

Yazılım Nedir?

Yazılım = Mantık + Veri + Belge + İnsan + Program

- Bileşenlerinin, belirli bir üretim amacıyla yönelik olarak bir araya getirilmesi, yönetilebilcek ve üretilen, yönetim, arac, bilgi ve belgelerin tümünü içermesi
- Yazılım en yalın bigiminde "Bir sistemin donanım bileşenleri arasında kalan her şey" olarak tanımlanabilir.

Mantık (Algoritma) = Bilginin yapısını incelerken, kesin sonuç ulaşmak için doğru ve yanlış açısından akıl yürütme yapmakdır.

• Bilgisayarlaşmak istenen işin mevcut manığı yazılıma yansıtılacak durumundadır. Bu nedenle mantık (algoritma) bilgesini yazılımın en önemli bileşenlerindenidir.

Veri (Bilgi) = işlenmemiş bilgi veya bilginin her halidir. Bilgi ise, en basit anlamda verinin işlenmemiş teknik metin yazılır. Mülaka bir vonัวzérinde salvonak durumundadır. Veri de ortadan alınabileceği gibi yazılım içerisinde de üretilebilir.

• Yazılımın temel amacı "veriyi" "bilgi"ye dönüştürmek.

Belge (Dokümanlar) = Yazılım üretimi bir mühendislik disiplin genelinde mühendislik çalışmalarında izlenen yol ya da Kullanılan yokluklar yazılım üretimi içinde genellidir. Kurumsal kültürün gelişimi ve devrimin etkisi açısından.

• Yazılım üretimi sırasında, yazılım yaşam döngüsü adımları, belli bir düzende belgelenmelidirler.

• Planlama - Analiz - Tasarım bölgelerinde sadece döküman var

↓
↓

SRS (Yazılım Gereklilikleri) **SAD (Yazılım Minaş Dokümanı)**

İnsan (Kullanıcı, Geliştirici) = İki boyutludur, yazılımı geliştirme ve kullanır.

• Gündümüzde artık tek kişi ile yazılım geliştirilmeyip. Yazılım üretimi işin bir takım oluşturmakta ve takımın uyumlu çalışabilmesi işin çeşitli yönlerin geliştirilmesidir.

Program (Kod) = Yazılımın ana çıktıısı. Sonunda bir bilgisayar programıdır. Program işlere alındıktan sonra takım çalışmaları sürekli olarak yapılın.

Yazılımlara Sınıflaması

- Bir programı tüm ağırlıklarıyla test etmek teorik olarak mümkün olmalla beraber, uygulamada bu mümkün değildir. Yazılım aneck sınırlı sayıda veri ile sınırlabilir
 - Mantıksal Tasarım → %20
 - İsteksel Tasarım → %15
 - Kodlama → %30

- Yazılım maliyetleri giderken artarken, donanım maliyetleri giderken azalır.

Yazılım Sistemlerinin Sınıflandırılması

Kullanım Alanlarına Göre:

- ↳ Ticari, Özgür ve Açık Kaynak, Gömülü, Kişisel Yazılımlar

İşletim Sistemine Göre:

- ↳ Windows, macOS, Linux, Android, iOS

Yazılım Türüne Göre:

- ↳ Sistem, Uygulama, Programlama, Veritabanı, Ağ, Yapay Zeka Yazılımları, Gelişime Dili ve Platforma Göre Sınıflandırma;

- ↳ Java, C++, Web tabanlı, Mobil uygulamalar.

Kullanıcı Arayüzüne Göre Sınıflandırma:

- ↳ GUI, CLI

Dağıtım Yerine Göre Sınıflandırma:

- ↳ Kurulum Gerektiren, Tuşlaklı, Bulut Tabanlı Yazılımlar

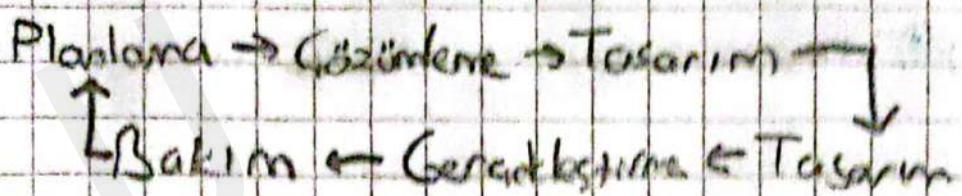
Kullanım Alanına Göre Sınıflandırma:

- ↳ Muhasebe, Sağlık, Eğitim, Oyun yazılımları,

Lisans Modeline Göre Sınıflandırma:

- ↳ Kapalı Kaynak, Açık Kaynak, Ücretsiz, Ticari Lisanslı Yazılımlar

Vapour Vision | Digital



- Herhangi bir yazılımın, Üretim aşaması ve Kullanım aşaması birlikte olmak üzere gerçekleştirtiği tüm olaylar bisiminde taranır.
 - Yazılım işlevleri ile ilgili genel bilimler sürekli olarak değiştiği ve genişlediği için, Söz Konusu olaylar bir döngü bisiminde ele alınır.
 - Döngü içerisinde herhangi bir aşamada geriye dönük ve ilerlek söz konusudur. Bu nedenle tek yönlü ve doğusul değildir.

Ne Sağlar?

- Mühendislik faaliyetlerinin (Üretim, İletme ve Bakım) yazılıma uygunlaşması
 - Üretilenin Kullanına Karar Tüm Sürecin Fazları ayrılmış ve böyledir (özetim Kolaylığı). Sağlar
 - Her osama birbirinden farklılıklar içindedir dolayısıyla her osanın da ne yapılacağını belirtir. Her osanın standartları uyumludur ve Kaliteli yazılımlar geliştirilir.

Vazilik Yecam Döngüsü Standartları IEEE 12207 standartı vardır.

Gercek Hayatta Program Geliştirme : Gelsigi gizli modelllene her furdan bittin forvara geri donus var. Planlara ve belgeleme yapilmadi, mühendislerin kullanmadigi, proje yorumunu etkili mi?

Planlama => Yazılım一生涯ının ve insan kaynaklarının planlandığı, birey,

Analiz (Gözdenleme) => Mevcut sistemin analizi ve önerilen sistemin modellenmesi

Tasarım => Mantıksal tasarruf, önerilen sistemin yapısı oluşturulur.

Oluştuğu örgütsel değişiklikler önerilir.

Fiziksel tasarruf, yazılımı içeren bileşenler ve bunların uyumlulukları

Gereklendirme => Kodlama, giàzden geçirme, test etme ve kurulum.

Bakım => Hata giderme ve yeni ekleneler yapma的过程

Yazılım Yaşam Döngüsü Temel Adımları

Yazılım yaşam döngüsünde belirtilen süreçlerin geliştirme aşamasında hangi düzen ve sıradır nasıl uygulanacağı, tanımlayan modellerdir.

Belirtim (Specification) Yöntemleri => Bir çekirdek süreçte ilişkili fonksiyonları yerine getirmek için kullanılan

Sürek Akışı => Süreçler arası ilişkilerin ve iletişimini gösterdiği yöntemler (Veri Akış Semaları, Yapısal Semalar, Nesne/Sınıf Semaları)

Sürek Tanımlama => Süreçlerin iş işlegiini göstermek için kullanılan yöntemler (Düz metin, Algoritmalar, Karar Tabloları/Algıcları, Anatomik Dil)

Veri Tanımlama => Süreçler tarafından kullanılan verilerin tanımlanması, için kullanılan yöntemler (Neone ilişkisel model, Veri tabanı tablolari, Veri sözlüğü)

Sürek (proses) Modelleri => Yazılım yaşam döngüsünde belirtilen süreçlerin geliştirme aşamasında, hangi düzen ve da sıradır, nasıl uygulanacağını tanımlayan modellerdir

Gelişgazi Modeli → Herzberg: bu model iyi fa yönetime aittidir, sadece geliştirilen kişiye bağlı, tek kişilik üretim ortamının bulunduğu, 1960'lı yıllarda Kullanıcılar konsit programlama

Başak Modeli → 70'li yıllar, yazılım yaşam döngüsünün degrusel olarak isletildiği modellerdir. Belgeleme: ayrı bir süreç olur. (Teslim öncesi) Geniş dönüştür hesaplanamamıştır.

Koçlayan (Sebil) Modeli → Yaşam döngüsü tarihi admaları baştan sona bir koza izleyenek gerçekleşir. Sadece projenin başında müsteri ile ilişkisi vardır. Uzun projelerde Kullanılmasının karsılıklı olursa kullanılabilir. Belgeleme: doğal bir parça olmak alır. Geniş dönüştür + anımlıdır.

Süres Modeli → Kullanıcı, mimar, Gerçeklestirme modelleri vardır. Her bir genel sınımlı sonrası testi öne plana çıkarır. Sol tarafta üretim, sağ tarafta ise sınamayı gösterir. **Kullanıcı Modeli**nde Kullanıcı ile olan ilişkiler tanımnanmaya ve sistemin nasıl kullanılacağına ilişkin sınıma belirtimleri ve planları ortaya çıkarır.

Mimar Modeli: tüm sistemin sınaması içinden olur.

Gerçeklestirme Modeli: yazılım modüllerinin kodlanması ve testi olur. Belirsizliklerin az, iş tanımlarlarının belirgin olduğu projelerde Kullanıcı Risk Gözümlere yok ve degrusal.

Heterozik (Spacial) Model → Risk analizi, üretim, Kullanıcı Dəyərlərdən ve Planlara hissəsi vardır. Olabildiğince erken hataları giderməyi sağlar. Prototip formadır ve dəlbildiğinde erkendir. Yarım Kullanıcı personelinin süreci erken katalınması lazımdır. Detallılaşdırılmış ve həkədil planlanması yapılır. Müşteriye / Kullanıcıya serum eksiyini giderir. Vireneli antitimsal bir yaglimı verdir.

Prototiplere → Gereksinim fərziyyətindən həzərətən hazırlanmışdır. Gerçeklestirme. Gereksinimlər netlegrikcə Kullanıcı ihtiyaclarını göstərər. Müşteri memuru olaraq Kader devenir. Belgeleme yoktur.

Evinimsel Geliştirme Modeli → Pilot uygulama şəklində Kullanıcı, test et, güncelle, dəqiqələrə təsdi. İlk evim basarılıysa model basarılidır.

- **Hesablı geliştirme (exploratory development)** → Müşteri ilə çalışıp son sistemini yaratır.
- **Atilicak Prototiplere (throw-away prototyping)** → Sistem gereksinimlerini anlamak

Araştırma Modeli => Üretilen her yazılım sürünumü birbirini kapsayacak ve giderek anton sayıda işlev içerecek şekilde geliştirilir. Öğrenci projeleri gibi, Uzun zaman alabilecek ve sistemin eksik işlevlikle çalışacağı türdeki projeler uygun olur. Bir taraftan kullanım, diğer tarafta üretim vardır. Geçerlek konur.

Araştırma Tabanlı Model => Yap-At, Özelliğe belirsizlik üzerine çalışan ortamlarda kullanılır. Sonuçlar belirgin değildir. Yazılım sınırlı sayıda kullanılır ve işlen sonrası çöp olur. Projelere uygun değildir.

Metodolojiler

! Yazılım mühendisliğinde yeni metot veya yöntemler ve metodolojiler geliştirmeniz bekleniyor

- Yazılım yaşam döngüsü aşamalarında kullanılacak birbirleriyle uyumlu yöntemler bütündür. Günümüzde metodolojiler çağlayan veya heteristik modeli tercih eder. Konfigürasyon, malzeti, kalite, risk, değişiklik yönetim modellerini, kullanıcı arayüzü ve ilişkisel modellerini, standartları içermelidir.
- Yerden Yanlış Sistem Tasarımı Metodolojisi, today uygulanabilirdir. Çağlayan modelini tercih eder. Bir çok CASE aracı tarafından desteklenir.

Bilgisayar Ölçü Birimleri

Bilgisayar ölçü birimleri, bilgisayının hafızasında (RAM) istenilen, veri kaynakları (Hard Disk, DVD vb.) kaydedilen ve ya ögde transfer edilen verinin boyunu ölçmede kullanılır birimlerdir.

Veri Birimleri

- Bit - 0 veya 1 en küçük veri birimi.
- Nibble - N, 4 bit
- Byte, 8 bit
- Word, 32 bit
- Bir gok 1SS, 1TB de cılyık veri kapasitesini kapsamaktadır.
- 1 TB ver 1 trilyon baytan biraz daha fazladır.

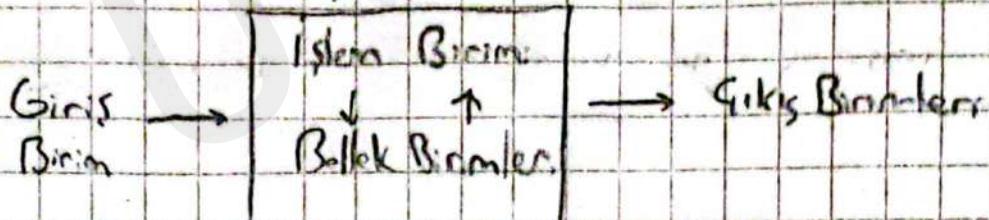
Ölçü Birimleri

- Inch - 2,54 cm
- Dots Per Inch (dpi) - Inch başına düşen nokta sayısı
- Pixel - Nokta - Ekrandaki En küçük birim (nokta)
- Resolution (Görürülük) - Yazıcılarda dpi birim, monitörde yazış ve çizim akşamda toplam noktaçık sayısı.

Hz Birimleri

- Her Hz - Hz - Bir saniyedeki devir/ yenileme hızı (frequency)
- Flops - floating point operations per sec - saniyedeki reel sayı işlem sayısı
- bps - bits per sec - Saniyede iletilen bit sayısı
- MIPS - Million Instruction Per Sec - Saniyede milyon circa'dan fazla sayıda
- RPM - Revolutions per minute - dakikada devir sayısı
- Bit rate - iletilen bit sayısı / bit oranı
- Refresh rate - saniyedeki görüntü yenileme hızı.

Bilgisayar Donanımı



CPU (Central Process Unit)

- Bilgisayar beyni
- Arithmetic logic unit - ALU - Hesaplamalar yapır. - Register yapar
- En fazla kullanılan küçük veriler Cache'de depolanır. Hızı artırmır.
- Hz birimi olarak Herz kullanılır.
- Assembly

Hafıza

- CPU'lar üzerindeki hafıza bitimleri geçicidir (Volatile).
 - Data yüksek kapasite için RAM, ROM
 - ↳ RAM (Random Access Memory), Ana bellek.
 - ↳ ROM (Read Only Memory)
 - * PROM → Programabilen
 - * EEPROM → Ultra hizli ışıkla silinip yeniden programlanabilir
 - * EEPROM → Elektriksel olarak silinip yeniden programlanabilir.
- Data kalıcı hafıza isim: HDD, SSD, USB, SD/DVD. Kalıcıdır (Non-Volatile)

CPU

GPU

TPU

Küçük Modeller

Küçük Veri Küçükler:

Tasarım Alanı: Arastırma
Central Processing Unit

Orta-Büyük Modeller

Orta-Büyük Veri Küçükler:

Görüntü, Video İşleme
Graphics Processing Unit

Matris Hesaplamaları

Yögen vektör İşleme

Özel Tensor Flow yok.
Tensor Processing Unit

- 1996 yılında proje başarısı oranı %15 iken 2004 de ise %31'dür. Bu sun nedenler arasında olan yazılım geliştirme yöntemlerinin gelişmesi olmalıdır.
- SQAP - Yazılım Kalite Garanti Planı (Software Quality Assurance Plan)
- SVVP - Yazılım Doğrulama ve Geçerlilik Planı (Software Verification and Validation Plan)

Günümüzdeki Yazılım Projelerinin Başarısız Olma Nedenleri:

- Müşterilerin isteklerini doğru analiz edememeleri.
- Proje için uygun ekibi kuramamak.
- Yanlış teknoloji ve mimari Seçimleri.
- Geleneksel yöntemlerin eksiklikleri.
- Müşteriyle iletişiminden kaçınmak.

Geçik Yazılım Vantajları:

- Tekrarlamalı ve arttırmalı bir ürün geliştirme yöntemidir.
- Bir kez ve etkileşimi, sürec ve oraca tercih eder.
- Çalışan bir yazılımı, detaylı ürün belgeleridir hale getirir.
- Müşteri ve işbirliğini, sözleşmedeki hedefin kurallara tercih eder.
- Değişikliklere uyum sağlayabilir: belirli bir plana tercih eder.
- Hızlı, docum. ve kullanışlı yazılım + müsteri memnuniyeti.
- Geliştiriciler ve iş.adamları arasında yakın işbirliği olmalı, yörük yörük.
- Yazılımın basit ve en diremlisi kullanılabilir olmasıdır.
- Kendi kendini organize eden tokam yapısı gereklidir.

Kısa vadeli planlar ve küçük parçalar halinde yazılımin gelişmesini öngörür. Kullanıcıdan alınan feedbackler almazsa olursa. Değişen ihtiyaçları karşılayıp, sık sık ürün teslim etmeli dir. Tokamlara sadece sorun söyleyen, müsteri ni istedigini hedefledir. Riskler: arızalara dair önlemlerdir.

Cevik Yazılım Teknikleri => Çapraz fonksiyonlu, Vendi kendine organize olan, testim edilebilecek ürünler ortaya koyan, hedefleri net olan 3-7 kişilik takımlardır. Özellikle birei tekim düşüncesidir.

Cevik Yazılım Sınıfları => Karbon, DSDM (^{Dynamic System} Development Method)

FDD - Feature-Driven Development

- Avustralyalı Jeff De Luca tarafından geliştirildi.
- Sistem hazırlamak için genel sistem bütünlüğe olmalıdır.
- Büyük projelerde kullanılabilir.
- Süreç adımları basit olmalı.
- İyi süreç arka plana sokular ve insanların sonucuna odaklanılmalıdır.
- Kısa, iteratif, özellik yaklaşımı yaşam döngülerini iyi sunır ve bu.

RUP - Rational Unified Process

- IBM tarafından oluşturuldu.
- Busansız bir yazılımdaki sorunların aşılması, boyutları, yazılım oluşturmak için oluşturulan süreçtir.
- Müşteriye ve yazılımcıya organize edebilmesi.
- Standart tanımlı adımları olması.
- Oluşturulan yazılımdaki sık değişiklikler öngörebilmesi.
- Proje yönetim aktivitelerinin çok fazla olmaması.
- Basit olması.

XP - Extreme Programming - Us Programlama

- Yazılım geliştirme süreci boyunca son derece kaliteli olmak koşuluyla çalıştırılabilir kod üretmektedir.
- XP metodolojisi süreci en başından itibaren çalıştırılabilir kod süreçin merkezinde tutmaktadır. İsmi bundan dolayı XP'dir.
- Planlama, sık ve hızlık评审, Basit tasarım, Önce test, Refactor etme, Haftalık 40 saat çalışma, Müşteriyle yakın iletişim, Kodlangıç standartları, özellikleri varır.
- Özellikler ürün sahibi tarafından derecelendirilir ve mühendislik pratiği içenir.
- Sprintler gerçekleştirilebilir ve t+2 hafta süren

Scrum

f (backlog sorumlusu)

(Plan)

g (Coder)

Scrum Takımı => Ürün Sahibi, geliştirme ekibi ve Scrum master'dan oluşur. Takım kendini örgütler. Varaklılık, esneklik ve verimliliği optimize etmek için tasarlanmıştır. (Geçik Yatırımlı Takımı)

Backlog => Müşteriden ve son kullanıcıdan gelen gereklilikler isein "Ne yapacağız?" sorusunun yanıtını içerir, herkese açık ve herkes tarafından müdahale edilebilir. Risk, iş değeri, zaman gibi kavramlara göre ürün sahibi tarafından sınırlanır. User Story'lerden oluşur.

User Story => Müşteri, son kullanıcı veya ürün sahibi için değerli olan ve aramız içinde eden genellikle fonksiyonel özelliklerin belirtildiği ifadelerdir.

As X I want Y so that Z

X kişisi olarak Z olsun diye Y'yi istiyorum

Sprint => Belirli bir sürede sahiptir. Sonunda ortada olan bir çıktı olmalıdır. Toplantılarla içerik bölünür. Her gün toplantı yapıltır.

- Scrum mühendislik pratiği içermey
- Sprintler en son halini aldıktan, toplantı yapıldiktan sonra değişmez. Sprintler 2 hafta - 1 ay sürer.
- Özellikler geliştiriciler tarafından derecelendirilir
- Herhangi bir mühendislik pratiği tanımaz

Mikroistansı

- Modüller ayrı ayrı CPU'ya ek olarak bulunur.
- CPU
- Maliyet yüksek.
- Genel amaçlıdır. Bir çok işi yürüre yetenir.
- Yüksek güç tüketimi.
- Hız yüksektir.
- Dönerimi karmaşıktr.
- Von Neumann mimarisi.
- Bilgisayarda kullanılır.
- Genellikle büyük.

Mikroarayızı

- Modüller miyelin kilit içersinde birer birerdir.
- CPU, RAM, ROM, Timer, I/O
- Maliyet düşük.
- Belirli bir işi gerçekleştirmek için kullanılır.
- Düşük güç tüketimi.
- Hız düşüktür.
- Dönerimi karmaşık değildir.
- Harvard mimarisi.
- Bir işlem yerine getiren cihazda çalışır.
- Genellikle küçük.

Von Neuman Mimarisi => Komutlar ve veriler ayrı belirlidir. İşlemci, ALU, hafıza, I/O birimler vardır. İşlemciin doğası gereği ya komutlarda ya da verilerde çalışabilir. Fakat ikisi ayrı anda çalışmaz. Sırayla komut yürütme, PC

Harvard Mimarisi => Komutlar ve veriler ayrı belirlidir. Veri ve komut aktarımında iletişim yolları da birbirinden bağımsızdır. İşlemci aynı esnada hem komutları değerlendirip hem de verileri işlemebilir. Önbelleğe ihtiyacı yoktur. Es zamanlı komut yürütme, mikroarayızıcı.

RISC (Reduced Instruction Set Computer) => Daha az karmaşık, daha az dönerim ihtiyacı, kod tekrarlı, hızlı ve optimize, gömülü sistemde mobil cihazlar. Daha fazla RAM harcar.

CISC (Complex Instruction Set Computer) => Çeşitli işlemler, tek bir komutta birlestiren bir komut seti kullanır, daha fazla karmaşık, daha fazla kaynak tüketimi, daha yavaş ama esnek, bilgisayar ve sunucular. Daha az RAM harcar.

Sayı Sistemleri

Decimal (Desimal) \Rightarrow Gündelik hayatımızda kullanırız. Tabanı 10'dur.
 ikili Sayı Sistemi (Binary) \Rightarrow Sistemin Tabanı 2'dir. Sadece 0 ve 1
 kullanılır. Her sayı digit olarak ifade edilir.

7 6 5 4 3 2 1 0	1 0 1 0 1 1 1 0
↓ MSB	↑ LSB
(Most Significant) Bit	(Least Significant) Bit

Binary - Decimal Çevrimi

- $(101011)_2 = 1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^3 + 1 \cdot 2^5 = 43$

5 4 3 2 1 0

- $(55)_{10} = (100001)_2$

- $(100,101)_2 = 1 \cdot 2^{-2} + 1 \cdot 2^{-1} + 1 \cdot 2^{-3}$

2 1 0 -1 -2 -3

$$\begin{array}{r} 33 \\ \hline 32 & 16 & 8 & 2 \\ - & 16 & 8 & |2 \\ \hline 1 & 0 & 2 & |2 \\ & 0 & 4 & 2 & |2 \\ & 0 & 2 & 1 & |2 \\ & & 0 & & |2 \\ \hline & & & 0 & \end{array}$$

$$\begin{array}{r} 8 \\ \hline 6 & 4 & 2 & 1 \\ - & 4 & 2 & |2 \\ \hline 2 & 2 & 1 & |2 \\ & 0 & 4 & 2 & |2 \\ & 0 & 2 & 1 & |2 \\ & & 0 & & |2 \\ \hline & & & 0 & \end{array}$$

- $(8,875)_{10} = (1000,111)_2$

$$\begin{array}{r} 0,875 \\ \times 2 \\ \hline 1,75 \\ \downarrow \\ 0,75 \\ \times 2 \\ \hline 1,5 \\ \downarrow \\ 0,5 \\ \times 2 \\ \hline 1,0 \end{array}$$

Binary Sisteminde 4 İşlem

Carry (C) \rightarrow Elde Borrow (b)

- $0+0=0, 0+1=1, 1+1=10, 1+1+1=11, 1+1+1+1=100$
- $0-0=0, 1-1=0, 1-0=1, 0-1=1$
- $0/0=0, 0/1=0, 1/0=0, 1/1=1$
- $0 \cdot 0=0, 1 \cdot 0=0, 0 \cdot 1=0, 1 \cdot 1=1$

$$\begin{array}{r}
 1001\ 1111\ 0001 \\
 + 0010\ 0101\ 1101 \\
 \hline
 11000100\ 1110
 \end{array}$$

Oktal Sayı Sistemi

- 0, 1, 2, 3, 4, 5, 6, 7 rakamlarını kullanır.

Oktal - Desimal Çevrimi

$$(564)_8 = 8^0 \cdot 4 + 8^1 \cdot 6 + 8^2 \cdot 5 = (372)_{10}$$

$$(365)_{10} = (555)_8$$

$$\begin{array}{r}
 365 \mid 8 \\
 -360 \quad | 45 \mid 8 \\
 \hline
 005 -40 \quad | 5
 \end{array}$$

↙ 5

$$\begin{array}{r}
 (340)_8 \\
 + (750)_8 \\
 \hline
 (1310)_8
 \end{array}$$

$$\begin{array}{r}
 9 \mid 8 \\
 -8 \quad | 1 \\
 \hline
 1
 \end{array}$$

$$\begin{array}{r}
 11 \mid 8 \\
 -8 \quad | 1 \\
 \hline
 3
 \end{array}$$

$$\begin{array}{r}
 (754)_8 \\
 - (456)_8 \\
 \hline
 (276)_8
 \end{array}$$

Heksadesimal Sayı Sistemi

- Tabanı 16'dır. 0..15 arası sayıları kullanılır.

$$\begin{array}{l}
 1, \dots, 9, 10, 11, 12, 13, 14, 15 \\
 1, \dots, 9, A, B, C, D, E, F
 \end{array}$$

1010	A	1101	D
1011	B	1110	E
1100	C	1111	F

Decimal - Hex Sayıları

$$\bullet (CFA54)_{10} = 4 \cdot 16^0 + 5 \cdot 16^1 + 10 \cdot 16^2 + 15 \cdot 16^3 = (64084)_{10}$$

$$\bullet (8125)_{10} = (1FB5D)_{16}$$

$$\begin{array}{r} 8125 | 16 \\ - 8112 | 16 \\ \hline 0013 - 496 | 16 \\ \hline 011 | 16 | 1 \\ \quad \quad \quad \downarrow \\ \quad \quad \quad 15 \end{array}$$

$$\begin{array}{r} (A9B30)_{16} \\ -(BFF50)_{16} \\ \hline (169A80)_{16} \end{array}$$

$$\begin{array}{r} 26 | 16 \\ - 16 | 1 \\ \hline 10 \\ \hline - 16 | 1 \\ \hline 6 \end{array} \quad \begin{array}{r} 25 | 16 \\ - 16 | 1 \\ \hline 9 \\ \hline - 16 | 1 \\ \hline 5 \end{array}$$

$$\begin{array}{r} (F9A40)_{16} \\ -(AF020)_{16} \\ \hline (4AA20)_{16} \end{array}$$

Binary - Octal Değişimi

$$(1\underset{1}{1}\underset{1}{0}\underset{1}{1}\underset{0}{0}1)_2 = (351)_8$$

Binary - Hex Değişimi

$$(FA752)_{16} = (1111\ 1010\ 0111\ 0101\ 0010)_2$$

Boole Cebri

- Önermeler veya nesneler arasında ilişkileri besinler
- Simgesel matematiksel bir mantık sistemi
- 1854, George Boole, mantık cebri
- 1938, Shannon, Aritmetik cebri
- Bilgisayarlarda kullanılan devrelerin tasarımını için gerekli teknik
- Sensörler, kart geçiş sisteminin temeli budur
- n bitlik bir ifadece en fazla 2^n ifade vardır

Degisimsiz Kuralı: $X+Y = Y+X \quad X \cdot Y = Y \cdot X$

Birlesme Kuralı: $X+Y+Z = (X+Y)+Z$

Dağılma Kuralı: $X \cdot (Y+Z) = XY + XZ$

Özdeşlik Kuralı: $X+X=X \quad X \cdot X=X$

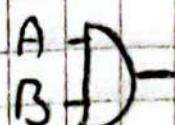
VE Kuralı: $X \cdot 1 = X \quad X \cdot 0 = 0$

VEYA Kuralı: $X+0=X \quad X+1=1$

Tanımlayıcı Kuralı: $X+\bar{X}=1 \quad X \cdot \bar{X}=0$

De Morgan Kuralı: $\overline{X \cdot Y} = \bar{X} + \bar{Y}$

Tersin Tersi Kuralı: $(X')' = X$



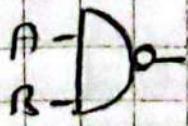
VE Kapsı
(AND)



VEYA Kapsı
(OR)



DEĞİL Kapsı
(NOT)



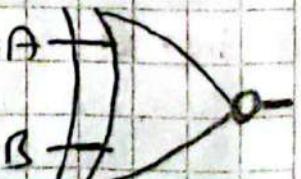
VE Değil Kapsı
(NAND)



VEYA Değil Kapsı
(NOR)



VE DA Kapsı
(XOR)



VE DA Değil Kapsı
(XNOR)

$$\bar{A} \cdot B + A \cdot \bar{B}$$

Farklı durum 1

Aynı durum 1

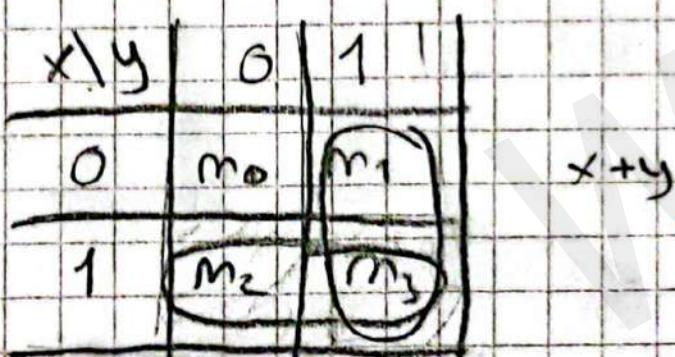
Minimum Terim (minterm) = i lgili deðiþkenin 0 ise tümleyenini 1 ile kendisini alip çarpma seklinde gösterilir. En sonda toplamı alınır.

- $F(x, y) = \underline{\bar{x}\bar{y}} + \underline{x\bar{y}} + \underline{x y}$ ifadesini mintermlerle gösteriniz.

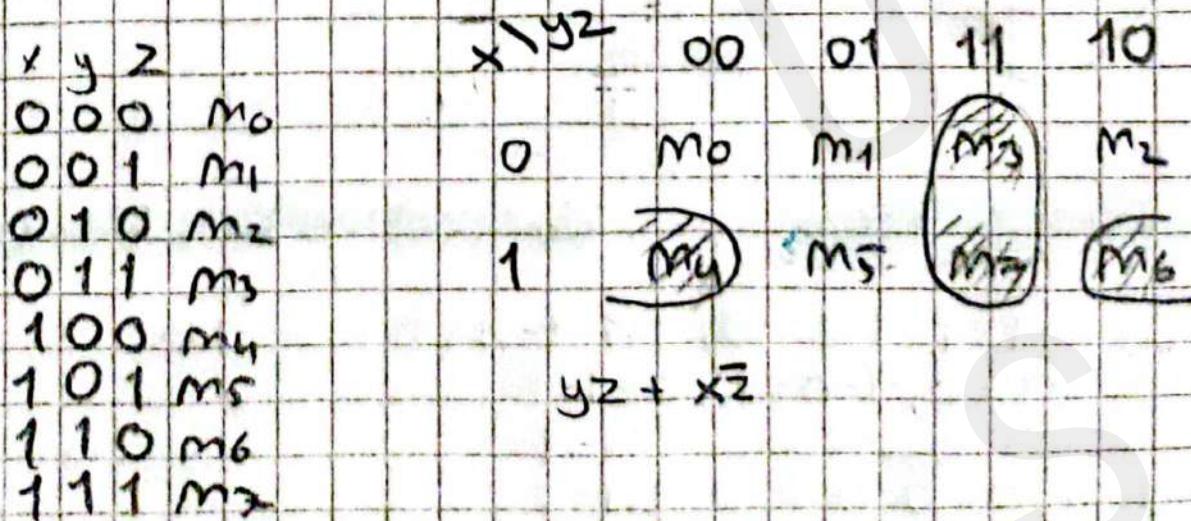
x	y	
0	0	m_0
0	1	m_1
1	0	m_2
1	1	m_3

$$\sum(0, 2, 3)$$

- $F(x, y) = x + \bar{x}y$ fonksiyonunu Karnaugh diyagrami ile sadeleştirin.



- $F(x, y, z) = \bar{x}yz + xyz + x\bar{y}\bar{z} + xy\bar{z}$ fonksiyonunu Karnaugh diyagrami ile sadeleştirin.



Algoritma \Rightarrow Bir problemin çözümünde izlenenek yolu anlatmaya gelir. Bir programlama dili ile ifade edilirse program adını alır.

• Problemlerin bir farklı ifade edis sekli vardır.

- 1- Akış diyagramı
- 2- Pseudo Kod (Kaba Kod)
- 3- Matisel Düzlemlerde

Algoritmada Olması Gereken Özellikler

Etkin ve Genel Olma \Rightarrow Algoritma etkin olma ve geneksiz tekrarlar bulunmamalıdır. Gerekiginde diğer algoritmalarında kullanılmelidir. Her koşulda ve giriş değerinde doğru sonuca ulaşmalıdır. Algoritmada çözülmeyen problem kalmamalıdır, optimum olmalı.

Sınırlı Olma \Rightarrow Algoritma sonsuz döngüye girmemelidir.

Yanalnzılık \Rightarrow Algoritma farklı yürütüklüklerde aynı giriş değerleri: iki aynı sonuc eide odurmelidir.

V/O Tonundi Olma \Rightarrow Kullanıcıdan veri almaları, bu veriyi de işlemelidir.

Başarım \Rightarrow Performans kriterlerini dikkate alarak yüksek başarımı programlar yazılmalıdır.

repeat-until

while

for

goto

yapısal
Programlama

Nass - Schneidman
Semantik

Veri => Algoritmalar tarafından istenen en temel elementlerdir.

Bir programın etkin, anlaşılır ve doğru olabilmesi için, algoritmanın işleyeceği verilerin düzenlemesi gereklidir.

Veri Yapısı (Data Structure) => Verinin veya bilginin bellekte tutulma şeklini veya düzenini gösterir.

Tanımlı Veri Yapıları => Daha çok Programlama dilleri tarafından előredefineden değişken veya sabit bildirimini yapılarak kullanılır. int, float, vb

Tanımlamalı (Biles.) Veri Yapıları => Kendisinden özetki tanımlamalı ve temel veri yapıları üzerine kurulurlar; yani Önceden geçerli olan veri yapıları kullanılarak sonradan tanımlanırlar. Struct (topluluk), Union (ortaklık), bit dizeyinde erişim

Veri Modeli (Data Model) => Verilenin birbirleriyle ilişkisel veya sırasal durumunu gösterir; problemin çözümü için kavramsal bir yaklaşım yöntemidir. Veri kalitesini, bütönlüğünü güvenliğini artırdığı; malzeme, harmanlığı ve hizalarını azalttığı için kullanılır.

Liste ve Bağlantılı Liste Veri Modeli

- Aynı hürmeye alt olan verilerin bellekte bir arada tutulması ilkesine dayanır. Veriler belirli bir düzende içeriğinde olabilir veya olmazabilir. Örneki olan tüm verilerin bir arada gelen sıradan tutulmalıdır.

- En yolda liste veri modeli bir boyutlu dizide üretilir ve tutulur.

- Bağlantılı liste (link list), elementların her bir değerlerine ek olarak bir de bağlantı bilgisinin kullanılmasına sahiptir. Bağlantı bilgisi bir sonraki elementin adresi niteliğindedir.

Ağ Veri Modeli

- Düğümlerden ve dallardan oluşur; düğümlerde verilerin kendileri veya bir kısmı tutulurken, dallar diğer düğümlere olan bağlantı ilişkilerini gösterir. Özellikle kümenin büyük olduğu ve arama işleminin çok kullanıldığı uygulamalarda etkin bir çözüm sunar.
- En üsteki düğüm kök(root), kendisine alttan bağlı olanların olmadığı düğüm yaprak(leaf), diğerleri de ana düğüm (internal node) olarak adlandırılır. Bir düğüme alttan bağlı düğümlere çocuk (child), isten bağlı düğüme de o düğümün ailesi (parent) denir. Veri tabanı, yapay zeka.

Graf Veri Modeli

- Aynı kümeye ait olan verilerin düğümleri, aylıklar (Keseler) ve burların birleştirilmesinden oluşur. Düğümler birleşme noktasını ayrıntıları da düğümlerin bağlantı ilişkisi gösterir. Verilerin kendileri veya bir kısmı her düğümlerde herde ayrıtları bilgi kısmında tutulabilir. Sosyal ağ analizi, ağ güvenliği, navigasyon.

Durum Makinesi Veri Modeli

- Bir sistemin davranışını tanımlamak ve ortaya çıkarmak için kullanılan bir yaklaşım şeklidir; işletim sistemlerinde, derleyici ve yorumlayıcılarla, kontrol amaçlı yazılımlarda sistemin davranışını durumlara indirger ve durumlar arası geçiş koşullarıyla sistem ortaya koyn. Gerçek zamanlı işletim sistemlerinde proses kontrolü, oyun programlama, robotik.
- Seklen yönlü graflara benzeyen; öncek birleşme nodlarını graf larda olduğu gibi düğüm olarak değil de durum, aylıklar da geçiş egrileri olarak adlandırılır. Durumlar arasında geçişler, sistemin o ana kadar ki durumlarına regis parametrelerle bağlıdır.

Veritabanında İlişkisel Veri Modelleri

- Veriler tablolar üzerinden kurulan ilişkilere dayanır.

Ağ Veri Modeli

- Hesaplanan ağ mimarilerinde, bilgisayar arasında eş katmanlar (Peer layers) düzeyinde veri akışını sağlayan dilim (segment) paket ve gerekçue koruyucu ortaya koyar ve iletişim için genelki davranışları tanımlar. Veri haberleşmesinde hemen hemen tüm mimariler katmanlı yapıdadır. OSI 1, TCP/IP (Transmission Control Protocol / Internet Protocol) 4, basısu modeli \Rightarrow katmanlıdır.

Yığıt ve Yağın \Rightarrow Elemanlarına erişim sınırlaması olan, liste uyarlı veri modeli:

Kuyruk \Rightarrow Elemanlarına erişim sınırlaması olan, liste uyarlı veri modeli:

Veri Yapıları

Dizi \Rightarrow Hızlı ekleme ve hızlı erişim - Sabit

Sıralı Dizi \Rightarrow Sıralamamış dizide göre daha hızlı arama, Sabit

Yığın \Rightarrow Son girer, ilk çıkar. Yavas

Kuyruk \Rightarrow İlk girer, ilk çıkar. Yavas

Bağlı Liste \Rightarrow Hızlı ekleme ve silme. Yavas arama

Hash Tablosu \Rightarrow Hızlı ekleme ve aranın bilindiğinde hızlı erişim.

Yavaş silme, yavaş erişim anahtar bilinmiyorsa, verimsiz.

Küne (Heap) \Rightarrow Hızlı eklene ve silme. Yavas. Basta en büyük öğe.

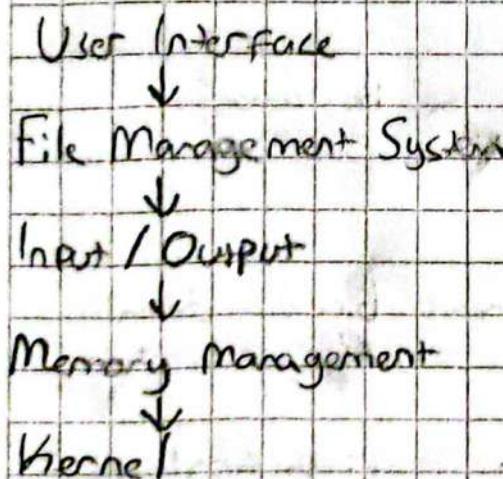
İkilli Ağacı \Rightarrow Hızlı arama, ekleme, silme - Silme algoritması karışık.

Graf \Rightarrow Gerçek dünya problemlerini modeller. Yavaş, karmaşık.

İşletim Sistemi - Operating System - Nedir?

- Uygulama yazılımları ile bilgisayar donanımı arasındaki bağlantıyı sağlayan özel yazılımlardır.
- Bilgisayarda programların çalıştırılmasını ve istenilen türde işlevlerin yerine getirilmesini işletim sistemi sağlar.
- Dosya ve dizin oluşturma, silme, kopyalama, taşıma, format gibi işlemler işletim sistemi tarafından gerçekleştirilir.
- Bilgisayarın çalışması için işletim sisteminin bilgisayara yükülmesi gereklidir. İşletim sistemi olmadan bilgisayar çalışmaz.

Operating Systems Layer



Operating Systems General Structure



- Kernel** => İşletim sisteminin kabiidir. Uygulamalar ve donanım seviyesindeki bilgi işlemleri arasında köprü görevi görür.
- Donanıma kullanıcıların ve uygulama yazılımlarının doğrudan erişimlerini sınırları ve düzenler. Bir çok sisteme kabuk ve gerekdeğ ayırmayı sadece hizmet eder.
 - Sistem kaynaklarının yönetilmesi görevleri arasındakidır.
 - I/O management, process management, memory management, device management, file management.

- Shell**) Kullanıcı komutları girdiğinde Kullanıcı veya uygulama aracılıkları tarafından gelen komutları işlemekten sorumludur.
- Girdidekten aldığı bildirimler ve girdideğin geçerli durumunu saygısına iletir.
 - Uygulama yazılımları, API ve Shell ile konuşur.
 - API, Shell'e dahil edilebilirler. API için SDK olarak bilinen geliştirme kılavuzları vardır.

Anayüz =) CLI (Command Line Interface)
GUI (Graphical User Interface)

- monoprogramming**) Birim zamanda bir görüntü ortam kurulur.
- Kullanıcı sistemin tüm kaynaklarını kullanabilir.
 - Kaynak alırma, bütünlük koruma gibi sorunlar yoktur.

multiprogramming) Sisteme çalışan herhangi bir iş beklenen durumuna gittiğinde işlençinin başka bir işe başlaması,
- işlenç hızı \gg I/O,
- Verimlilik artar

multitasking) Banyak görev için atanmış birdeki adının birdikte çalıştırılması.

Karma Yöntemi) En öncelikli, donanımdan veya yazılımdan kaynakları bir anda sonucunda işlençinin yürütüldüğü görevi birden fazla, ilgili, uygun, üretken birime hizmet verecek görevin anıltarlanmasına yol açan olaydır.

I/O Yönetimi) Girdidek sistemin sağladığı veri akımları ile birlikte I/O istekleri üst katmanların I/O istekleri ilişkilendirir.

File Management) Data üst düzeydeki yazılım katmanlarının simgesel olarak tanımlanan dosyalar ile sistemdeki fiziksel I/O birimleri arasındaki eşleşmeyi yapar. Mesnödir.

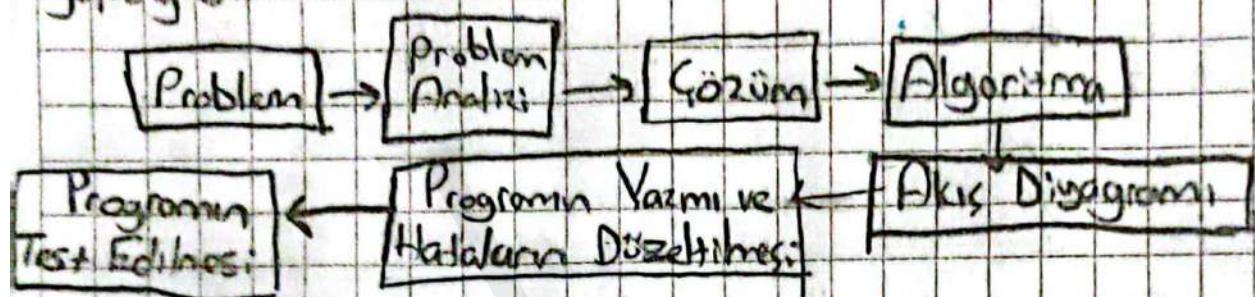
Deadlock - Ölümçü Kilitlenme \Rightarrow Proseslerin hiçbir zaman ele geçiremeyecekler, bir birime veya kaynakta ihtiyaç duyuları durumunda sürekli olarak askida kalmalıdır.

- Bir prosesin işinin bitmesi bir başka prosesin işinin bitmesini bekleyen bir durum oluşturuyorsa da olur.

Semafor \Rightarrow Genel olarak kaynakların kritik bölgelerde kilitlenme olmaması, işin kullanımın bir değişken türüdür. Üzerinde iki temel işlem hanımlıdır. Wait (S) semaforun içeriğini bir azaltır. Signal(S) semafor içeriğini bir artırır. Semafor değeri negatif ise bekleyen proseslerin olduğu aralığı gelir. Bu prosesler, Semafor değeri sıfır veya pozitif olana kadar beklenmeye devam ederler.

Program => Bilgisayarın bir işlevi yapması için tarafından komutlar zinciridir. Ardışık sınımler dizidir.

Programlama Dil => Bir makinenin davranışını kontrol etmek için kullanılabilen ve program üretimi için kullanılan yapay dil



1. Yazım Güvenilir Olmalıdır

Bazı kalite unsurları saglayamaması gerekmektedir.

a. **Yazılabilirlik** = Bir programlama dilinde gerçek hedeftaki problemleri gözterken insanlar olarak kullandığımız metanumaların programlama dilinde şifre olmaları.

b. **Okunabilirlik** = Yazdığımın kol. okurabilir olmalıdır. Sıral. blokklarından oluşmalı.

c. **Sıra Dışı Durumlar Karşılayılabilir** = Herhangi durumlarda program bunu karşılayabilmesi.

2. Yazılımlar Bakırma Evetli Olmalıdır

3. Yazım Verimli Çalışmalıdır

Tanıtımı

İlk program fiziksel olarak yazılıyordu. Yüksek voltajda 1, düşük voltajda 0 şeklinde. Bu şekilde programlar yazmak, sistem oluşturmak elektronik devrelerin baştan kurulması gerekiyordu. Bundan dolayı bazı karakterler geraçlarında yazılmıştır.

- 1957 yılında IBM düşük seviye FORTRAN'ı, tanıttı.
- 1959 yılında bu programlara dilinin özelliklerini alıp, gizli - açık gibii yani işlevler saglayan COBOL dilini.
- 1963 yılında, COBOL ve FORTRAN dilinin en iyi özelliklerini biraraya Pascal ortaya çıkardı, işaretçi (pointer) geldi.
- 1972 yılında C, Pascal dilinde, birçok hatayı gidererek ortaya çıktı. Windows - Unix bu yazılım dili ile yazıldı.
- C++ dil, nesne tabanlı yazılım dilidir.
- Sun Microsystems tarafından Java tanıtıldı ve C++ tabanlı GC ortaya çıktı.
- Microsoft, 2000 yılında .NET platformu ile 3'den fazla yazılım dilini aynı çatı altında topladı.

Program Dillerinin Sınıflandırılması

1. Genel Sınıflandırma

- Tümel Prog. D. Fortran, C, Cobol, Basic, Pascal
- Venije Yöndük P. D. LISP, APL, Snobol, Icon
- Mesnaye Yöndük P. D. Simula, C++, Java, VB

2. Uygulama Amacılarına Göre Sınıflandırma

- Bilimsel ve Müh. Dilleri Fortran, C, Pascal
- Sistem P. D. C, Assembler
- Veri Tabanı D. DBase, Clipper

- Yapay Zeka D. Prolog, LISP, Python
- Genel Amacı P. D. C, Pascal, Basic

3. Seviyelerine Göre Sınıflandırma

- | | | | |
|------------------------------------|--------------------------------|---|---|
| • <u>Düşük Seviye</u>
Assembler | • <u>Orta S.</u>
C, C++, C# | • <u>Yüksek S.</u>
Fortran, Basic, COBOL, Pascal | • <u>Cok Yüksek S.</u>
DBase, Clipper, VB, Paradox, Access |
|------------------------------------|--------------------------------|---|---|

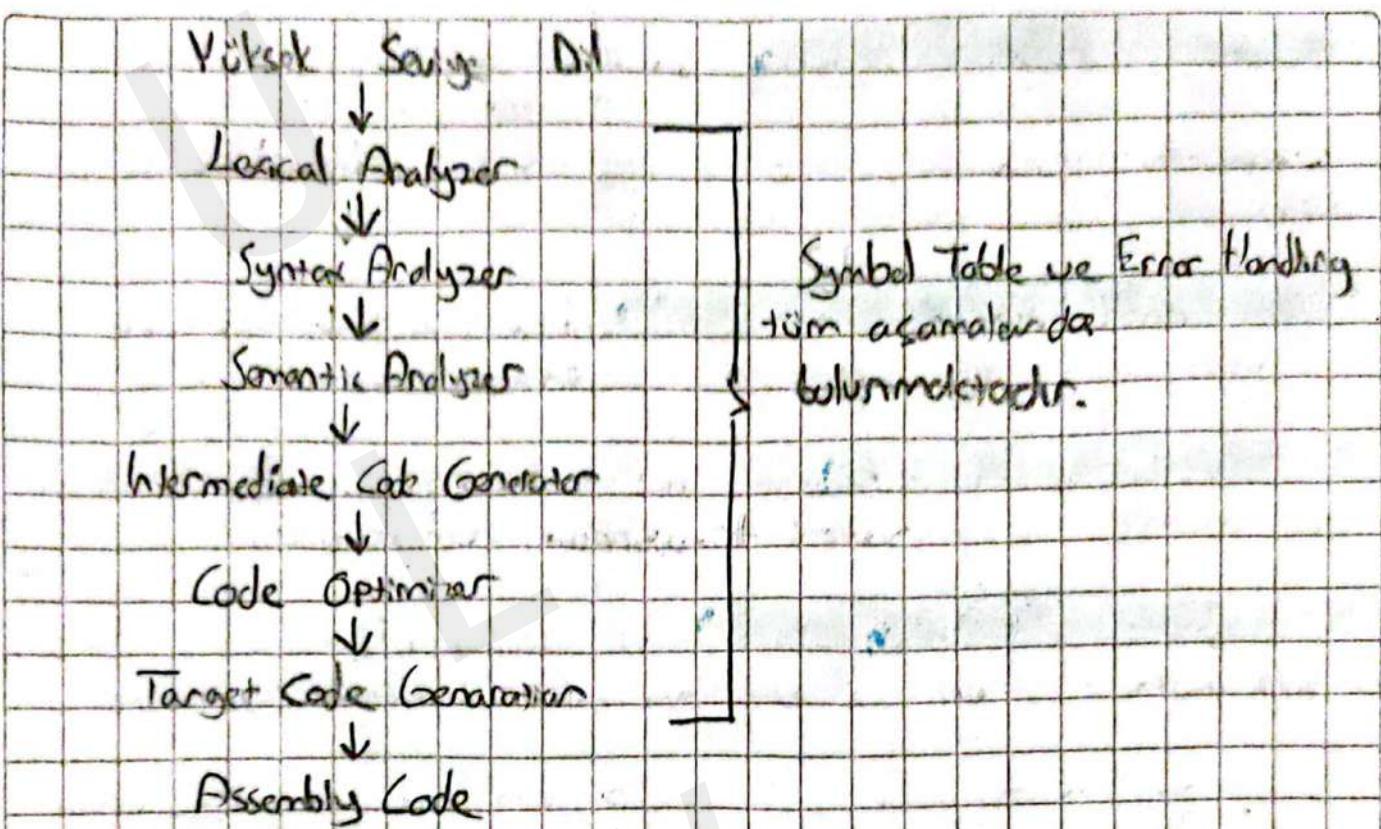
Nesne ve Yöndük Programlama

Gerek hayatta var olan veya programcı tarafından oluşturulan mantıksal varlıkların yazılıbilirlik - okurabilirlik güçlerini için gelen yazılım yöntemi.

- **Veri Soyutlama (Data Abstraction)** \Rightarrow Kullanıcı tarafından yer. ver. türlerini modelleyen sınıfların oluşturulmasıdır.
 - **Kalıtım (Inheritance)** \Rightarrow Oluşturulan bu sınıfların genişletilerek veya özelleştirerek yeni sınıflar oluşturulmasıdır.
 - **Cole Biçimlilik (Polymorphism)** \Rightarrow Aynı isimdeki işlevlerin değişik nesne grupları tarafından farklı algılanmasıdır.
- Yöndüklü Programlarda veri, prosedürler arası geçer. Fonksiyonlar arasında veri etkileşimi.
- Mesne Tabanlı Programlarda veri, nesneler arasındaki geçer.

Programlama Ortamları

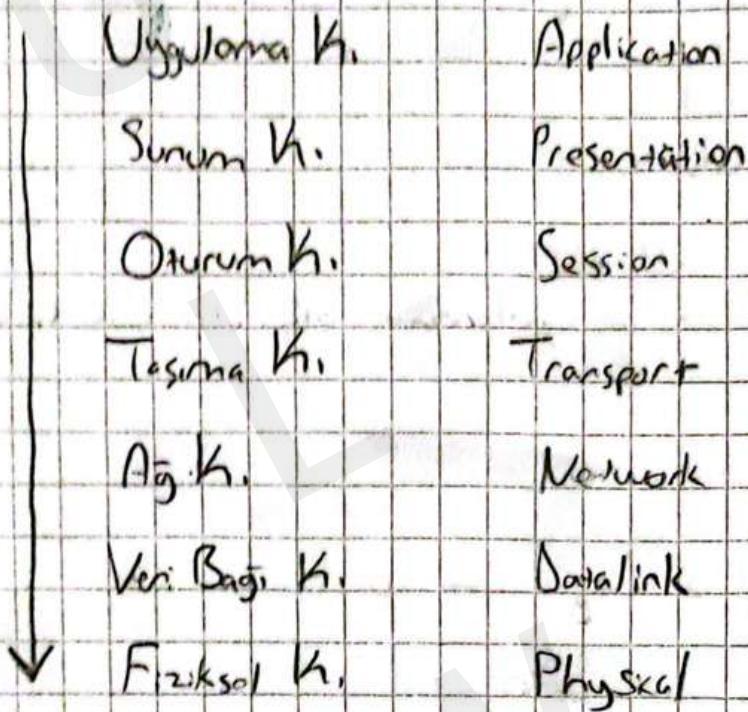
- **Editor (Editor)** \Rightarrow Kaynak Kodu oluşturmak ve gerektiğinde değişiklik yapmak için kullanılır.
- **Dergici (Compiler)** \Rightarrow Editörde yazılan kaynak kodunu makine koduna çevirir.
- **Kütüphane (Library)** \Rightarrow Nesne dosyalarından oluşur.
- **Bağlayıcı (Linker)** \Rightarrow Programın içeriği tüm nesne dosyalarının birleştirerek tek dosya haline getirir.
- **Hata Ayıklamacı (Debugger)** \Rightarrow Programın hataları ayıklayabilmesi için programın adım adım yürütülmesini sağlar.
- **Yorumlayıcı (Interpreter)** \Rightarrow Programın kaynak kodunu satır satır yürüten programlardır.



- **Sözdizimi / Vazımı (Syntax)** \Rightarrow Temel olarak bir dilde tanımlı olan öğelerin anlamlı bir dizim oluşturmasıyla ilgilenen bilimdir.
- **Antambilim (Semantics)** \Rightarrow Bir programınca dilinde bir ifadein ne anlama geldiğidir.
- **Ayarla Deyimi (Assignment Statement)** \Rightarrow Beltek adresinde değer atama
- **Tür Kontrolü** \Rightarrow Programlarda dillerinde yapılan her işlem öncesi hatalar önlenerek için verilerin tip kontrolü yapılır. compile-time veya run-time olarak yapılır. Bu farklılık static ve dinamik tipli dillerin temel ayırmadır.
- **Kontrol Deyimleri** \Rightarrow if-else, switch-case
- **Alt Programlar** \Rightarrow Tekrarlayan komutları bir fonksiyon içinde toplayıp çağırırız.

OSI Model

Open Systems Interconnection (Açık Sistem Aracılığıyla Modeli)



Uygulama(7)

(Remote Desktop Protocol)

- HTTP, HTTPS, FTP, SSH, SSL, RTP, Telnet
- Uygulama katmanı: Sağlamlıkla yüklenir

Sunum(6)

- ISO 8822, ISO 8823, ISO 8824, ITU-T T.73
- Uygulama Katmanı için standart bir arayüz sağlamak üzere ver. biçimlerini dönüştürür.

Öturum(5)

- SMB, ISO 8526, NFS, ISO 8527, ITU-T T.629
- Nerel ve uzak uygulama arasındaki bağlantıları oluşturur, yönetir ve sonlandırır.

(Paketlerin) RIP → Router Information Protocol (Mesajları, yönlendirme)
ICMP → Internet Control Message Protocol (Sınırma, sorun uyarı)
ARP → Address Resolution Protocol (IP adresi bilinmemişse MAC'yi bul)
MAC → Media Access Control
DNS → Domain Name Service (MAC ile IP'yi ilişler)

(Ulaşım)

Taşıma (L) (TCP-UDP)

- TCP (Transmission Control Protocol), UDP (User Datagram Protocol)
- Ağ üzerinde güvenilir taşıma ve okus doritimi sağlar.

Ağ (3) (IP)

- IP, IPv4, IPv6, ICMP, ARP, IGMP
- Manşetsel adreslerden ve yönlendirme etki alanından sorumludur.

Veri Roğ (2) (MAC)

- Ethernet, HDLC, Wi-Fi, Token ring
- Fiziksel adreslere ve ortam erişim yardımıcısı sağlar

Fiziksel (1)

- ISDN, Fiber Optik, RS-232, EIA-422, RS-485, EIA-685
- Aygıtlar için tüm elektriksel ve fiziksel özellikleri tanımlar.

TCP/IP Modeli

Amerika Savaṇı, Bütçesi tarafından heterojen ağlarda kesintisiz bağlantılı iletişim için geliştirildi.

Uygulama

Ulaşım (TCP-UDP)

Internet

Düğünden - Ağ (Host-to-Network)

OSI(7)
OSI(6)
OSI(5)

TCP/IP
Uygulam

OSI(4) } TCP/IP Transport

OSI(3) } TCP/IP Internet

OSI(2) } TCP/IP Network
OSI(1) } Access

Ağın Sınıflandırılması

- Coğraf. Koşullar;

- LAN, MAN, WAN

- Topolojilerine Göre;

- Bus, Ring, Star, Tree, Mesh

- Ortambanına Göre;

- OSI, TCP/IP

- Ethernet, TokenRing, FDDI, ATM

- İletim Yöntemleri

- Aktif (Poj. Cihazları)

* Modem, NIC, Repeater, Hub, Switch, Router

- Positif (Kablolar)

* Coaxial, UTP, STP, Fiber

LAN - Local Area Network - Yerel Alan Ağları

- Ev, okul, laboratuvar, iş yerlerinden gibi sınırlı coğrafi alanda bilgisayarları ve araların birbirine bağlan ağı.

MAN - Metropolitan Area Network - Metropoliten Alan Ağları

- LAN ağlarından daha büyük yapıldı
- Şehrin bir kısmını veya tamamını kapsar
- Kampüs ağları adı ile anılır

WAN - Wide Area Network - Geniş Alan Ağları

- Birden fazla cihazın birbirini ile iletişim kurmasını sağlayan fiziksel veya mantıksal büyük ağlardır.
- Yerel alan ağlarının birbirine bağlanmasını sağlayan çok geniş ağlardır. En geniş alan ağı Internet'tir.

Ağ Topolojileri

Bir ağdaki bilgisayardan nasıl yerlesceğini, nasıl bağlanacağını, veri iletiminin nasıl olacağını belirleyen yapıdır.

- Fiziksel T. \Rightarrow Ağın fiziksel olarak nasıl görüneceğini belirler
- Mantıksal T. \Rightarrow Bir ağdaki veri akışının nasıl olacağını belirler.

Ağ Döşemeleri (Cihazları)

Oaklımla fork./
Protokol Dönüşümü

← Gateway

Ağ

Router

Veri Bağlantısı

Köprü

Fiziksel

Repeater

Brouter

NIC - Network Interface Card (Ağ Aracı) (1-2.1)

- Fiziksel Adresi Sahiptir. (Mac Adresi - 48bit - 16'lı)
- Ağ adaptörü veya ağ kartı (Ethernet) kartı olarak adlandırılır.
- Veri paketlerini parçalara ayırp, birleştirir.

Repeater (Tekrarlayıcı - Yineleyici) (1. katman)

- Ağın genişletilmesinde kullanılır.
- Sinyallerin daha uzun mesafelere taşınmasına sağlar.

Hub (Dönüşüm) (1. katman)

- Star topolojisi.
- Network'in boş yeri yoksun ve güvenli değil.
- Kablolar ile ağ birimlerini birbirine bağlar.

Switch (Archer) (Göndere 2. katman) (Bazen 3.)

- Akıllı Hub
- Paket aktarımında MAC adreslerini kullanır. Dinamiktir.
- MAC tablosuna baktıgın geneksiz port kullanılmaz.

Bridge (Kapru) (2. katman)

- Mac adresi kullanır.
- Ağları bölmelere ayırip birleştirmede kullanılır.

Router (Yönlendirici) (3. katman)

- Ağın ve paketlerin yönlendirilmesini sağlar.
- LAN-LAN, LAN-MAN, LAN-WAN'da işlev yapar.
- IP adresi kullanır.

Gateway (Ağ Geçidi) (Tüm Kullanıcılar)

- Routerler üzerinde tanımlanır.
- Protokol dönüşümü veya haritalama aracığı sağlayan donanım veya yazılımdır.

Firewall (Ağ Duvarı)

- Yetkisiz her erişime engel olur ve paketleri filtreler.
- Kurallar, IP'ler, tanımlanır.
- Port'lar kullanılır.
- Sanal istemci sistemlerde kullanılır.
- WAF

UTP Kabloları → CAT5, CAT6 gibi → Ucuna RJ45 Konnektörleri takılır.

Protokoller

- Bilgisayarlar arası iletişimde kullanılan ağ silleridir. (kuralları)
- TCP/IP en sıkılır.

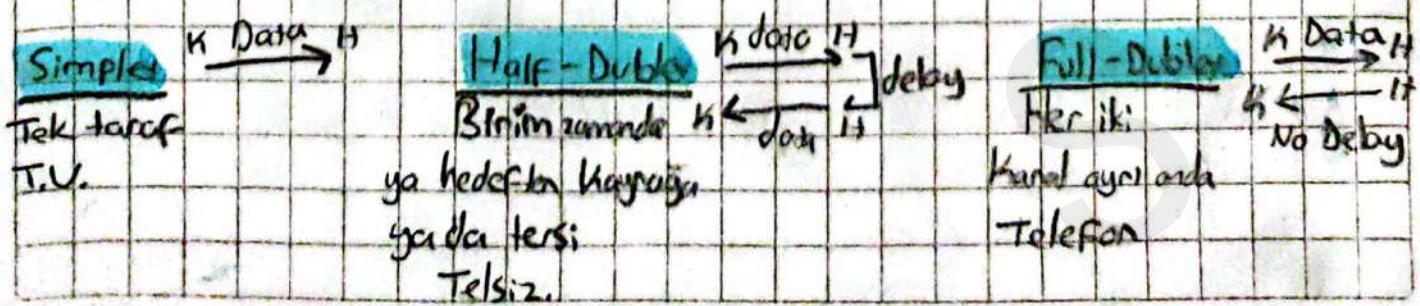
Modem

Bilgisayarın internete bağlanması ve birbirlerinle haberleşmesi sağlar.

Modulator → Bilgisayardan gelen digital veriyi analog sinyale dönüştürme

Demodulator → Analog sinyallere digital veriye dönüştürme

Ver. Aktarımı



Ver. Tabanının Tasarlanması \Rightarrow Ver. tabanını oluşturucu verilen tip ve varlıklarının belirlenmesidir.

Ver. Tabanının Olusutlaması \Rightarrow Ver. için yer belirlenmesi ve sıklıkla ortamına verilerin yüklenmesidir.

Ver. Tabanı Üzerinde İşlem Yapmak \Rightarrow Belirli bir veri üzerinde soruluma yapma, meydana gelen değişikliklerin yansıtmak için veri tabanının güncellendirilmesi ve rapor üremesi.

Verinin Etkin ve Sınırlılığı \Rightarrow Veri tabanına yenilikçi eklemek, ekleteri, çağırmak ve gerekli düzenlemeler, doldurma ve silme işlemlerini yapmak gibi işlemlerin gerçekleştirilebilmesi ifade eder. Veri T.Y.S. verinin geni kognitibilitesini sağlar.

Ver. Tabanı Genişletme \Rightarrow Veri veri eklemeye ve yeni kayıtlar düşürmeye

Hierarsik Veri Tabanı \Rightarrow Ağac - şeklinde

İşlekSEL Veri Tabanı \Rightarrow Sadece birbirini ile ilişkili olan verilerin tutulması.

***Attribute - nitelik** \Rightarrow Veri tabanındaki baslik - özellikler

ANSI SQL
American Standardis

T-SQL
ANSI SQL
Gelenisi

PL-SQL
Oracle

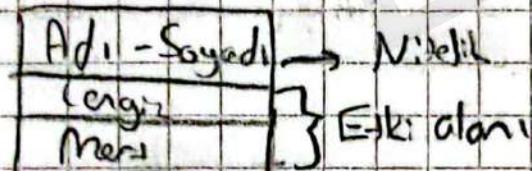
PostgreSQL
MySQL
Açık Kaynak

Varlık \Rightarrow Aynı türden her birinden ayrı edilebilir
her şey.

Varlık Kümesi \Rightarrow Aynı türden her birinden farklıdır
küme.

Nitelik \Rightarrow Varlık kümelerinde varlıkların özelliklerini gösteren
ve varlıkların birbirinden ayrıntılarını tanımlar.

Eski Alan \Rightarrow İler niteliğin bir etkili alanı (domain) vardır.
Niteliğin olabildiği tüm değerleri içermesi gereklidir.



Tümelleş Nitelik \Rightarrow Bir nitelik kümelerin her birini varlık
niteliği elde edebilir gerek, yani niteliğe denir (yaş)

Birleşik Nitelik \Rightarrow Birbirinden fazla nitelik birleştirilerek
yen bir nitelik oluşturmak. (mahalle, caddesi adres)

İlişki \Rightarrow Varlıklar arasındaki ilişkileri denir.
Öğrenci - ders, işçi - ürün - makine

İlişki Kümesi \Rightarrow Aynı türden ilişkilerin oluşturduğu küme deur.

$$\begin{aligned} E_1 &= \{ \text{Ali}, \text{Mehmet} \} & E_2 &= \{ \text{Matematik}, \text{Fizik} \} \\ R_1 &= \{ \text{Ali}, \text{Matematik} \} & R_2 &= \{ \text{Mehmet}, \text{Fizik} \} \end{aligned}$$

Rol \Rightarrow Birbirinden ilişkilerin varlıklarından her birinin
ilişkideki islevine varlığın rolü deur
Öğrenci: ders, ders ise öğrenci tarafından alınır.



Anahatlar \Rightarrow Varslık kümelerindeki varlıklar ya da bir ilişkili kümeleri içindeki ilişkilere ağırlık etmek için kullanılır.

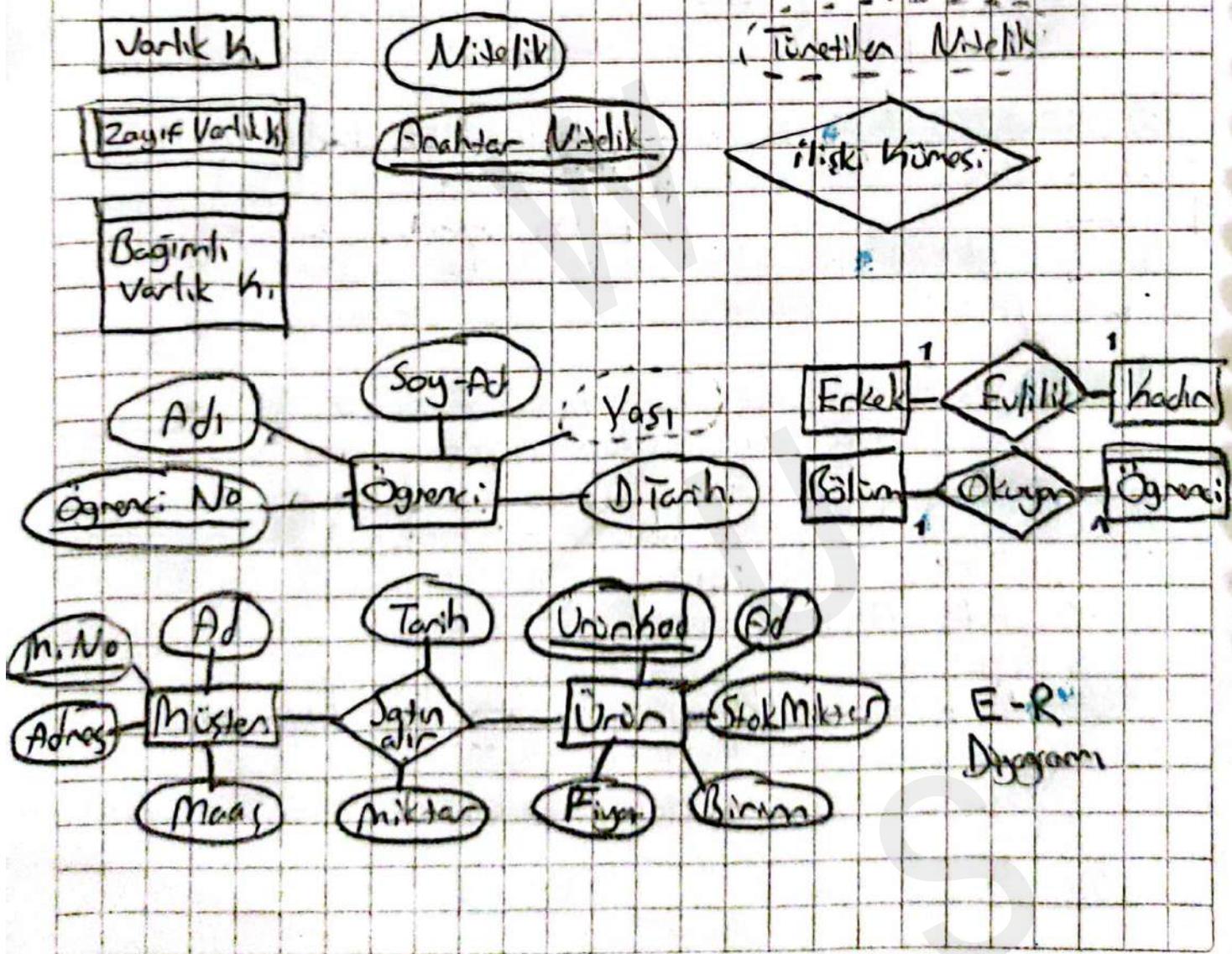
Süper Anahat \Rightarrow İhümede ağırlık edilgərsə

Ağır Anahat \Rightarrow Altkürede ağırlık edilgərsə

Göktü \Rightarrow Anahat bulmak kolaysa

Zayıf \Rightarrow Anahat bulmak zorlsa

* Zayıf kümələr, boşca bur kümə ilə ilişkilendirdiğən anahatlar eklənilip zayıf kümə güdünləndirilebilir.



Bire-Bir ilişkili TablosuE1 Bölüm (bölümNo, bölümAd)E2 Personel (sciNo, ad, soyad, adres, maas, bölümNo)Bire-Gok ilişkili TablosuE1 Öğrenci (öğrenciNo, ad, soyad, bölümNo, KayıtTarihi, mezuniyet)E2 Bölüm (bölümNo, bölümAd)İlgili Kütühesinden
geliç.Gok-Cıck ilişkili TablosuE1 Müşteri (müşteriNo, adres, telefon)E2 Ürün (ürünNo, ürünAd, birimFiyat, stokMiktarı)R1 SatınAlır (SatınAlırNo, müşteriNo, ürünNo, tarih, miktar)

İlgili Kütühegi:

SQLYokal Sorgulama Dil: Yazılım Dil'i Değişdir

SELECT → Veri seçme, Sorgulama

INSERT → Veri Kayıt ekleme

UPDATE → Verilen değiştireme

DELETE → Kayıt(ları) silmemizi sağlar

Domain
Alan - ADJ Nedle
adı, soyadı Öğrenci

! * bütün anlamında

SELECT * FROM personel WHERE ad = 'Ahmet'
=, >=, <=, >, <

Sayı
Bölümüze.

SELECT * FROM ucretler WHERE aylık-ucret >= 30000
ORDER BY aylık-ucret

Sıralama
yaptırı
aylık-ucret'e
göre sırala

SELECT * FROM öğrenciler WHERE öğrenci-No
BETWEEN 200 AND 400;

Arasındaki
Değerleri;
Ara

ASC

ORDER BY

Sırası, kılınılıp
küçükten büyüğe
sıralar

DESC

ORDER BY

Sırası kılınılıp
büyükten küçükçe
sıralan.

SELECT * FROM personel WHERE adres LIKE '%İstanbul%'

% İstanbul → Sonunda geçen
İstanbul% → Başında geçen

Başında
ve Sonunda
geçenlerdir.

MAX, MIN, SUM, AVG
 ↓ ↓ ↓ ↓
 max değer min değer toplama ortalaması

SELECT MAX(kolsuz_adi) FROM tablo;

CREATE TABLE Musteri:

```

    (
        mus_id char(4) NOT NULL → Zorunlu
        mus_adi varchar(40) NULL,
        il varchar(20) NULL,
        ulke char(2) NULL,
        adres varchar(30) NULL → Zorunlu Değil
    )
  
```

CREATE DATABASE Personel

Nesne değişimi sağlar.

ALTER TABLE Musteri:

ADD tel varchar(20) NOT NULL

DROP TABLE MUSTERI

Bir nesneyi siler

INSERT INTO CARIANA

(kodu, adi, grubu, adresi) VALUES ('600', 'FARUK', 'A', '36 Sokak')

UPDATE MUSTERI

SET bakiye = bakiye * 1.5

WHERE ad = 'Muhammet';

DELETE * FROM muster WHERE bakiye <= 10000

DELETE muster

WHERE grup = 'ihal'

Bilgisayar Ağları (Network)

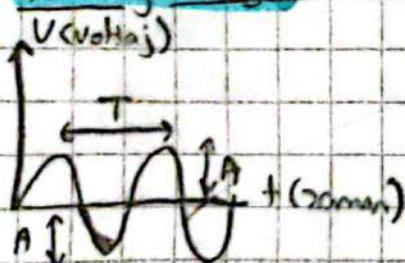
İki veya daha fazla bilgisayarı kablosuz veya kabloşur iletişim araçları, üzerinden yazılım ve donanım bileşenleri ile birlikte bağlanması ile meydana gelen sistem bütünüdür.

- Kaynakların ve bilginin paylaşılması ve kişiler arasında iletişim sağlamak
- Cesitli yöntem ve teknolojiler kullanarak bilgisayar ağları oluşturur.
- kaynakların etkin paylaşımını sağlayıp bilgi akışını hızlandırarak verimli bir iletişim ortamı sunar.

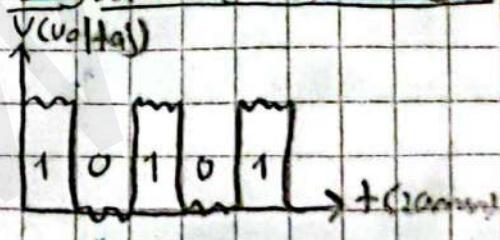
İletim Sinyal

- Sayısal iletişim ikili tabanda kodlanmış bilgi veya verinin sistemler arasında aktarılmasını kapsar
- Cihazlar analog mantıkla çalışır dörusümlü yapılır.

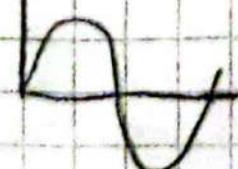
Analog Sinyal



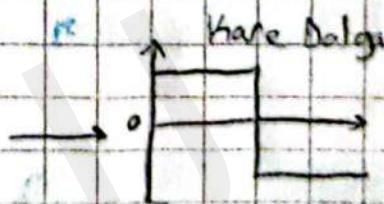
Sayısal (Digital) Sinyal



ANALOG S.



DİJİTAL S.



Bant Genişliği => Borunun genişliği gibi düşünülebilir. Boru iletişim hattındaki pompalar, musluk, direğiz, bağlantı cihazlarını beraberler.

Networkdeki veri ve paketler suya benzettir. Boru hattı ne kadar geniş olursa o kadar su iletilir. Yani bağlantı cihazları ne kadar fazlaysa o kadar paket gönderebiliriz.