  
- Строка (Row) - набор полей для отдельной записи в таблице.  
- Индекс (Index) - структура данных, используемая для ускорения операций поиска в таблице. Он содержит ссылки на строки таблицы, отсортированные по определенным критериям.  
- Ключ (Key) - это столбец или набор столбцов, который идентифицирует уникальную запись в таблице. Таблицы могут иметь один или несколько ключей. Основной ключ является ключом, который уникально идентифицирует каждую запись в таблице.  
- Запрос (Query) - это запрос на извлечение данных из таблицы или таблиц.  
- JOIN - оператор, который позволяет объединить данные из двух и более таблиц на основе указанных условий соединения.  
- Синтаксис SQL - структура языка SQL, которая определяет правила написания команд и запросов в СУБД.  
  
Вот несколько примеров синтаксиса операторов SQL:  
Создание таблицы  
CREATE TABLE table\_name (  
column1 datatype1,  
column2 datatype2,  
column3 datatype3,  
.....  
);  
Выборка данных  
SELECT column1, column2, ... FROM table\_name WHERE condition;  
Вставка данных  
INSERT INTO table\_name (column1, column2, column3,...) VALUES (value1, value2, value3, ...);  
Обновление записи  
UPDATE table\_name SET column1 = value1, column2 = value2, ... WHERE some\_column = some\_value  
Удаление записи  
DELETE FROM table\_name WHERE some\_column = some\_value;  
В SQL также есть разные типы данных, которые могут использоваться в таблицах:  
- INTEGER - целочисленный тип данных  
- REAL - тип данных для чисел с плавающей точкой  
- TEXT - строковый тип данных  
- BLOB - тип данных двоичных объектов  
- NULL - специальный тип данных, используемый для обозначения отсутствующих значений

***22.***

Организация запросов на выборку данных при помощи SQL - это важная часть работы с СУБД. Она позволяет извлекать данные из таблицы(таблиц) по определенным критериям и агрегировать их для дальнейшей обработки и анализа.  
  
Вот несколько примеров запросов на выборку данных при помощи SQL:  
 -Выбор всех столбцов из таблицы:  
SELECT \* FROM table\_name;  
- Выбор конкретных столбцов из таблицы:  
SELECT column1, column2 FROM table\_name;  
- Выбор уникальных значений из столбца:  
SELECT DISTINCT column\_name FROM table\_name;  
- Условные выборки:  
SELECT \* FROM table\_name WHERE condition;  
- Выборка данных из нескольких таблиц:  
SELECT table1.column1, table2.column2 FROM table1 INNER JOIN table2 ON [table1.id](https://vk.com/away.php?to=http%3A%2F%2Ftable1.id&cc_key=) = [table2.id](https://vk.com/away.php?to=http%3A%2F%2Ftable2.id&cc_key=);  
- Агрегирование данных:  
SELECT MAX(column\_name) FROM table\_name;  
SELECT MIN(column\_name) FROM table\_name;  
SELECT SUM(column\_name) FROM table\_name;  
SELECT AVG(column\_name) FROM table\_name;  
- Группировка данных:  
SELECT column1, AVG(column2) FROM table\_name GROUP BY column1;