1.1 Аналогові обчислювальні машини (АВМ) - обчислювальні машини безперервної дії, працюють з інформацією, представленою в безперервній (аналогової) формі, тобто у вигляді безперервного ряду значень будь-якої фізичної величини (найчастіше електричного напруги).

Цифрові обчислювальні машини (ЦВМ) - обчислювальні машини дискретної дії, працюють з інформацією, представленою в дискретної, а точніше, в цифровій формі.

Обчислювальна машина є машиною з архітектурою фон Неймана, якщо:

1.Програма та дані зберігаються в одній загальній пам'яті. Це дає можливість виконувати над командами ті ж дії, що і над даними.

2.Кожна комірка пам'яті машини ідентифікується унікальним номером, який називається адресою.

3.Різні слова інформації (команди та дані) розрізняються за способом використання, але не за способом кодування та структурою представлення в пам'яті.

4.Кожна програма виконується послідовно, починаючи з першої команди, якщо немає спеціальних вказівок. Для зміни цієї послідовності використовуються команди передачі управління

Гарвардська архітектура – те саме тільки команди та данні розділені в пам’яті.

Що вкходить в ПК

1. Керуючий пристрій КП.
2. АЛП.
3. RAM
4. ROM
5. Пристрій введення
6. Пристрій виведення.

Процесор:

Пристрій керування (ПК).Здійснює координацію роботи всіх інших пристроїв.

Арифметико-логічний пристрій(АЛП).Так називається пристрій для цілочислових операцій.

AGU (Address Generation Unit)— пристрій генерації адрес.

Математичний співпроцесор(FPU).Процесор може містити декілька математичних співпроцесорів.

Дешифратор інструкцій(команд).Аналізує інструкції з метою виділення операндів і адрес.

RISC(англ.*restricted (reduced) instruction set computer*[1][2]— компьютер с сокращённымнабором команд)—архитектура процессора, в котором быстродействие увеличивается за счёт упрощенияинструкций, чтобы их декодирование было более простым, а время выполнения— меньшим.

CISC(англ.*Complex Instruction Set Computer*— комп'ютер зі складним набором команд)— цеархітектура системи команд, в якій більшість команд є комплексними, тобто реалізують певний набір простіших інструкційпроцесораабо шляхом зіставлення з кожною CISC-командою певноїмікропрограми, або принаймні можуть бути зведені до набору таких простих інструкцій.

1.2 Для запам'ятовування інформації (символів, цифр, графічних зображень) комп'ютер використовує двійковий код, тобто 0 і 1 через те, що комп'ютер складається з фізичних елементів, які можуть перебувати в двох стійких станах. У комп'ютерній техніці на носій інформації записуються дані в двійковому коді, тобто у вигляді нулів і одиниць.
Двійковий код позначається *бітом*. Слово біт походить від англійських слів Binary Digit (скорочено біт).
***Біт** – це найменша одиниця інформації, яка позначається двіійовим кодом 0, 1. Вона також виражає логічні значення так, ні.*

Число з плаваючою комою складається з:

- Знака мантиси (вказує на негативних чи позитивних числа)
- Мантису (виражає значення числа без урахування порядку)
- Знака порядку
- Порядку (виражає ступінь основи числа, на яке множиться мантиса)

Назва формата SPARC v9	Розрядність	Діапазон
Байт зі знаком	8(7+1)	−2 7…2 7-1
Півслово зі знаком	16(15+1)	−215…2 15-1
Слово зі знаком	32(31+1)	−231…2 31-1
Подвійне слово зі знаком	64(63+1)	−263…2 63-1
Байт без знака	8	0…2 8 -1
Беззнакове півслово	16	0…2 16-1
Беззнакове слово	32	0…2 32-1
Беззнакове подвійне слово	64	

1.3 Паралелізм на рівні команд(ILP)— є мірою того, яка кількість операцій в комп'ютерній програмі може виконуватися одночасно. Потенційне поєднання виконання команд називається паралелізмом на рівні команд.

Є два підходи до паралелізму на рівні команд:

- Апаратне_забезпечення
- Програмне_забезпечення

1. e = a + b 2. f = c + d 3. m = e * f

1,2 – паралельно бо незалежні. 3. не паралельно.

В основі концепції конвеєризації обчислень є твердження про те, що процес обробки машинної команди можна розбити на декілька практично незалежних етапів, які потім можна суміщати в часі для декількох команд в відповідній апаратурі (конвеєрі команд).

Загальноприйнятим в теорії конвеєрних структур є така послідовність етапів:

1. Вибірка (instruction fetch, IF)— завантаження нової команди зпам'яті
2. Декодування (instruction decode, ID)— інтерпретація та відправка команди у відповідний операційний пристрій в залежності від різновиду операції
3. Виконання (execution, EX)— виконання команд та обчислення ефективної адреси пам'яті для результату або операндів, які необхідно завантажити
4. Звертання до пам'яті (memory, MEM)— виконання операцій з пам'яттю (для команд завантаження/збереження)
5. Збереження результату (writeback, WB)— збереження результату обчислень врегістрі

Параметри ефективності:

- Пропускна здатність— максимальна кількість команд, які виконуються за один такт машинного часу (instructions per cycle, IPC)
- Тривалість етапу (стадії)— кількість машинних циклів для виконання одного етапу конвеєрного обчислення (може бути різною для різних етапів)
- Необхідний (максимальний) ступінь паралелізму

Суперскалярність— архітектура обчислювального ядра, що використовує кілька декодерів команд, які можуть навантажувати роботою безліч виконавчих блоків. Планування виконання потоку команд є динамічним і здійснюється самим обчислювальним ядром.

Intel Pentium. Благодаря использованию суперскалярной архитектуры процессор может выполнять 2 команды за 1такт. Такая возможность существует благодаря наличию двухконвейеров— U и V. U-конвейер— основной, выполняет все операции над целыми и вещественными числами; V-конвейер— вспомогательный, выполняет только простые операции над целыми и частично над вещественными. Чтобы старые программы (для 486) в полной мере использовали возможности такой архитектуры, необходимо было их перекомпилировать. Pentium— первый ISC-процессор, использующий многоконвейерную архитектуру.

Сучасна суперскалярність

•Модуль передбачення умовних переходів , Кеш

•Конвеєр команд— використовується у всіх сучасних суперскалярах. КПЛ

2.1 Типи ядер.

Монолітне: Всі модулі ядра працюють в адресному просторі ядра і можуть користуватися всіма функціями, що надаються ядром. Тому модульні ядра продовжують залишатися монолітними. Модульні ядра здійснюється на рівні бінарного образу. При помилці в драйвері вилітають. Довго перекомпільовувати (бо треба все). **UNIX, MS-DOS.**

Модульное ядро— современная, усовершенствованная модификация архитектуры монолитных ядер операционных систем компьютеров.

В отличие от «классических» монолитных ядер, считающихся ныне устаревшими, модульные ядра, как правило, не требуют полной перекомпиляции ядра при изменении состава аппаратного обеспечения компьютера. Вместо этого модульные ядра предоставляют тот или иной механизмподгрузкимодулей ядра. **Linux**

Микроядро предоставляет только элементарные функции управления процессами и минимальный набор абстракций для работы с оборудованием.Большая часть работы осуществляется с помощью специальных пользовательских процессов, называемых *сервисами*. **MacOS X**

Экзоядро — ядро операционной системы компьютеров, предоставляющее лишь функции для взаимодействия между процессами и безопасного выделения и освобождения ресурсов. Экзо — приставка, обозначающая нечто внешнее, находящееся снаружи. **libOS**

Наноядро— архитектура ядра операционной системы компьютеров, в рамках которой крайне упрощённое иминималистичноеядро выполняет лишь одну задачу — обработку аппаратных прерываний, генерируемых устройствами компьютера. После обработки прерываний от аппаратурынаноядро, в свою очередь, посылает информацию о результатах обработки (например, полученные с клавиатуры символы) вышележащему программному обеспечению при помощи того же механизма прерываний.

Гибридное ядро(англ.Hybridkernel) — модифицированные микроядра (минимальная реализация основных функций ядра операционной системы компьютера), позволяющие для ускорения работы запускать «несущественные» части в пространстве ядра. **Windows NT**

2.2 Розрізняють два способи реалізації багатозадачності:

- створити один процес, що має декілька потоків виконання (threads);
- створити декілька процесів, кожен з яких має один або декілька потоків виконання.

Процес - це поняття, що відноситься до операційної системи. Кожного разу, як ви запускаєте додаток, система створює і запускає новий процес. Процес можна грубо ототожнити з exe-кодом, що виконується в окремому процесорі.

Потік (thread) - це основний елемент системи, якому ОС виділяє машинний час. Потік може виконувати якусь частину загальної коди процесу, у тому числі і ту частину, яка в цей час вже виконується іншим потоком. Наприклад, код функції, що відображує на екрані міру просування процесу передачі інформації, може одночасно виконуватися двома потоками, які обслуговують двох клієнтів одного сервера.

Всі потоки*(threads)*одного процесу користуються ресурсами породжувача їх процесу. Крім того, кожному потоку система і програміст приписує пріоритет виконання і набір структур мови З, що описують контекст потоку.

Потоки подібні до процесів, але вимагають менших витрат при своєму створенні. Вони у меншій мірі, чим процеси, захищені один від одного, але дозволяють поєднати виконання операцій і виграти в загальній продуктивності процесу.

Зазвичай ефективнішою є реалізація багатозадачності у вигляді одного процесу з декількома потоками, чим у вигляді багатьох процесів з одним потоком

У системах, подібних до традиційних версій UNIX, допускається наявність багатьох процесів, але в рамках адресного простору процесу виконується тільки один потік. Це традиційна однопотокова модель процесів. Поняття потоку в даній моделі не застосовують, а використовують терміни «перемикання між процесами», «планування виконання процесів», «послідовність команд процесу» тощо (тут під процесом розуміють його єдиний потік).

У більшості сучасних ОС (таких, як лінія Windows XP, сучасні версії UNIX) може бути багато процесів, а в адресному просторі кожного процесу — багато потоків. Ці системи підтримують багатопотоковість або реалізують модель потоків. Процес у такій системі називають багатопотоковим процесом. Надалі для позначення послідовності виконуваних команд вживатимемо термін «потік», за винятком ситуацій, коли обговорюватиметься реалізація моделі процесів.

2.3 Для потоку дозволені такі стани:

- створення (new) - потік перебуває у процесі створення;
- виконання (running) — інструкції потоку виконує процесор (у конкретний момент часу на одному процесорі тільки один потік може бути в такому стані);
- очікування (waiting) — потік очікує деякої події (наприклад, завершення операції введення-виведення); такий стан називають також заблокованим, а потік — припиненим;
- готовність (ready) — потік очікує, що планувальник перемкне процесор на нього, при цьому він має всі необхідні йому ресурси, крім процесорного часу
- завершення (terminated) — потік завершив виконання (якщо при цьому його ресурси не були вилучені з системи, він переходить у додатковий стан -стан зомбі).

Перехід потоків між станами очікування і готовності реалізовано на основі планування задач, або планування потоків. Під час планування потоків визначають, який з потоків треба відновити після завершення операції введення-виведення, як організувати очікування подій у системі.

Алгоритми планування:

- First-Come, First-Served (FCFS)
- Round Robin (RR) – типу карусель всі близько процесору по трошки виконуються
- Shortest-Job-First (SJF)
- Гарантоване планування – у всіх N користувачів ~ 1/N частини процесорного часу.

Windows

Пріоритети потоків і процесів Для визначення порядку виконання потоків диспетчер ядра використовує систему пріоритетів. Кожному потокові присвоюють пріоритет, заданий числом у діапазоні від 1 до 31 (що більше число, то вище пріоритет). Пріоритети реального часу – 16-31; їх резервує система для дій, час виконання яких є критичним чинником. Динамічні пріоритети – 1-15; вони можуть бути присвоєні потокам застосувань користувача.

Linux

Розглянемо алгоритм планування звичайних процесів [62]. В основі алгоритму лежить розподіл процесорного часу на епохи (epochs). Упродовж епохи кожен процес має квант часу, довжину якого розраховують у момент початку епохи. Здебільшого різні процеси мають кванти різної довжини. Коли процес вичерпав свій квант, його витісняють і протягом поточної епохи він більше не виконуватиметься. Керування передають іншому процесові.

2.4 Головна операція управлінняпам'яттю— розміщення програми в основній (оперативній, фізичній) пам'яті для її виконанняпроцесором. Майже у всіх сучаснихбагатозадачнихсистемахця задача передбачає використання складної схеми, відомої яквіртуальна пам'ять, яка, в свою чергу, базується на використанні однієї або двох базових технологій —сегментів чи сторінок.

Основні технології управління пам'яттю:

- фіксований розподіл
- динамічний розподіл
- звичайна сторінкова організація
- звичайна сегментація
- сторінкова організація віртуальної пам'яті
- сегментація віртуальної пам'яті

Віртуа́льна па́м'ять — схема адресації пам'яті комп'ютера, при якій пам'ять для запущеної програми реалізується однорідним масивом, в той час як насправді операційна система виділяє пам'ять блоками в різних видах пам'яті, включаючи короткочасну (оперативну) і довгочасну (тверді диски, твердотілі накопичувачі).

Також під віртуальною пам'яттю часто розуміють файл підкачки (*Windows*-системи), або окремий розділ на диску (*Unix*-системи). Ця пам'ять використовується для того, щоб дати можливість системі або користувачу одночасно виконувати більшу кількість програм, ніж це дозволяє фізична оперативна пам'ять.

Виртуальное адресное пространство каждого процесса делится на части одинакового, фиксированного для данной системы размера, называемые виртуальными страницами. В общем случае размер виртуального адресного пространства не является кратным размеру страницы, поэтому последняя страница каждого процесса дополняется фиктивной областью. Вся оперативная память машины также делится на части такого же размера, называемые физическими страницами (или блоками). Размер страницы обычно выбирается равным степени двойки: 512, 1024 и т.д., это позволяет упростить механизм преобразования адресов. При загрузке процесса часть его виртуальных страниц помещается в оперативную память, а остальные - на диск.

Учитывая, что размер страницы равен 2 в степени к, смещение s может быть получено простым отделением k младших разрядов в двоичной записи виртуального адреса.

Виртуальное адресное пространство процесса делится на сегменты, размер которых определяется программистом с учетом смыслового значения содержащейся в них информации. Отдельный сегмент может представлять собой подпрограмму, массив данных и т.п. Иногда сегментация программы выполняется по умолчанию компилятором.

При загрузке процесса часть сегментов помещается в оперативную память (при этом для каждого из этих сегментов операционная система подыскивает подходящий участок свободной памяти), а часть сегментов размещается в дисковой памяти.

Виртуальный адрес при сегментной организации памяти может быть представлен парой (g, s), где g - номер сегмента, а s - смещение в сегменте.

2.5

Однією з головних функцій ОС є керування всіма пристроями введення-виведення комп'ютера. ОС повинна передавати пристроїв команди, перехоплювати переривання і обробляти помилки; вона також повинна забезпечувати інтерфейс між пристроями та іншою частиною системи. З метою розвитку інтерфейс повинен бути однаковим для всіх типів пристроїв (незалежність від пристроїв).

Зовнішнє пристрій зазвичай складається з механічного та електронного компонента. Електронний компонент називається **контролером** пристрою або адаптером.

Драйвери — це особливий тип ПЗ, розробленого для коректної взаємодії з пристроями. Вони представляють інтерфейс для взаємодії з пристроєм через певну шину комп'ютера, до котрої даний пристрій під'єднано, за допомогою ряду команд що відправляють та отримують дані з пристрою.

Переривання — сигнал, що повідомляє процесор про настання якої-небудь події. Залежно від можливості заборони зовнішні переривання поділяються на:

- ті, що можна маскувати — переривання, які можна забороняти установкою відповідних бітів у регістрі маскування переривань (в x86-процесорах — скиданням IF у регістрі прапорців);
- ті, що не можна маскувати (англ. Non maskable interrupt, NMI) — обробляються завжди, незалежно від заборон на інші переривання. Наприклад, таке переривання може бути викликане збоєм в мікросхемі пам'яті. Обробники переривань зазвичай пишуться таким чином, щоб час їх обробки був якомога меншим, оскільки під час їх роботи можуть не оброблятися інші переривання, а якщо їх буде багато, то вони можуть губитися.

Управление вводом-выводом (англ. *IO Control, IOCTL*). Зачастую драйвер поддерживает интерфейс ввода-вывода, специфичный для данного устройства. С помощью этого интерфейса программа может послать специальную команду, которую поддерживает данное устройство. Например, для SCSI-устройств можно послать команду GET_INQUIRY, чтобы получить описание устройства. В Win32-системах управление осуществляется через API-функцию DeviceIoControl(). В UNIX-подобных — ioctl().

2.6 **Фа́йлова систе́ма**— спосіб організаціїданих, який використовуєтьсяопераційною системоюдля збереження інформації у виглядіфайлівнаносіях інформації. Також цим поняттям позначають сукупність файлів та директорій, які розміщуються на логічному або фізичному пристрої. Створення файлової системи відбувається в процесі форматування.
Сучасні файлові системи (ФС) являють собою ієрархічні структури каталогів. Хоча загальна концепція всіх ФС, в принципі, однакова, в реалізації є деякі відмінності. Два вартих уваги приклади — це символи-розділювачі каталогів та чутливість до регістру. Юнікс-подібні операційні системи (ОС) (BSD, Linux, Mac OS X) та AmigaOSвикористовують у якості розділювача каталогів символ похилої риски (/), в той час як DOS використовує цей символ для завдавання додаткових опцій у командному рядку, а в якості розділювача прийнято вживати символ зворотної похилої риски (\). У Microsoft Windows прийнята та ж конвенція за винятком китайської та корейської версій, де розділювачем є знак запитання (?). Версії MacOS до X використовували у якості розділювача двокрапку; RISC OS — дефіс.

FAT32 (від англ. *File Allocation Table* — «таблиця розташування файлів») — ця файлова система підтримує томи (логічні диски) обсягом до 8 ТБ і використовує для зберігання файлів менші фрагменти диска, ніж файлова система FAT16. Це збільшує вільний простір на диску. Файлова система FAT32 не підтримує диски, менші за 512 МБ.[1]

NTFS поддерживает систему метаданных и использует специализированные структуры данных для хранения информации о файлах для улучшения производительности, надёжности и эффективности использования дискового пространства. NTFS хранит информацию о файлах в главной файловой таблице — Master File Table (MFT). Max 2^64b

EXT3 - журналируемая файловая система, используемая в операционных системах на ядре Linux, является файловой системой по умолчанию во многих дистрибутивах. Основана на ФС ext2, начало разработки которой положил Стивен Твиди.
Максимальное число блоков для ext3 равняется 232. Размер блока может быть различным, что влияет на максимальное число файлов и максимальный размер файла в файловой системе.[2]

2.7 **Мью́текс** (англ. *mutex*, от *mutual exclusion* — «взаимное исключение») — аналог одноместного семафора, служащий в программировании для синхронизации одновременно выполняющихся потоков.

Критическая секция (англ. *critical section*) — объект синхронизации потоков, позволяющий предотвратить одновременное выполнение некоторого набора операций (обычно связанных с доступом к данным) несколькими потоками.

Семафо́р — объект, ограничивающий количество потоков, которые могут войти в заданный участок кода. Определение введено Эдсгером Дейкстрой. Семафоры используются при передаче данных через разделяемую память.

Отображение файла в память (на память) — это такой способ работы с файлами в некоторых операционных системах, при котором всему файлу или некоторой непрерывной части этого файла ставится в соответствие определённый участок памяти (диапазон адресов оперативной памяти).

3.1 **Реляційна модель даних** — логічна модель даних.

До складу реляційної моделі даних зазвичай включають теорію нормалізації. Крістофер Дейт визначив три складові частини реляційної моделі даних:

- структурна
- маніпуляційна
- цілісна

Структурна частина моделі визначає, що єдиною структурою даних є нормалізоване n-арне відношення.

Маніпуляційна частина моделі визначає два фундаментальних механізми маніпулювання даними — реляційну алгебру і реляційне числення. Основною функцією маніпуляційної частини реляційної моделі є забезпечення заходів реляційності будь-якої конкретної мови реляційних БД: мова називається реляційною, якщо вона має не меншу виразність і потужність, ніж реляційна алгебра або реляційне числення.

Цілісна частина моделі визначає вимоги цілісності сутностей і цілісності посилань. Перша вимога полягає в тому, що будь-який кортеж будь-якого відношення відмінний від будь-якого іншого кортежу цього відношення, тобто іншими словами, будь-яке відношення має володіти первинним ключем. Вимога цілісності щодо посилань, або вимога зовнішнього ключа полягає в тому, що для кожного значення зовнішнього ключа, що з'являється у відношенні, на яке веде посилання, повинен знайтися кортеж з таким же значенням первинного ключа, або значення зовнішнього ключа повинно бути невизначеним (тобто ні на що не вказувати).

Функціональна залежність (далі часто ФЗ) — концепція, що лежить в основі багатьох питань, пов'язаних з реляційними базами даних, включаючи, зокрема, їхнє проектування. Математично являє собою бінарне відношення між множинами атрибутів даного відношення і є, по суті, зв'язком типу «один-до-багатьох». ФЗ забезпечує основу для наукового підходу до розв'язання деяких проблем, оскільки володіє багатим набором цікавих формальних властивостей.

Теорема Хіта [ред. • ред. код]

Нехай дане відношення $r\,(A, B, C)$.

Якщо r задовільняє функціональній залежності $A \to B$, тоді воно дорівнює поєднанню його проекцій $r\,[A, B]$ і $r\,[A, C]$.

$$(A \to B) \Rightarrow (r\,(A, B, C) = r\,[A, B] \ \text{ JOIN } r\,[A, C])$$

3.2 Реляційна алгебра - замкнена система операцій над відношеннями в реляційній моделі даних. Операції реляційної алгебри також називаються реляційними операціями.

Група **основних** операцій:

- *виборка* (selection або restrict);
- *проекція* (projection або rpogect);
- *декартовий добуток* (cartesian product);
- *об'єднання* (union);
- *різниця* (set difference або minus),

та **три додаткові**, виведених з основних операцій

- *з'єднання* (join);
- *перетинання* (intersection);
- *ділення* (division або divide).

Приоритети операцій реляційної алгебри.

1.Перейменування (**RENAME**)

2.Виборка (**RESTRICT**) = Проекція (**PROJECT**)

3.Декартовий добуток (**TIMES**) = з'єднання (**JOIN**) = Перетинання (**INTERSECT**) = Ділення (**DIVIDE**)

4.Об'єднання (**UNION**) = Різниця (**MINUS**)

3.3 **Аномалія оновлення** - поява в базі даних неузгодженості даних при виконанні операцій вставки, видалення, модифікації записів.
Аномалії модифікації - виникнення неузгодженості записів у

таблиці при зміні даних у одного запису.

Для відносини "Студент" (ПІБ, Група, Староста), де у стовпці "Група" зберігається повна назва групи, а стовпчик "Староста" містить ПІБ старости групи, зміна значення "Староста" (наприклад, для усунення помилки) може призвести до існування більше одного старости однієї і тієї ж групи.

Аномалії видалення - видалення зайвої інформації при видаленні запису.

Нормалізація схеми бази даних — покроковий процес розбиття одного відношення (на практиці: таблиці) відповідно до алгоритму нормалізації на декілька відношень на базі функціональних залежностей.

Нормальна форма — властивість відношення в реляційній моделі даних, що характеризує його з точки зору надмірності, яка потенційно може призвести до логічно помилкових результатів вибірки або зміни даних. Нормальна форма визначається як сукупність вимог, яким має задовольняти відношення.

Нормальні форми:

- 1NF
- 2NF
- 3NF
- 4NF
- 5NF

3.4 Для проектування реляційної бази даних потрібно:

1. Визначити об’єкти, які містяться в базі даних.
2. Визначити зв’язки між об’єктами.
3. Визначити основні властивості об’єктів.
4. Визначити зв’язки між властивостями об’єктів.
5. Створити робочий словник даних для визначення таблиць, що входять до бази даних.
6. Визначити відношення між таблицями баз даних, засновуючись на зв’язках між об’єктами даних, що містяться в таблиці, і включити цю інформацію до словника даних.
7. Продумати операції, що виконуються при створенні та зміні інформації таблиць, включаючи забезпечення цілісності даних.
8. Визначити, як використовувати індекси для прискорення виконання запитів, щоб уникнути сильного уповільнення роботи при додаванні даних до таблиці і надмірного збільшення об’єму дискового простору, що займається базою.
9. Визначити користувачів, яким дозволений доступ до даних, їх редагування, а також зміна при необхідності структури таблиць.
10. Описати структуру бази даних в цілому, завершити створення словників даних для своєї бази та для кожної таблиці, що міститься в ній, розробити процедури для операцій з базою даних, включаючи створення резервних копій і відновлення вихідних файлів.

ER-модель зручна при проектуванні інформаційних систем, баз даних, архітектур комп'ютерних застосунків та інших систем (моделей). За допомогою такої моделі виділяють найсуттєвіші елементи (вузли, блоки) моделі і встановлюють зв'язки між ними.

Існує ряд моделей для представлення знань. Одним з найзручніших інструментів уніфікованого представлення даних, незалежного від реалізовуючого його програмного забезпечення, є модель "сутність-зв'язок" (*entity - relationship model, ER - model*).

Сутність – таблиця, звязок – зовнішні ключі.

3.5 **SQL** (*Structured query language* — мова структурованих запитів), складається з:

- **DDL** (Data Definition Language) — робота із структурою бази,
- **DML** (Data Manipulation Language) — робота із стрічками,
- **DCL** (Data Control Language) — робота з правами,
- **TCL** (Transaction Control Language) — робота з транзакціями.

Data Definition Language[ред. • ред. код]

Основна стаття: DDL

- CREATE — створення об'єкта (наприклад, таблиці);
- ALTER — зміна об'єкта (наприклад, додавання/зміна полів таблиці);
- DROP — видалення об'єкта.

Data Manipulation Language[ред. • ред. код]

Основна стаття: DML

- INSERT — вставка стрічки;
- SELECT — вибірка;
- UPDATE — зміна;
- DELETE — видалення.

З’єднання таблиць – JOIN (Inner, Outer, Left, Right)
Агрегатні функції – тільки при GROUP BY (AVG,SUM,COUNT)
Вкладені запити SELECT * FROM books WHERE id=(SELECT id FROM new_books WHERE name='blablabla');

3.6 Індекс (англ. *index*) — об'єкт бази даних, що створений з метою підвищення ефективності виконання запитів. Таблиці в базі даних можуть мати велику кількість рядків, які зберігаються у довільному порядку, і їх пошук за заданим значенням шляхом послідовного перегляду таблиці рядок за рядком може займати багато часу. Індекс формується зі значень одного чи кількох стовпчиків таблиці і вказівників на відповідні рядки таблиці і, таким чином, дозволяє знаходити потрібний рядок за заданим значенням.

Все индексы MySQL (PRIMARY, UNIQUE, и INDEX) хранятся в виде B-деревьев. Строки автоматически сжимаются с удалением пробелов в префиксах и оконечных пробелов (see section 6.5.7 Синтаксис оператора CREATE INDEX).

B-дерево (по-русски произносится как **Б-дерево**) — структура данных, дерево поиска. С точки зрения внешнего логического представления, сбалансированное, сильно ветвистое дерево во внешней памяти.

Использование B-деревьев впервые было предложено Р. Бэйером (англ. *R. Bayer*) и Е. МакКрейтом (англ. *E. McCreight*) в 1970 году.

Сбалансированность означает, что длина любых двух путей от корня до листов различается не более, чем на единицу.

Ветвистость дерева — это свойство каждого узла дерева ссылаться на большое число узлов-потомков.

С точки зрения физической организации B-дерево представляется как мультисписочная структура страниц внешней памяти, то есть каждому узлу дерева соответствует блок внешней памяти (страница). Внутренние и листовые страницы обычно имеют разную структуру.

3.7 **Транза́кція** (англ. *transaction*) — група послідовних операцій з базою даних, яка є логічною одиницею роботи з даними. Транзакція може бути виконана або цілком і успішно, дотримуючись цілістності даних і незалежно від інших транзакцій, що йдуть паралельно, або не виконана зовсім, і тоді вона не може справити ніякого ефекту. Транзакції оброблюються транзакційними системами, в процесі роботи яких створюється історія транзакцій.

Розрізняють послідовні (звичайні), паралельні і розподілені транзакції. Розподілені транзакції вбачають використання більш ніж однієї транзакційної системи і потребують набагато більш складної логіки (наприклад, two-phase commit — двофазний протокол фіксації транзакції). Також, в деяких системах реалізовані автономні транзакції, або під-транзакції, які є автономною частиною батьківської транзакції.

Властивості транзакцій (**ACID**)

Atomicity — Атомарність гарантує, що ніяка транзакція не буде зафіксована в системі частково.

Consistency — Узгодженість. відповідності до цієї вимоги, система знаходиться в узгодженому стані до початку транзакції і повинна залишитись в узгодженому стані після завершення транзакції.

Isolation — Ізольованість Під час виконания транзакції паралельні транзакції не повинні впливати на її результат. Ця властивість не дотримується на рівні ізольованостіRepeatable Read та нижче.

Durability — надійність. Іншими словами, якщо користувач отримав підтвердження від системи, що транзакція виконана, він може бути впевнений, що зміни, які він зробив, не будуть відмінені через будь-який збій.

3.8 К основным средствам защиты информации можно отнести следующие средства:
- парольная защита;
- шифрование данных и программ;
- установление прав доступа к объектам БД;
- защита полей и записей таблиц БД.

Парольная защита представляет простой и эффективный способ защиты БД от несанкционированного доступа. Пароли устанавливаются конечными пользователями или администраторами БД. Учет и хранение паролей производится самой СУБД. Обычно пароли хранятся в определенных системных файлах СУБД в зашифрованном виде. Поэтому просто найти и определить пароль невозможно. После ввода пароля пользователю СУБД предоставляются все возможности по работе с защищенной БД.

Шифрование данных (всей базы или отдельных таблиц) применяется для того, чтобы другие программы не могли прочитать данные. Шифрование исходных текстов программ позволяет скрыть от несанкционированного пользователя описание соответствующих алгоритмов.

В целях контроля использования основных ресурсов СУБД во многих системах имеются **средства установления прав доступа к объектам БД**. Права доступа определяют возможные действия над объектами. Владелец объекта (пользователь, создавший объект), а также администратор БД имеют все права. Остальные пользователи к разным объектам могут иметь различные уровни доступа.

По отношению к таблицам в общем случае могут предусматриваться следующие права доступа.
- просмотр (чтение) данных;
- изменение (редактирование) данных;
- добавление новых записей;
- добавление и удаление данных;
- все операции, в том числе изменение структуры таблицы.

4.1 Модель OSI (EMBBC) (*базова еталонна модель взаємодії відкритих систем*, англ. *Open Systems Interconnection Basic Reference Model*, 1978 р.) — абстрактна мережева модель для комунікацій і розробки мережевих протоколів. Представляєрівневий підхід до мережі. Кожен рівень обслуговує свою частину процесу взаємодії. Завдяки такій структурі спільна робота мережевого обладнання й програмного забезпечення стає набагато простішою, прозорішою й зрозумілішою.

Модель складається з 7-ми рівнів, розташованих вертикально один над іншим. Кожен рівень може взаємодіяти тільки зі своїми сусідами й виконувати відведені тільки йому функції.

Рівень OSI	Протоколи
прикладний	HTTP, gopher, Telnet, DNS, DHCP, SMTP
відображення	ASN.1, XML, TDI, XDR, NCP, AFP, ASCII, Unicode
сеансовий	ASP, ADSP, DLC, Named Pipes, NBT, NetBIOS, NWLink, Printer Access Protocol, Zone Information Protocol, SSL, TLS, SOCKS, PPTP
транспортний	TCP, UDP, NetBEUI, AEP, ATP, IL, NBP, RTMP, SMB, SPX, SCTP, DCCP, RTP, STP, TFTP
мережевий	IPv4, IPv6, ICMP, IGMP, IPX, NWLink, NetBEUI, DDP, IPSec, ARP, SKIP
канальний (Ланки даних)	Ring, PPP, PPPoE, StarLan, WiFi, PPTP , L2F, L2TP, PROFIBUS
фізичний	RS-232, RS-422, RS-423, RS-449, RS-485, ITU-T, RJ-11, T-carrier (T1, E1), модифікації стандарту Ethernet:

- Прикладний - Верхній (7-й) рівень моделі, забезпечує взаємодію мережі й користувача.
- Відображення - Цей рівень відповідає за перетворення протоколів і кодування/декодування даних.
- Сеансовий - ідповідає за підтримку сеансу зв'язку, дозволяючи додаткам взаємодіяти між собою тривалий час.
- Транспортний - призначений для доставлення даних без помилок, втрат і дублювання в тій послідовності, у якій вони були передані.
- Мережевий- Відповідає за трансляцію логічних адрес й імен у фізичні, визначення найкоротших маршрутів, комутацію й маршрутизацію пакетів, відстеження неполадок і заторів у мережі.
- Канальний - Цей рівень призначений для забезпечення взаємодії мереж на фізичному рівні й контролю за помилками, які можуть виникнути.
- Фізичний - Найнижчий рівень моделі, призначений безпосередньо для передачі потоку даних.

4.2 **Кана́льний рівень**(англ.*Data Link layer*)— рівень мережноїмоделі OSI, призначений для передачі даних вузлам, що знаходяться в тому ж сегменті локальної мережі. Також може використовуватися для виявлення і, можливо, виправлення помилок, що виникли на фізичному рівні. Прикладами протоколів, що працюють на канальному рівні, є: Ethernet для локальних мереж (багатовузловий), Point-to-Point Protocol (PPP),HDLCі ADCCP для підключень точка-точка.

Канальний рівень відповідає за доставку кадрів між пристроями, підключеними до одного мережевого сегмента. Кадри канального рівня не перетинають кордонів мережевого сегмента. Функції міжмережевої маршрутизації і глобальної адресації здійснюються на більш високих рівнях моделі OSI, що дозволяє протоколам канального рівня зосередитися на локальній доставці і адресації.

На цьому рівні працюють комутатори, мости.

Функції канального рівня[ред.•ред. код]

- Отримання доступу до середовища передачі. Забезпечення доступу— найважливіша функція канального рівня. Вона потрібна завжди, за винятком випадків, коли реалізована повнозв'язна топологія (наприклад, два комп'ютери, з'єднаних черезкросовер, або комп'ютер зісвітчемв повнодуплексному режимі).
- Виділення меж кадру. Ця задача так само вирішується завжди. Серед можливих рішень цієї задачі— резервування певної послідовності, яка позначає початок або кінець кадру.
- Апаратна адресація (або адресація канального рівня). Потрібна в тому випадку, коли кадр можуть отримати відразу декілька адресатів. У локальних мережах апаратні адреси (MAC-адреси) застосовуються завжди.
- Забезпечення достовірності прийнятих даних. Під час передачі кадру є ймовірність, що дані спотворені. Важливо це виявити і не намагатися обробити кадр, який містить помилку. Зазвичай на канальному рівні використовуються алгоритми контрольних сум, що дають високу гарантію виявлення помилок.
- Адресація протоколу верхнього рівня. У процесі декапсуляції вказівка формату вкладеногоPDUістотно спрощує обробку інформації, тому найчастіше вказується протокол, що знаходиться в полі даних, за винятком тих випадків, коли в полі даних знаходиться один-єдиний протокол.

4.3 Логічна структуризація мережі - це процес поділу мережі на сегменти з локалізованим трафіком. Для логічної структуризації мережі використаються мости, комутатори, маршрутизатори та шлюзи [1 - 4]. Міст (bridge) ділить розділюванесередовище передачі мережі на частини (логічні сегменти), передаючи інформацію з одного сегмента в іншій лише коли така передача дійсно необхідна (тобто коли адреса комп'ютера призначення належить інший під мережі).

VLAN (англ. Virtual Local Area Network — віртуальна локальна комп'ютерна мережа) — є групою хостів з загальним набором вимог, що взаємодіють так, ніби вони прикріплені до одного домену, незалежно від їх фізичного розташування. VLAN має ті самі атрибути, як і фізична локальна мережа, але дозволяє кінцевим станціям бути згрупованими разом, навіть якщо вони не перебувають на одному мережевому комутаторі.

Алгоритм "прозрачный мост" назван так потому, что присутствие и работа моста являются прозрачными для хостов сети.
Мост строит свою адресную таблицу на основе пассивного наблюдения за трафиком, проходящим через его порты. При этом извлекается информация об адресах источников кадров данных. По адресу источника делается вывод о принадлежности конкретного узла тому или иному сегменту сети.

Spanning Tree Protocol (STP) (протокол кістякового дерева) - мережевий протокол, що працює на другому рівні моделі OSI. Заснований на однойменному алгоритмі, розробником якого є «Мама Інтернету» - Радья Перлман[en].
Основним завданням STP є приведення мережі Ethernet з множинними зв'язками до деревоподібної топології (кістякове дерево), що виключає цикли пакетів. Відбувається це шляхом автоматичного блокування надлишкових в цей час зв'язків для повної зв'язності портів. Протокол описаний в стандарті IEEE 802.1D.

4.4 **Маршрутиза́ція** (англ. *Routing*) — процес визначення маршруту прямування

інформації між мережами. Маршрутизатор (або роутер від англ. *router*) приймає рішення, що базується на IP-адресі отримувача пакету. Для того, щоб переслати пакет далі, всі пристрої на шляху слідування використовують IP-адресу отримувача. Для прийняття правильного рішення маршрутизатор має знати напрямки і маршрути до віддалених мереж. Є два типи маршрутизації:

- Статична маршрутизація — маршрути задаються вручну адміністратором.
- Динамічна маршрутизація — маршрути обчислюються автоматично за допомогою протоколів динамічної маршрутизації — RIP, OSPF, EIGRP, IS-IS, BGP, HSRP та ін, які отримують інформацію про топологію і стан каналів зв'язку від інших маршрутизаторів у мережі.

Протокол маршрутизації (RIP, OSPF, IGRP, EIGRP, IS-IS, BGP, HSRP тощо) може працювати тільки з пакетами, які належать до одного з маршрутизованих протоколів, наприклад, IP, IPX чи AppleTalk.

4.5 Стек протоколів TCP/IP (модель взаємодії відкритих систем DoD (Department of Defence)[1]міністерства оборони США) ділиться на 4 рівні:**прикладний**(application),**транспортний**(transport),**міжмережевий**(internet) та**рівень доступу до середовища передачі**(англ.*network access layer, link layer*,рос.*Канальный уровень*). Терміни, що використовуються для позначення блоку переданих даних, різні при використанні різних протоколів транспортного рівня: TCP і UDP.
На прикладному рівні цепотік(TCP) і повідомлення (UDP); на транспортному— сегмент іпакет.
Як і в моделіOSI, дані більш верхніх рівнів інкапсулюються в блоки даних більше нижніх рівнів, наприклад, сегмент (TCP) або пакет (UDP) зі своїми даними і службовими заголовками інкапсулюється всередині поля «Дані»дейтаграми.[2]

TCP/IP— это название набора сетевых протоколов. На самом деле передаваемый пакет проходит несколько уровней. (Как на почте: сначала вы пишете писмо, потом помещаете в конверт с адресом, затем на почте на нем ставится штамп и т.д.).

IPпротокол — это протокол так называемого сетевого уровня. Задача этого уровня — доставка ip-пакетов от компьютера отправителя к компьютеру получателю. По-мимо собственно данных, пакеты этого уровня имеют ip-адрес отправителя и ip-адрес получателя. Номера портов на сетевом уровне не используются. Какому порту, т.е. приложению адресован этот пакет, был ли этот пакет доставлен или был потерян, на этом уровне неизвестно — это не его задача, это задача транспортного уровня.
TCP и UDP— это протоколы так называемого транспортного уровня. Транспортный уровень находится над сетевым. На этом уровне к пакету добавляется порт отправителя и порт получателя.
TCP— это протокол с установлением соединения и с гарантированной доставкой пакетов. Сначала производится обмен специальными пакетами для установления соединения, происходит что-то вроде рукопожатия (-Привет. -Привет. -Поболтаем? -Давай.). Далее по этому соединению туда и обратно посылаются пакеты (идет беседа), причем с проверкой, дошел ли пакет до получателя. Если пакет не дошел, то он посылается повторно («повтори, не расслышал»).
UDP— это протокол без установления соединения и с негарантированной доставкой пакетов. (Типа: крикнул что-нибудь, а услышат тебя или нет — неважно).

IP-адреса обычно представлены в виде 4-х разрядов, разделенных точками, например 192.168.123.132. Чтобы понять использование масок подсетей для распознавания узлов, сетей и подсетей, обратите внимание на IP-адрес в двоичном обозначении.

Например, в виде разрядов, разделенных точками, IP-адрес 192.168.123.132 – это (в двоичном обозначении) 32-разрядный номер 11000000010100011110111000100. Такой номер сложно интерпретировать, поэтому разбейте его на четыре части по восемь двоичных знаков.

5.1 **Канал** — частина комунікаційної системи, яка зв'язує між собою джерело та приймач повідомлень. Канал поширення сигналу може бути штучним, природним і комбінованим. У першому і (або) третьому випадку – це сукупність технічних засобів та середовища розповсюдження, що забезпечує передавання повідомлень від відправника до одержувача.

В залежності від типу сигналів, що передаються, розрізняють два великих класи каналів зв'язку – цифрові та аналогові.

Цифровий канал є бітовим трактом із цифровим (імпульсним) сигналом на вході і виході каналу. На вхід аналогового каналу надходить неперервний сигнал, і з його виходу також знімається неперервний сигнал (рис 5.2). Як відомо, сигнали характеризуються формою свого подання.

Основними характеристиками є:

1. Швидкість передвання інформації: $R = v_x * H_x$ б/с, де v_x - шкидкість передавання канальних (кодових) символів. H_x - ентропія алфавіту канальних символів.
2. Пропускна спроможність каналу – максимальне значення передавання інформації – $C = max\, R$ б/с.
3. Коефіцієнт використання каналу: $n_{еф} = R/C$ - інформаційна ефективність

Передача даних (**обмін даними**, ***цифрова передача***, ***цифровий зв'язок***) — фізичне перенесення даних цифрового (бітового) потоку у вигляді сигналів від точки до точки або від точки до множини точок засобами електрозв'язку каналом зв'язку; як правило, для подальшої обробки засобами обчислювальної техніки. Прикладами подібних каналів можуть бути *мідні проводи*, *оптичне волокно*, *бездротові канали зв'язку* або *запам'ятовуючі пристрої*.

Передача даних може бути ***аналоговою*** чи ***цифровою*** (потік двійкових сигналів), а також ***модульованою*** за допомогою аналогової модуляції, або за допомогою цифрового кодування.

5.2 Первинна мережа –це сукупність направляючих систем, систем передавання, мережевих вузлів та мережевих станцій, що забезпечують утворення типових каналів та мережевих трактів для надання їх вторинним мережам електрозв'язку та окремим користувачам.
Основною функцією первинної мережі є передача, тобто транспортування інформації між пунктами. Це може бути інформація телефонних абонентів, користувачів Інтернет, програм телебачення, різна керуюча інформація контролю і технічного обслуговування мережі, сигнали взаємодії між комутаційними станціями, інформація обліку вартості послуг, виділені сигнали синхронізації і т.ін.

На базі Єдиної первинної мережі створюються окремі **вторинні мереж**і, що призначені для організації різних видів зв'язку. Будь-яка мережа, побудована на базі каналів первинної мережі може бути названавторинною мережею. Таких вторинних мереж може бути велика кількість. Будь-яка з них визначається сукупністю:

1) кінцевих пристроїв, що перетворюють інформацію в електричні сигнали, які передаються по індивідуальним абонентським або з'єднувальним лініям, каналам, які з'єднують кожний кінцевий пристрій з найближчим вузловим пунктом мережі;

2) комутаційні пристрої даної вторинної мережі;

3) каналів, що виділені із загальної первинної мережі в дану вторинну.

Канал передачи данных определяется наличием минимум двух каналов связи, обеспечивающих передачу сигнала во взаимопротивоположных направлениях.

Один из каналов связи в таком случае объединяет порты Tx источника и Rx получателя, а другой канал объединяет порты Rx источника и Tx получателя.

В зависимости от среды распространения сигнала, для организации каждого из каналов могут быть использованы как одна, так и несколько физических линий связи.

Тракт передачи (рис. 1.1) —это электрический путь сигнала от места его возникновения до излучения в пространство.

5.3 В основе построения **аналоговых** систем передачи лежит принцип *частотного уплотнения каналов,* который называют также мультиплексированием с частотным разделением каналов. Этот принцип, в свою очередь, базируется на том, что ширина спектра передаваемых сигналов обычно существенно ниже, чем полоса пропускания физической среды распространения. По этой причине передавать только один сигнал по линии связи невыгодно, поскольку общая полоса пропускания канала будет использована незначительно. Например, полоса частот (спектр) речевого сигнала, обеспечивающая уровень разборчивости слов 90 %, составляет 3100 Гц и размещается в полосе стандартного телефонного канала связи в диапазоне 300…3400 Гц.
Для исключения влияния соседних каналов друг на друга из-за наложения спектров, вызванных неидеальностью полосовых фильтров, в качестве расчетной ширины полосы телефонного канала принимается величина 4 кГц. При этом защитная полоса частот между двумя соседними каналами составляет 900 Гц.

Вместе с тем полоса пропускания кабельной линии связи (спектр эффективно передаваемых частот) может составлять несколько мегагерц, что позволяет передавать по данной линии связи сотни и тысячи речевых сигналов. Для реализации такой многоканальной системы передачи частотные спектры различных сигналов должны быть сдвинуты относительно друг друга так, чтобы они занимали неперекрывающиеся частотные полосы. Это достигается применением в аналоговых системах передачи высокочастотных несущих синусоидальных колебаний, параметры которых (амплитуда, частота и фаза) изменяются (модулируются) пропорционально величине передаваемых полезных сигналов.

Организация многоканальной аналоговой системы передачи осуществляется путем модуляции полезными сигналами амплитуды или частоты несущих синусоидальных колебаний, имеющих различные несущие частоты. При этом происходит перенос спектра полезных сигналов на величину несущих частот. Соответствующим выбором значений несущих частот можно разместить в полосе пропускания линии связи (проводной или эфирной) спектры полезных сигналов, как это показано на рис. 3.1 (спектры сигналов во всех каналах условно показаны в виде треугольников).

5.4 Структура первичной сети предопределяет объединение и разделение потоков передаваемой информации, поэтому используемые на ней системы передачи строятся по *иерархическому принципу*. Применительно к цифровым системам этот принцип заключается в том, что число каналов ЦСП, соответствующее данной ступени иерархии, больше числа каналов ЦСП предыдущей ступени в целое число раз.
Аналоговые системы передачи с ЧРК также строятся по иерархическому принципу, но в отличие от ЦСП для них ступенями иерархии являются не сами системы передачи, а типовые группы каналов.

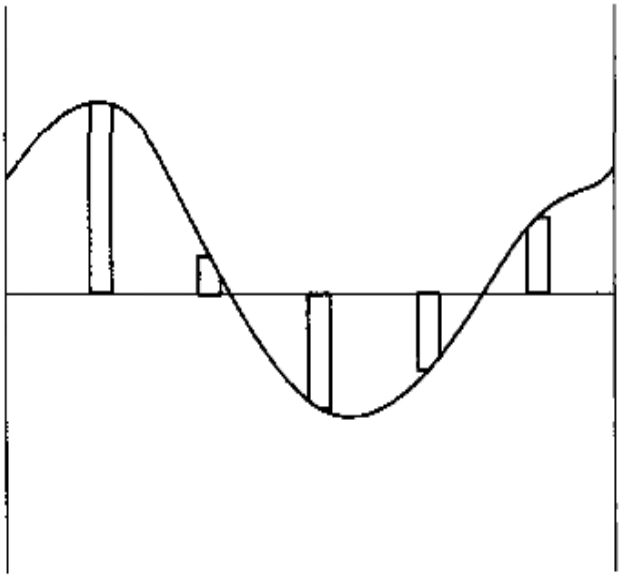
Цифровая система передачи, соответствующая первой ступени иерархии, называется первичной; в этой ЦСП осуществляется прямое преобразование относительно небольшого числа первичных сигналов в первичный цифровой поток. Системы передачи второй ступени иерархии объединяют определенное число первичных потоков во вторичный цифровой поток и т.д.

Первичным сигналом для *всех* типов ЦСП является цифровой поток со скоростью передачи 64 кбит/с, называемый основном цифровом каналом (ОЦК). Для объединения сигналов ОЦК в групповые высокоскоростные цифровые сигналы используется рассмотренный ранее принцип *временного разделения каналов*.

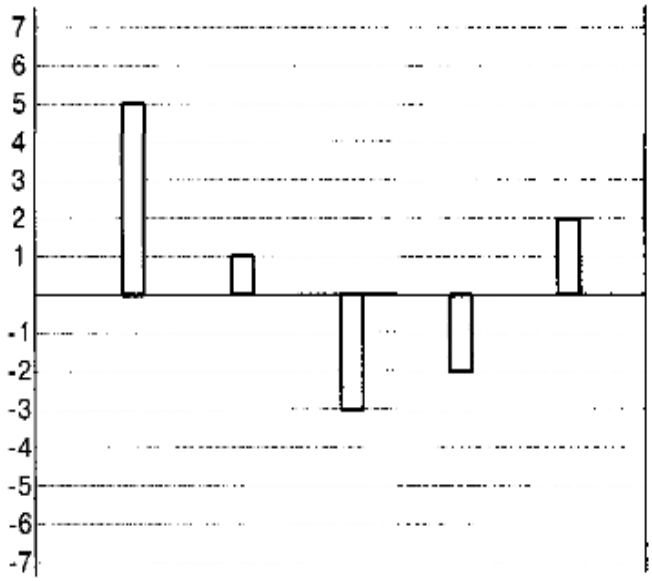
5.5 При перетворенні мовного сигналу в цифрову форму, так чи інакше, мають місце два процеси - дискретизація (sampling), тобто формування дискретних у часі відліків амплітуди сигналу, і квантування, тобто дискретизація отриманих відліків по амплітуді (кодування безперервної величини - амплітуди - числом з кінцевою точністю). Ці дві функції виконуються т.зв. аналого-цифровими перетворювачами (АЦП), які розміщуються в сучасних АТС на платі абонентських комплектів, а у разі передачі мови по IP-мереж - в терміналі користувача (комп'ютері або IP-телефоні).

Так звана теорема відліків свідчить, що аналоговий сигнал може бути успішно відновлений з послідовності вибірок з частотою, що перевищує, як мінімум, удвічі максимальну частоту, присутню в спектрі сигналу. У телефонних мережах смуга частот мовного сигналу навмисно, за допомогою спеціальних фільтрів, обмежена діапазоном 0.3 - 3.4 кГц, що не впливає на розбірливість мови і дозволяє дізнаватися співрозмовника по голосу. З цієї причини частота дискретизації при аналого-цифровому перетворенні обрано рівної 8кГц, причому така частота використовується в усіх телефонних мережах на нашій планеті.

При квантуванні безперервна величина відображається на безліч дискретних значень, що, природно, призводить до втрат інформації. Для того, щоб забезпечити в такій схемі достатній динамічний діапазон (здатність передавати без спотворень як сильні, так і слабкі сигнали), дискретна амплітуда сигналу кодується 12/13-ті розрядним двійковим числом за лінійним законом. Процес аналого-цифрового перетворення отримав, стосовно до систем зв'язку, назва імпульсно-кодової модуляції (ІКМ).



Дискретизация



Квантование

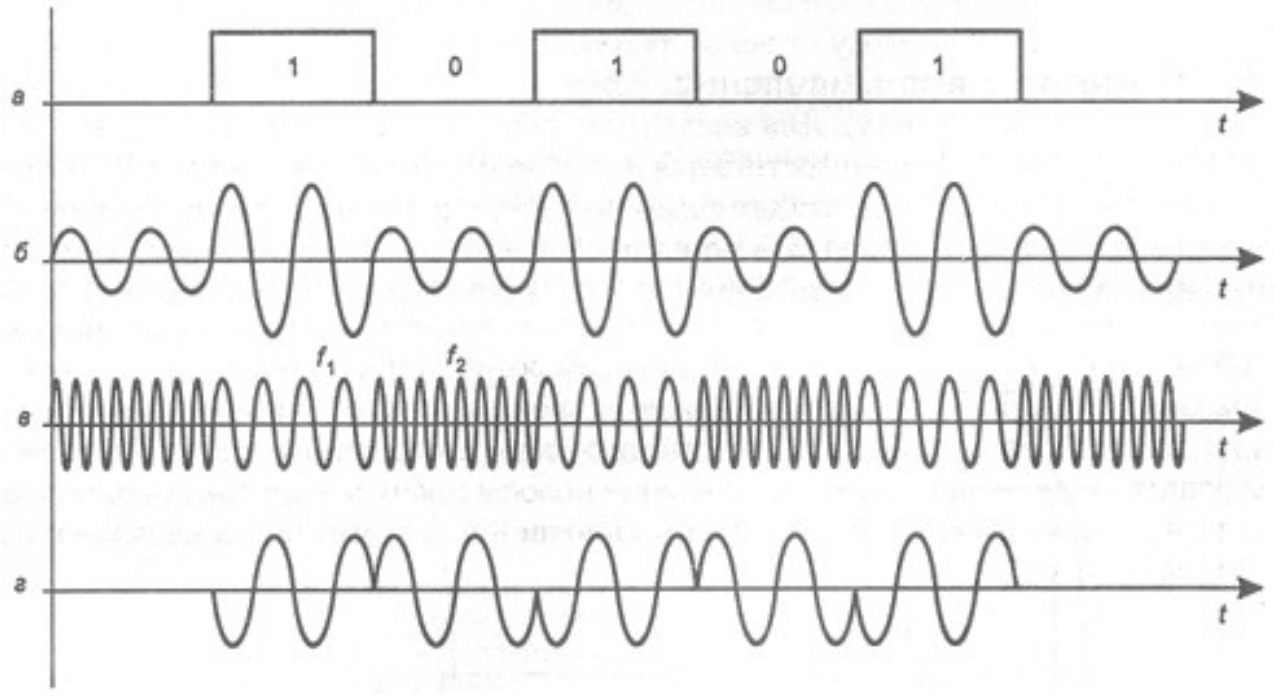
5.6 Аналогова модуляція застосовується для передачі дискретних даних по каналах з вузькою смугою частот, типовим представником яких є *канал тональної частоти*, наданий у розпорядження користувачам загальних телефонних мереж.

При *амплітудній модуляції* (мал. 2.13, б) для логічної одиниці вибирається один рівень амплітуди синусоїди несучої частоти, а для логічного нуля — іншої. Цей спосіб рідко використовується в чистому вигляді на практиці через низку перешкодостійкість, але часто застосовується в сполученні з іншим видом модуляції — фазовою модуляцією.

При *частотній модуляції* (мал. 2.13, в) значення 0 і 1 вихідних даних передаються синусоїдами з різною частотою — f_o і f_1. Цей спосіб модуляції не вимагає складних схем у модемах і звичайно застосовується в низько швидкісних модемах, що працюють на швидкостях 300 чи 1200 біт/с.

При фазовій модуляції (мал. 2.13, г) значенням даних 0 і 1 відповідають сигнали однакової частоти, але з різною фазою, наприклад 0 і 180 градусів чи 0, 90,180 і 270 градусів.

У швидкісних модемах часто використовуються комбіновані методи модуляції, як правило, амплітудна в сполученні з фазовою.



5.7

5.8

5.9 Вокодер (англ. voice coder — кодировщик голоса) — устройство синтеза речи на основе произвольного сигнала с богатым спектром.

Полосовой вокодер – анализатор

Полосовой вокодер расщепляет речевой сигнал на расширяющиеся, неперекрывающиеся частотные субполосы. Полный диапазон охватывает все частоты, которые может слышать человеческое ухо. Поступающий речевой сигнал разделяется на сегменты длительностью примерно 20 мс.

Сигнал на выходе каждой субполосы выпрямляется и фильтруется для определения его спектральной огибающей. Далее огибающая преобразуется в цифровую форму и поступает на устройство временного уплотнения (мультиплексор) для передачи по каналу связи. Обычно используются 16 субполос, охватывающих полный диапазон звуковых частот.

· Полосовой вокодер – синтезатор

Задача синтезатора состоит в изменении процесса кодирования на обратный. Полученный сигнал сначала демультиплексируется, чтобы выделить различные параметры сигнала. Часть сигнала, несущая информацию об огибающей спектра, преобразуется в аналоговую форму. Если она принадлежит сегменту огласованной речи, для возбуждения используется последовательность импульсов с частотой основного тона, в результате чего «заполняется» огибающая спектра. Если огибающая принадлежит неогласованному сегменту, для восстановления звука используется генератор шума. Наконец, сегмент сигнала фильтруется полосовым фильтром в его первоначальной частотной области.

5.10

5.11

6.1 Прототип функции описывает ее интерфейс и состоит из типа возвращаемого функцией значения, имени и списка параметров.
Кожна функція повинна мати оголошення і визначення. Оголошення функції називається її прототипом. Загальний вид прототипу виглядає в такий спосіб.

тип_значення_що_повертається ім'я_функції(тип_параметра1, ... , тип_параметра N);

У мові C++ є й інша можливість: при виклику функції можна задавати меншу кількість аргументів, ніж зазначено в прототипі. Цей механізм заснований на застосуванні параметрів за замовчуванням. Для цього в списку параметрів необхідно вказати, яке значення повинне приймати аргумент, якщо він не зазначений явно.

тип_ щоповертається_значення f(параметр_1, параметр_2=значення);

Припустимо, нам потрібно знайти максимальне з двох заданих чисел. Виникає питання, про який тип чисел мова йде: int, short, long, unsigned int, float чи double? Хоча в кожнім з цих випадків порівняння виконується зовсім однаково, для обчислення максимального значення нам довелося б написати шість різних функцій і викликати їх у залежності від типу аргументів. У мові C++ є можливість уникнути цієї незручності — механізм перевантажених функцій. У його основі лежить здатність компілятора розрізняти однойменні функції, що мають різні чи типи різна кількість аргументів. Це явище іноді називають найпростішою формою поліморфізму. Розглянемо описаний вище приклад.

6.2 В результаті дослідження Дебори Дж. Армстронг (англ. *Deborah J. Armstrong*)[4] комп'ютерної літератури, що була видана протягом останніх 40 років, вдалось відокремити фундаментальні поняття (принципи), використані у переважній більшості визначень об'єктно-орієнтованого програмування. До них належить:

Клас

Клас визначає абстрактні характеристики деякої сутності, включаючи характеристики самої сутності (її **атрибути** або **властивості**) та дії, які вона здатна виконувати (її**поведінки**, **методи** або **можливості**).Властивості та методи класу, разом називаються його **членами**.

Об'єкт

Окремий *екземпляр* класу (створюється після запуску програми і ініціалізації полів класу).

Метод

Можливості об'єкта. (функції)

Обмін повідомленнями

«Передача даних від одного процесу іншому, або надсилання викликів методів.»

Успадкування (наслідування)

Клас може мати «підкласи», спеціалізовані, розширені версії надкласу. Можуть навіть утворюватись цілі дерева успадкування.

Приховування інформації (інкапсуляція)

Приховування деталей про роботу класів від об'єктів, що їх використовують або надсилають їм повідомлення. Часто, члени класу позначаються як **публічні** (англ. *public*), **захищені** (англ. *protected*) та **приватні** (англ. *private*), визначаючи, чи доступні вони всім класам, підкласам, або лише до класу в якому їх визначено.

Абстрагування

Спрощення складної дійсності шляхом моделювання класів, що відповідають проблемі, та використання найприйнятнішого рівня деталізації окремих аспектів проблеми.

Поліморфізм

Поліморфізм означає залежність поведінки від класу, в якому ця поведінка викликається, тобто, два або більше класів можуть реагувати *по-різному* на *однакові повідомлення*.

6.3 Одиночне успадкування.

В основі механізму, що дозволяє створювати ієрархії класів, лежить принцип успадкування. Клас, що лежить в основі ієрархії, називається базовим. Класи, що успадковують властивості базового класу, називаються похідними. Похідні класи, у свою чергу, можуть бути базовими стосовно своїх спадкоємців, що в результаті приводить до ланцюжка успадкування. З одного базового класу можна вивести декілька похідних. Крім того, похідний клас може бути спадкоємцем декількох базових класів.
Успадкування буває одиночним і множинним. При одиночному успадкуванні в кожного похідного класу є лише один базовий клас, а при множинному — декілька.

Усі члени базового класу автоматично стають членами похідного. Керуючись оголошенням похідного класу, компілятор ніби збирає його з різних частин — спочатку він бере усі властивості базового класу, а потім додає до них нові функціональні можливості похідного.
Для цього використовується наступна синтаксична конструкція.

class ім'я_похідного_класу:специфікатор_доступу ім'я_базового_класу
{ // тіло класу };

Хоча всі члени базового класу автоматично стають членами похідного класу, однак доступ до цих членів визначається видом успадкування. У залежності від специфікатора доступу, зазначеного при оголошенні похідного класу, розрізняють відкрите, закрите і захищене успадкування. За замовчуванням використовується
закрите успадкування (специфікатор private).

Множинне успадкування

У похідного класу може бути декілька базових. У цьому випадку члени базових класів стають членами похідного.

class ім'я_похідного_класу:
специфікатор_доступу ім'я_базового_класу1,
...,
специфікатор_доступу ім'я_базового_класса
{ // тіло класу };

*Для вирішення проблеми доступу (class A, class B : A, class C:A, class D: B,C) використовується оператор « **::** »або віртуальне успадковування. В останньому випадку вирішується проблема з надлишковим виділенням пам'яті.*

Типи за специфікатором доступу.

*Успадкування буває публічним, захищеним і приватним. При **публічному успадкуванні**, публічні і захищені члени базового класу зберігають свій статус, а до приватних не можуть звертатися навіть функції-члени нащадка. **Захищене успадкування** відрізняється тим, що при нім публічні члени базового класу є захищеними членами нащадка. При **приватному успадкуванні**, до жодного члена базового класу навіть функції-члени нащадка права звертатися не мають. Як правило, публічне успадкування зустрічається значно частіше за інші.*

6.4 Поліморфізм в C++ досягається за допомогою абстрактних класів. Вони слугують класом-шаблоном для усіх нащадків.

```
abstract class Publication
{ //body }
```

В об'єктно-орієнтованому програмуванні **абстрактний клас –** це базовий клас, від якого не можна створити екземпляру. На практиці абстрактні класи реалізують один з принципів ООП – поліморфізм. В абстрактному класі можна описати (або не визначити) абстрактні методи та властивості. Абстрактний метод не реалізовується в класі в якому описується, але має бути реалізований в неабстрактному нащадку. Абстрактні класи вважаються найбільшими узагальненими абстракціями, тобто відношення об'єму описів до об'єму реалізації найбільше.

Віртуальний метод або **віртуальна функція** — метод об'єкта в об'єктно-орієнтованому програмуванні, різний для базового класу і класу нащадка.

Концепція віртуальної функції вирішує наступну проблему:

У ООП, якщо клас-нащадок наслідується від базового класу, об'єкт екземпляр класу-нащадку може використовуватись або як екземпляр батьківського класу (бути приведеним до батьківського класу), або як екземпляр класу-нащадка. Якщо у класі-нащадку є функції, що перекривають (мають таку ж сигнатуру) функції із батьківського класу, то поведінка при виклику таких методів (при використанні даного об'єкта як екземпляра батьківського класу) є невизначеною.

Відмінність між віртуальністю і невіртуальністю функцій вирішує цю невизначеність. Якщо функція описана як віртуальна у базовому класі, тоді буде викликана функція із класу нащадка (якщо така існує). Якщо вона не віртуальна, тоді — із батьківського класу

6.5

6.6 Существует два основных способа проектирования программных систем - структурное проектирование, основанное на алгоритмической декомпозиции, и объектно-ориентированное проектирование, основанное на объектно-ориентированной декомпозиции. Разделение по алгоритмам концентрирует внимание на порядке происходящих событий, а разделение по объектам придает особое значение агентам, которые являются либо объектами, либо субъектами действия. Однако эти способы, по сути, ортогональны, поэтому нельзя сконструировать сложную систему одновременно двумя способами. Необходимо начать разделение системы либо по алгоритмам, либо по объектам, а затем, используя полученную структуру, попытаться рассмотреть проблему с другой точки зрения.

Алгоритмическую декомпозицию можно представить как обычное разделение алгоритмов, где каждый модуль системы выполняет один из этапов общего процесса. При объектно-ориентированной декомпозиции каждый объект обладает своим собственным поведением и каждый из них моделирует некоторый объект реального мира. С этой точки зрения объект является вполне осязаемой вещью, которая демонстрирует вполне определенное поведение. Объекты что-то делают, и мы можем, послав им сообщение, попросить их выполнить некоторые операции.

Объектная декомпозиция имеет несколько преимуществ перед алгоритмической.

• Объектная декомпозиция уменьшает размер программных систем за счет повторного использования общих механизмов, что приводит к существенной экономии выразительных средств.

• Объектно-ориентированные системы более гибки и проще эволюционируют со временем, потому что их схемы базируется на устойчивых промежуточных формах. Действительно, объектная декомпозиция существенно снижает риск при создании сложной программной системы, так как она развивается из меньших систем, в которых мы уже уверены.

• Объектная декомпозиция помогает нам разобраться в сложной программной системе, предлагая нам разумные решения относительно выбора подпространства большого пространства состояний.