Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

Kurs Hedefleri:

Önerilen müfredat, veritabanları teorisindeki temel konuları ele almaktadır:

Veritabanlarının kavramsal modeli, belirlenen sorunları çözen ana görevleri içerir;

-Veritabanının mantıksal modeli, geliştirilen veritabanının temelini oluşturan farklı veriler arasındaki mantıksal bağlantıları içerir;

-Veritabanlarının fiziksel modeli, bunların fiziksel olarak gerçekleştirilmesidir (konum, bağlantılar ve bilginin yönetimi);

Dersin amacı, bir DBMS'den bilgilerin nasıl verimli ve etkili bir şekilde organize edileceği, sürdürüleceği ve alınacağı konusuna vurgu yaparak veritabanı yönetim sistemlerine bir giriş sunmaktır.

Kurs, farklı veritabanı modellerini dikkate alır ancak veri modelleri, veritabanı mimarileri ve veritabanı manipülasyonları dahil olmak üzere ilişkisel sistemlerin temellerinin anlaşılmasını vurgular.

Ders ayrıca sorgu işleme ve optimizasyona yönelik algoritmaları ve biyoinformatiğin farklı alanlarındaki bazı uygulama veritabanlarını da içerir.

Öğrenme Çıktıları:

Dersin sonunda öğrenci şunları yapabilir:

-Gerçekçi bir problem spesifikasyonundan bir 'Varlık-İlişki' modeli üretir.

-Bir veritabanının kavramsal şemasını açıklar.

-Bir veritabanının fiziksel şemasını açıklar.

-Veritabanı şeması oluşturmak için resmi tasarım tekniklerini kullanır.

-Normalleştirme tekniklerini uygular.

-Mantıksal yapıyı hazırlar.

-Veritabanını mantıksal şema modelinden tasarlar ve uygular.

-Tasarlanmış bir veritabanını yönetir.

-İlişkisel cebiri kullanarak veritabanını düzenler.

-SQL kullanarak veritabanını düzenler.

-İlişkisel ortamın göreceli yararlarını tartışır.

-İlişkisel veritabanı yönetim sistemlerinin temel unsurlarını açıklar.

-İlişkisel veri modeli, varlık-ilişki modeli, ilişkisel veritabanı tasarımı, ilişkisel cebir ve SQL ile ilgili temel kavramları açıklar.

-Veritabanı tasarımı ve uygulama ekibinin değerli bir üyesi olarak çalışın.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

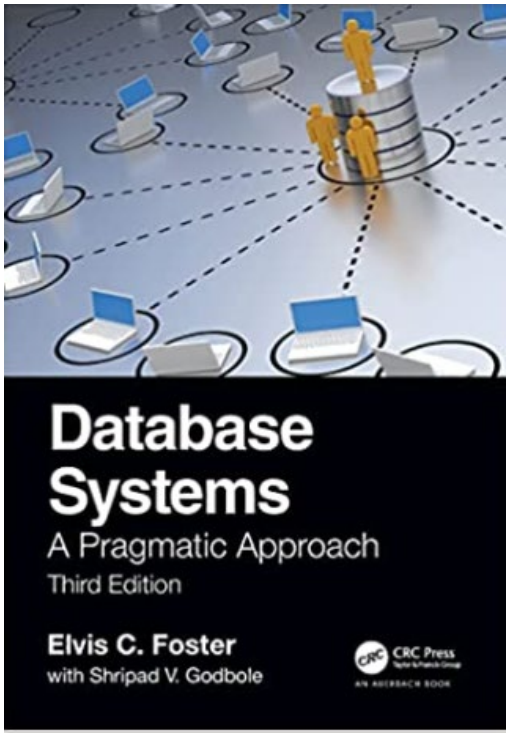
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

- **Modül 1. Veritabanlarının Teorik Temelleri**
- **Konu 1. Temel kavramlar**
- Ders 1. Veritabanına giriş

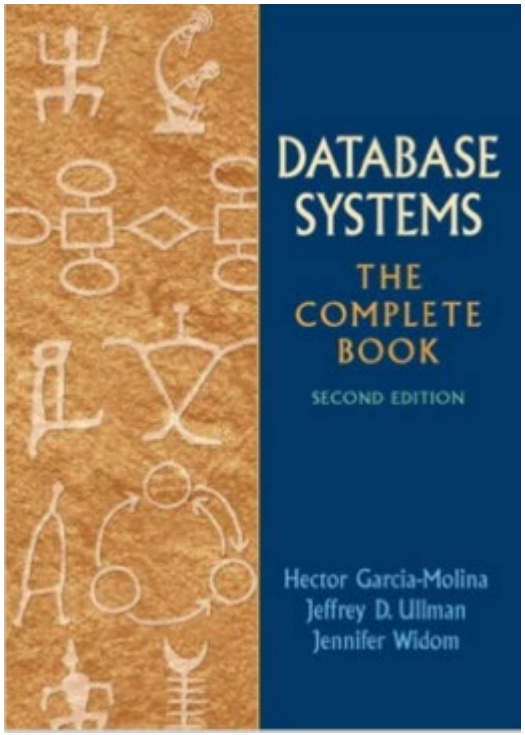
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANINA GİRİŞ



Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.



Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

Azalov, P. (1991) Database, Sofia.

Co-funded by the Erasmus+ Programme of the European Union

Giriş

- Veritabanı (DB) terimi 1960'ların başında ortaya çıktı, ancak onu verilerin bilgisayarda işlenmesiyle bağlantılı olarak ilk kez kimin kullandığı bilinmiyor.
- C. Bachmann'ın 1964 ve 1965 tarihli iki yayınında yansıtılan fikirleri de temeldir. Onlarla birlikte bilişimde bugün "Veri Tabanı Yönetimi" olarak bilinen yeni bir alanın ilk problemlerini ortaya attı.

Giriş

- Aynı zamanda, başka yerlerden uzmanlar veritabanı yönetimi sorunları üzerinde çalışmaya başladı ve büyük yazılım şirketleri bu alanda liderlik çabalarında hızla ilk modern veritabanı yönetim sistemlerinin (DBMS) program uygulamasına yöneldiler.
- DBMS, büyük miktarda veriyi oluşturmak ve verimli bir şekilde yönetmek için güçlü bir araçtır. Veriler gerektiği sürece muhafaza edilmeli ve saklanmalıdır. Ders 2'de DBMS derinlemesine ele alınacaktır.

VERİTABANI NEDİR

- Veritabanı teriminin kısa ve kesin bir tanımını vermek zordur ve belki de bu, onun çeşitli yorumlarındaki kaosu açıklamaktadır.
- Kavramın süregelen gelişimi de farklı görüşlerin oluşmasına katkıda bulunmuş olabilir.
- Bu soruyu ilk düşünen uzmanların algısı şu şekildedir:
- - Veritabanı (VT) bir dosya koleksiyonudur;

VERİTABANI NEDİR

- *Dosya bir dizi kayıttır;*
- *kayıt bir veya daha fazla anahtar ve veriden oluşur.*

Bu tanım hala birçok DBMS (Veritabanı yönetim sistemi) geliştiricisi ve kullanıcısı tarafından kabul edilebilir.

Ancak her veritabanı için talep edilen en önemli özelliği tasvir etmez.

Önemli olan, veritabanının entegre bir veri kümesi olarak oluşturulması gerektiğidir.

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANI NEDİR

- Öncelikle VT'nin kullanılma ve yönetilme şeklini ifade etmek için kullanılan VT'nin bu özelliğinin derin bir anlamı vardır.
- Bir VT'nin entegre bir veri seti olması, pratikte sıklıkla görüldüğü gibi sadece bir uygulamaya (kullanıcıya) yönelik değil, farklı uygulamalara (kullanıcılara) yönelik verileri içerdiği anlamına gelir.
- Ancak bu, her kullanıcının veri setinin tamamını kullanması veya diğerlerinin çalışmalarını bilmesi gerektiği anlamına gelmez.

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANI NEDİR

- Çoğu zaman, her kullanıcı veri tabanının yalnızca belirli bölümleriyle ilgilenir ve veri almak ve işlemek, veri tabanına yeni veri girmek, mevcut verileri güncellemek veya kaldırmak için uygulama programlarını kullanır.

- **Farklı uygulama programlarının aynı veritabanı verileri alt kümesini kullanabileceğini belirtmek de ilginçtir.**

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANI NEDİR

- **Bir örnek düşünün**: Bir kurumda “İnsan Kaynakları”, “Muhasebe”, “Araştırma Projeleri” ve “Uluslararası İşbirliği” bölümlerinin ihtiyaçlarına yönelik dört bilgisayar sistemi kurulmuştur.
- Bu sistemlerin her biri bir veri dosyasını destekler. Bu dosyaların alanlarından bazıları bir tabloyla temsil edilir:

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANI NEDİR

File "Personnel"	File "Salary"
1. Office number	1. Office number
2. Name of employee	2. Name of employee
3. Department	3. Department
4. Date of commencement of work	4. Duration of employment
5. Education	5. Base salary
6. Speciality	6. Family status

File "Projects	File "Business trips"
1. Task	1. Office number
a) Number	2. Name of employee
b) Name	3. Department
2. Task manager	4. Type of business trip
a) Office number	5. Destination
b) Name of employee	6. Number of days
c) Department	7. Task
d) Position	
3. Period	
a) Date of beginning	
b) Date of ending	
4. Financing organization	
5. Periods	
6. Lead organization	
7. Way of reporting	

VERİTABANI NEDİR

- Her dosyanın alanlarının oldukça az bir kısmı diğer dosyalarda bulunur. Doğal olarak bu, rahatsızlıklara yol açar:

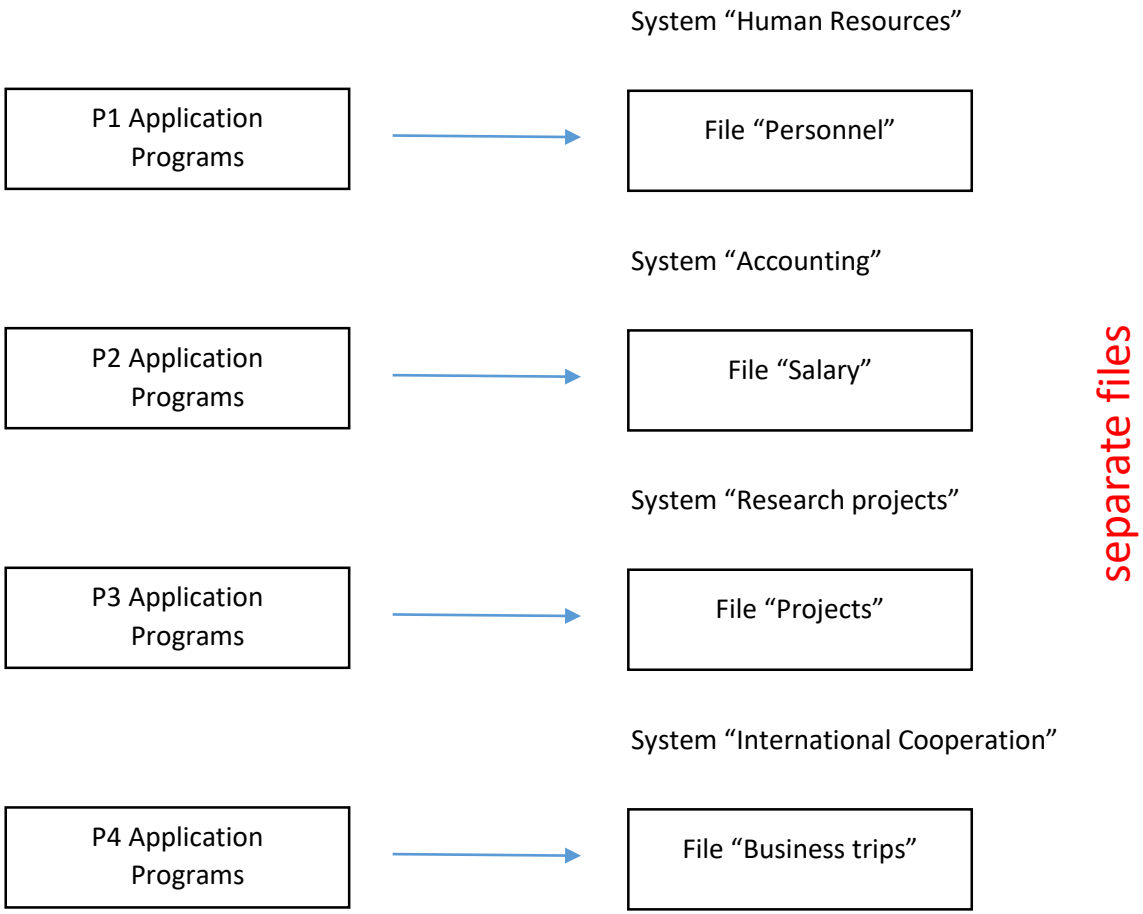
1. Veri girişinin farklı dosyalara birden fazla kopyalanması için **ekstra kaynak harcaması**. Örneğin A Kişisi işe başlıyorsa bu durumun “Personel” ve “Maaş” dosyalarına yansıtılması gerekmektedir. Ancak Çalışan A'nın daha sonra belirli bir görevin yöneticisi olması durumunda, "Personel" dosyasında kendisine ilişkin bazı bilgilerin "Proje" dosyasına da girilmesi gerekir.

VERİTABANI NEDİR

2. Herhangi bir çalışanın tarihinin değiştirilmesi, bireysel dosyalarda birden fazla değişiklik yapılmasını gerektirecektir. Bir yandan bu, yine gereksiz fon harcamalarının olacağı anlamına geliyor. Öte yandan, bireysel dosyalarda değişiklik yapmak genellikle bir noktada bireysel sistemlerin verilerinde tutarsızlıklara yol açar. A Çalışanı kurumdan ayrılmış olabilir ve bu durum Personel dosyasına yansımış ancak Bordro dosyasına yansımamış olabilir. Bir çalışanın evlilik nedeniyle soyadını değiştirmesi ve bir süre için Personel ve Maaş dosyalarında iki farklı isimle görünmesi mümkündür.

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANI NEDİR



Co-funded by the Erasmus+ Programme of the European Union

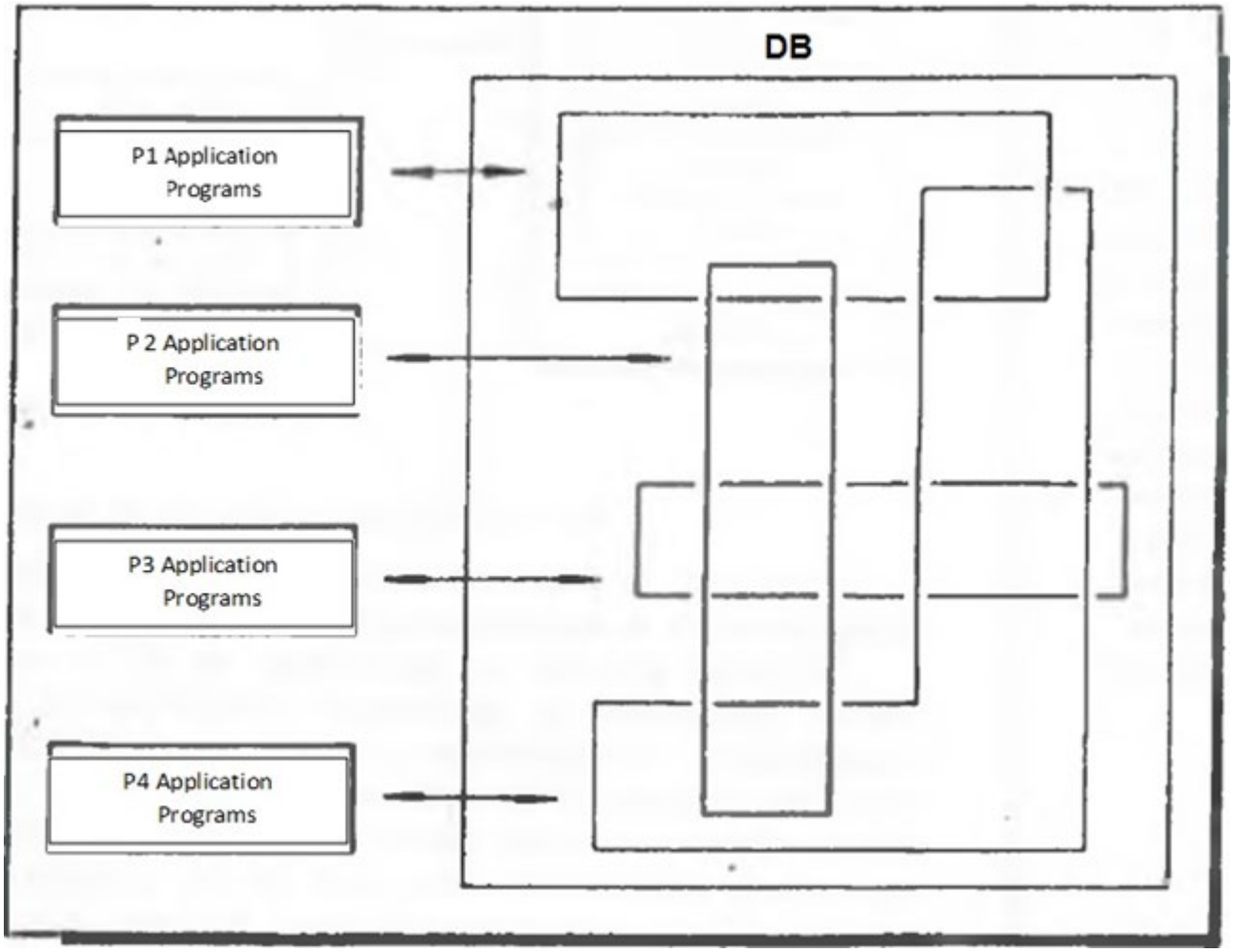
VERİTABANI NEDİR

- Dört sistem (yukarıdaki şekil) çalışmasında yukarıda belirtilen zayıflıkları gidermeye yönelik mantıklı bir çözüm, depoladıkları verileri tek bir veri arşivinde **birleştirerek (entegre ederek)** veya sıklıkla söylendiği gibi, entegre bir veritabanı oluşturarak işlerini yeniden düzenlemektir.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANI NEDİR



integrated set of data

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANI NEDİR

- Birleşik bir veri arşivi oluşturmayı ve merkezi olarak yönetmeyi amaçlayan yaklaşımın birçok **avantajı** var.
- Veritabanının sürekli genişletilmesi ve değiştirilmesi için koşullar yaratmaya yönelik mevcut fırsat vurgulanmalıdır.

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANI NEDİR

- Bu yaklaşımın amacı, mevcut uygulama programlarının işlevlerini aksatmadan, basit araçlarla veritabanında genişletme ve değişiklik yapabilme yeteneğinde aranmalıdır ve bu, herhangi bir veritabanının oluşturulmasında en temel husus olmalıdır.
- Veritabanı, uygulamalarının zaman içinde değişmesine ve genişlemesine izin veren, dinamik ve sürekli gelişen bir veri kümesi olmalıdır. Bu konulara ilişkin tartışma, sözde **veri bağımsızlığı** meselesine indirgeniyor.

VERİTABANI NEDİR

- **Veritabanı -** Kolay ve hızlı erişime, gözden geçirmeye, aramaya izin verecek ve bilgilerin tekrarını en aza indirecek şekilde yapılandırılmış bir veri kümesi.
- Veritabanlarının bir özelliği, verilerin yazılımdan bağımsız olmasıdır. Bu, onları hem farklı programlar tarafından hem de farklı zaman dilimlerinde kullanım için çok yönlü hale getirir.

VERİ BAĞIMSIZLIĞI

- **Örneğe geri dönelim**. **P1** uygulama programlarında "Ofis Numarası" alanının değerinin negatif olmayan bir tam sayı olduğunu, **P2** uygulama programlarında ise alanın bir karakter dizisi olarak tanımlandığını varsayalım.
- Veriler tek bir arşivde birleştirildiğinde programların “Ofis numarası” alan tipine bağımlılığı nedeniyle çelişkilerin ortaya çıkması doğaldır.
- Bu çelişkiyi aşmanın yolları var ve bunlardan biri de **P1** ve **P2** programlarında uygun değişiklikleri yapmak.

Co-funded by the Erasmus+ Programme of the European Union

VERİ BAĞIMSIZLIĞI

- Ancak bu yaklaşım tatmin edici değildir çünkü hem ek harcama gerektirmekte hem de verilerdeki yeni değişiklikler yine ilgili programlarda değişiklik yapılmasını gerektirecektir.
- Programlar ve veriler arasındaki bağımsızlık iki taraflı olmalıdır.
- Bu, harici bellekteki verilerin organizasyonuyla ilgili değişikliklerin uygulama programlarında değişikliklere yol açmaması gerektiği ve tam tersine, kullanılan verilerin açıklamasıyla ilgili kullanıcı programlarında yapılan değişikliklerin, verilerin veritabanında depolanma şeklini etkilememesi gerektiği anlamına gelir.

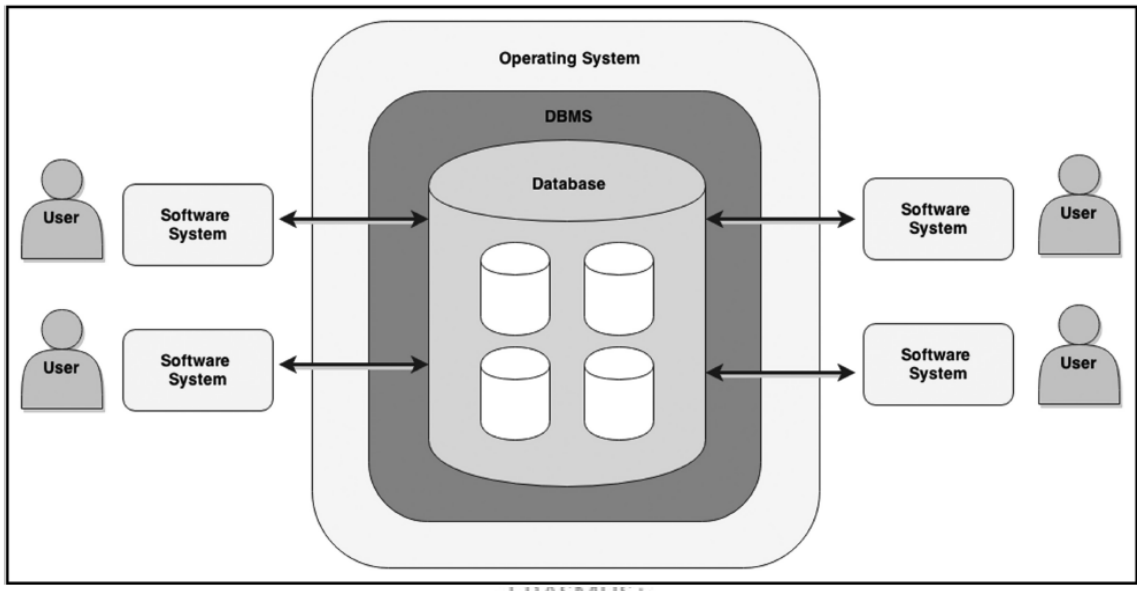
Veritabanı Sistemi (VTS): Tanımlar ve Gerekçe

- Bir veritabanı sistemi (VTS), genel amacı bilgiyi korumak ve gerektiğinde kullanılabilir hale getirmek olan bilgisayarlı bir kayıt tutma sistemidir. Veritabanı genellikle ilgili verileri bir bilgisayar sisteminde saklar.
- Bir **VTS**'nin bileşenleri şunları içerir:
- ■ Donanım ve işletim sistemi
- ■ Veritabanı yönetim sistemi
- ■ Veritabanı
- ■ İlgili yazılım sistemleri ve/veya uygulamalar
- ■ Kullanıcılar — teknik kullanıcılar ve son kullanıcılar dahil

Veritabanı Sistemi (VTS): Tanımlar ve Gerekçe

- Veritabanı kullanıcıları, yazılım sistemleri/uygulamaları ile iletişim kurar ve bunlar da (programlama arayüzü aracılığıyla) Veritabanı yönetim sistemi ile iletişim kurar. Veritabanı yönetim sistemi, verileri depolamak ve/veya veritabanından veri çıkarmak için işletim sistemiyle (bu sistem de donanımla iletişim kurar) iletişim kurar; bu Şekilde gösterilmiştir:

Simplified illustration of a DBS



Veritabanı Sistemi (VTS): Tanımlar ve Gerekçe

- Veritabanları yazılım mühendisliği için gereklidir; birçok yazılım sisteminin, son kullanıcı için şeffaf olacak şekilde sürekli erişilen temel veritabanları vardır.

Software Category	*Database Need*
Operating systems	A sophisticated internal database is needed to keep track of various resources of a computer system including external memory locations, internal memory locations, free space management, system files, and user files. These resources are accessed and manipulated by active jobs. A job is created when a user logs on to the system and is related to the user account. This process (also called a job) can in turn create other jobs, thus creating a job hierarchy. When you consider that in a multi-user environment, there may be several users and hundreds to thousands of jobs, as well as other resources, you should appreciate that underlying an operating system is a very complex database that drives the system.
Compilers	Like an operating system, a compiler has to manage and access a complex dynamic database consisting of syntactic components of a program as it is converted from source code to object code.
Information systems	Information systems all rely on and manipulate related databases, in order to provide mission-critical information for organizations. All categories of information systems are included. Common categories include (but are not confined to) decision support systems (DSS), executive information systems (EIS), management information systems (MIS), Web information systems (WIS), enterprise resource planning systems (ERPS), and strategic information systems (SIS).
Expert systems	At the core of an expert system is a knowledge base containing cognitive data, which is accessed and used by an inference engine, to draw conclusions based on input fed to the system.
...	....

Veritabanı Sistemi (VTS): Tanımlar ve Gerekçe

- Pazarda rekabet eden şirketlerin, görev açısından kritik verilerini ve diğer önemli verilerini depolamak ve yönetmek için veritabanlarına ihtiyacı vardır.
- Çağdaş veritabanı sistemleri olmasaydı hayat nasıl olurdu? Yeterince yaşlı birini tanıyorsanız, ona dosya dolapları, elle yazılmış kayıtlar veya daktiloda üretilen belgeler gibi bir çağ hakkında sorun. O zamanlar hayat çok yavaştı ama normaldi.
- Bir veritabanı sisteminin bilgisayar bilimini ilgilendirmesi gereken çeşitli birincil ve ikincil amaçları vardır.

Veritabanı Sistemi (VTS) : Sonuç

Bir veritabanı sisteminin temel amaçları aşağıdakileri içerir:

- Güvenlik ve koruma — yetkisiz kullanıcıların önlenmesi; işlemler arası girişime karşı koruma
- Güvenilirlik — istikrarlı, öngörülebilir performansın güvencesi
- Birden fazla kullanıcının kolaylaştırılması
- Esneklik – çeşitli yöntemlerle veri elde etme ve eylem gerçekleştirme yeteneği
- Veri erişimi ve veri değişimi kolaylığı
- Doğruluk ve tutarlılık
- Açıklık - belirsizliği önlemek için verilerin standartlaştırılması
- Beklenmedik isteklere cevap verebilme yeteneği
- Yatırımın korunması — genellikle yedekleme ve kurtarma prosedürleriyle sağlanır
- Veri çoğalmasının en aza indirilmesi — yeni uygulama ihtiyaçları, yeni dosyalar ve programlar oluşturmak yerine mevcut verilerle karşılanabilir
- Kullanılabilirlik- Veriler ihtiyaç duyulduğunda kullanıcılara sunulur.

Veritabanı Sistemi (VTS) : Sonuç

Yukarıdakilere ek olarak, aynı derecede önemli olduğu iddia edilebilecek bazı ek hedefler de vardır:

- Fiziksel veri bağımsızlığı — depolama donanımı ve depolama teknikleri uygulama programlarından yalıtılmıştır
- Mantıksal veri bağımsızlığı - veritabanına erişen mevcut uygulama programlarını etkilemeden veri öğeleri eklenebilir veya çıkarılabilir veya genel mantıksal yapı değiştirilebilir
- Artıklık kontrolü — genel kural, verileri minimum düzeyde depolamak ve kesinlikle gerekli olmadıkça bu depolamayı birden fazla yere kopyalamamaktır
- Bütünlük kontrolleri — aralık kontrolleri ve diğer kontroller geçersiz verilerin girmesini önlemelidir
- sistem
- Açık veri tanımı — veritabanında saklanan her veri öğesini açık bir şekilde tanımlayan bir veri sözlüğü tutmak gelenekseldir
- Uygun şekilde kullanıcı dostu arayüz — grafiksel, komut tabanlı veya menü tabanlı
- Ayarlanabilir — uygulama programlarını değiştirmeden performansı artırmak için veritabanını kolayca yeniden düzenleme
- Performansı artırmak için otomatik yeniden düzenleme veya geçiş

Sorular ve alıştırmalar:

1. Veritabanı nedir ve faydası nedir?

2. Veritabanının birkaç avantajını açıklayın.

3. Veritabanı yönetim sistemi nedir ve özellikleri nelerdir?

4. Veritabanı Sistemi nedir ve bileşenleri nelerdir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

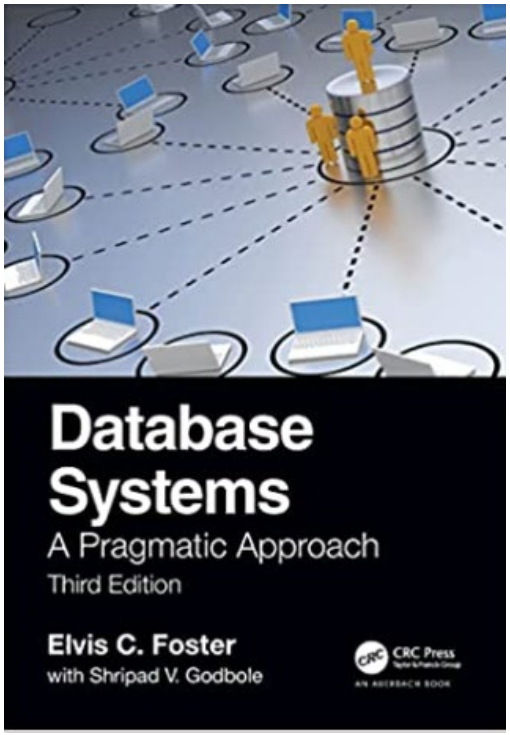
ON-LINE DISTANCE COURSE ON DATABASES

- **Modül 1. Veritabanlarının Teorik Temelleri**
- **Konu 1. Temel kavramlar**
- Ders 2. Veritabanı yönetim sistemine (DBMS) giriş

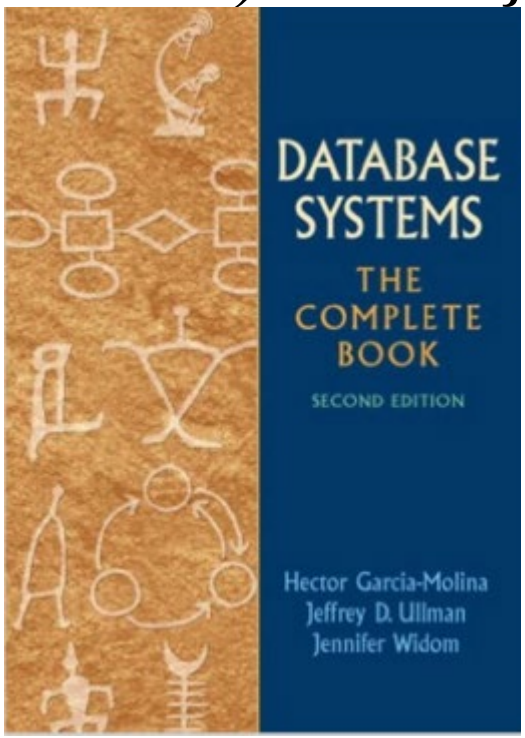
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANI YÖNETİM SİSTEMİNE (DBMS) GİRİŞ



Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.



Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

Azalov, P. (1991) Database, Sofia.

VERİTABANI YÖNETİM SİSTEMİ (DBMS)

- Herhangi bir veritabanı yönetim sistemi, amacı bir veritabanında düzenlenen verileri oluşturmak ve yönetmek olan bir yazılım sistemidir.
- Buradaki yönetim, verileri aramak, işlemek ve güncel tutmak anlamına gelir.
- Veritabanı yönetim sisteminin son işlevi, veritabanında yeni veri ekleme, mevcut verileri değiştirme ve kaldırma işlemlerinin varlığını gerektirir.
- Bir Veritabanı yönetim sisteminin düzgün çalışması, önceki paragraflarda açıklanan Veritabanı gereksinimlerini karşılamasını gerektirir:

VERİTABANI YÖNETİM SİSTEMİ (DBMS)

- veri açıklamasının işlenmesinden ayrılması;
- mantıksal ve fiziksel bağımsızlık;
- DB'de minimum veri yedekliliği;
- kullanışlı ve etkileyici güçlü kullanıcı arayüzüne sahip;
- Taleplerin verimli bir şekilde işlenmesi
- Veritabanındaki verilerin bütünlüğünün sağlanması, örneğin veritabanında mantıksal tutarlılığın ve bireysel kullanıcıların verilerinin uygun gizliliğinin sağlanması.

VERİTABANI YÖNETİM SİSTEMİ (DBMS)

- Kullanıcı açısından bakıldığında, bir Veritabanı Yönetim Sisteminin aşağıdaki özelliklere sahip olması gerekir:
- kalıcı bellek - veriler süreçlerden bağımsız olarak saklanmalıdır; veritabanları çok büyük miktarda veriye etkili erişime izin vermelidir; verilere erişim süresi, verinin hacminden bağımsız olmalıdır;
- program arayüzü - Veritabanı Yönetim Sistemi, kullanıcı veya verileri kullanan uygulama programı için güçlü bir sorgulama dili sağlar;

VERİTABANI YÖNETİM SİSTEMİ (DBMS)

- Kullanıcı açısından bakıldığında, bir Veritabanı Yönetim Sistemi'nin aşağıdaki özelliklere sahip olması gerekir:
- **işlem yönetimi** - Veritabanı Yönetim Sistemi verilere eşzamanlı erişimi destekler; işlemler adı verilen birden fazla süreç aracılığıyla uygulanır; verilerle çalışan her kullanıcının böyle bir süreci başlatması gerekir; DBMS, işlem izolasyonunu korur - her işlem atomiktir, yani. e. ya tamamen tamamlanır ya da tamamen reddedilir; bir işlemin yürütülmesi sırasında bir sorun ortaya çıkarsa, veritabanında neden olduğu değişikliklerle birlikte iptal edilir;

VERİTABANI YÖNETİM SİSTEMİ (DBMS)

- Kullanıcı açısından bakıldığında, bir Veritabanı Yönetim Sistemi 'nin aşağıdaki özelliklere sahip olması gerekir:
- **veri kararlılığı** - arıza veya hata durumunda veriler kurtarılabilir olmalıdır; kaba, bu ek okuma ve yazma işlemlerini içerir.
- Mimari olarak bir Veritabanı Yönetim Sistemi nin aşağıdaki işlevleri sağlaması gerekir:

VERİTABANI YÖNETİM SİSTEMİ (DBMS)

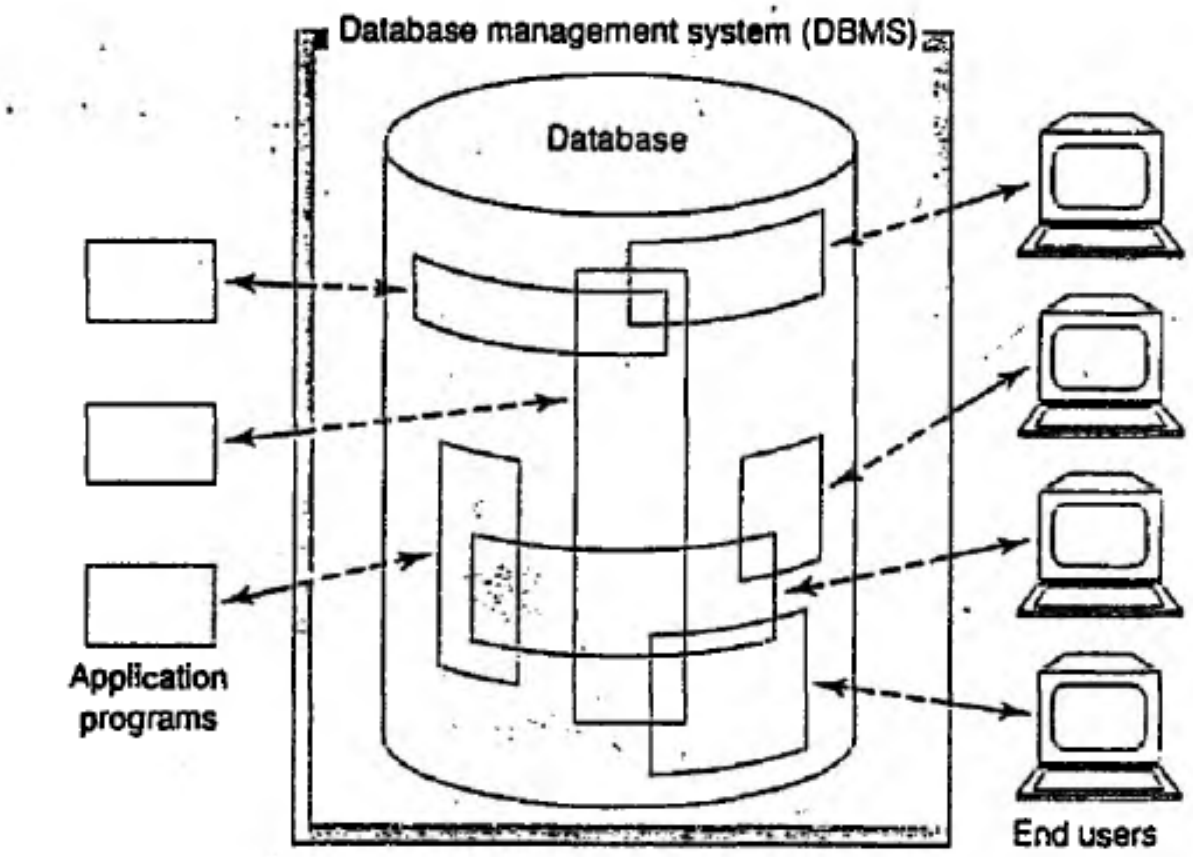
- Mimari olarak bir Veritabanı Yönetim Sistemi'nin aşağıdaki işlevleri sağlaması gerekir:
- Veritabanı Yönetim Sistemi, kullanıcının yeni veritabanları oluşturmasına olanak tanır; bu amaçla kullanıcı veritabanının şemasını belirler; **şema** mantıksal bir veri yapısıdır ve **özel bir veri tanımlama dili (DLL)** kullanılarak tanımlanır; bu dil sembolik veya grafiktir;

VERİTABANI YÖNETİM SİSTEMİ (DBMS)

- Mimari olarak bir Veritabanı Yönetim Sistemi 'nin aşağıdaki işlevleri sağlaması gerekir:
- Veritabanı Yönetim Sistemi, verilerde arama yapma ve verileri değiştirme yeteneğini sağlamalıdır; bu, bir veri işleme dili (DML) kullanılarak elde edilir; sorgulama dili, veri ararken kullanılan veri işleme dilinin bir alt dilidir;

VERİTABANI YÖNETİM SİSTEMİ (DBMS)

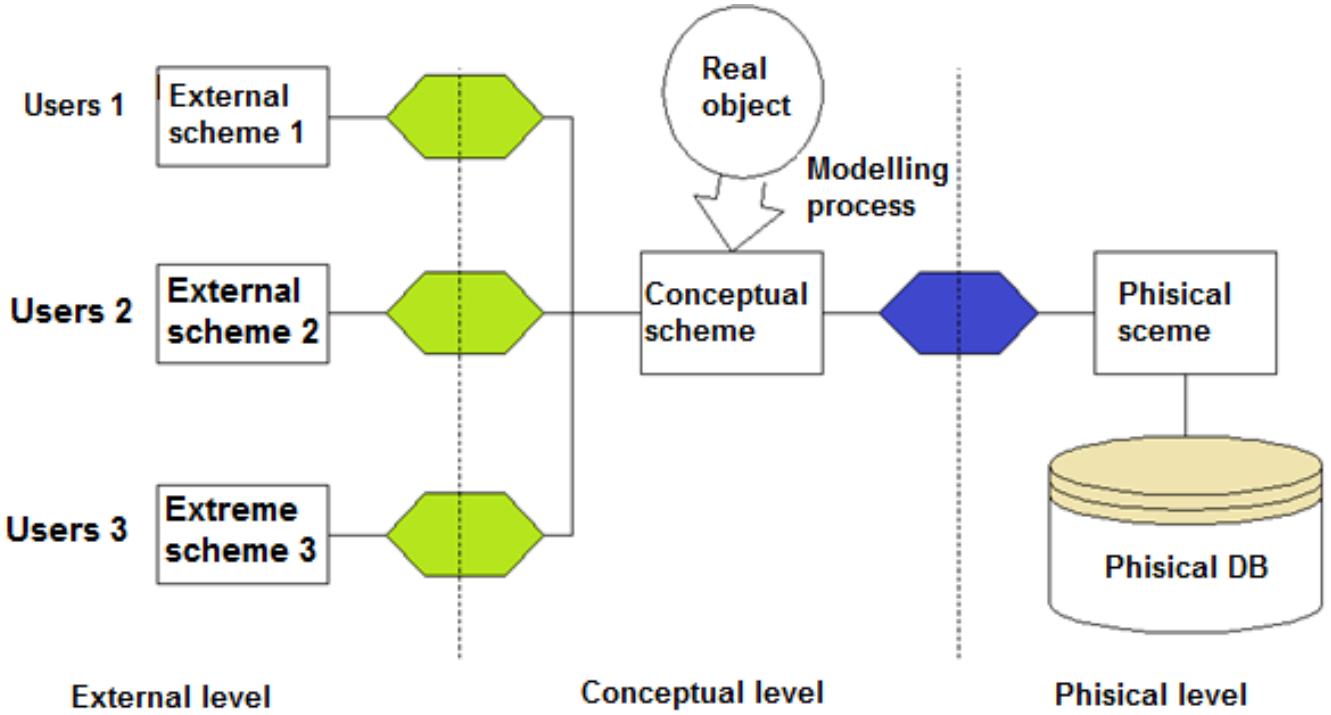
- Mimari olarak bir Veritabanı Yönetim Sistemi 'nin aşağıdaki işlevleri sağlaması gerekir:
- Veritabanı Yönetim Sistemi, çok büyük hacimli verileri güvenli ve uzun vadeli olarak depolamalıdır; depolama, verilerin verimli bir şekilde aranmasını ve değiştirilmesini sağlamalıdır;
- Veritabanı Yönetim Sistemi, birden fazla kullanıcının verilere eşzamanlı erişimini yönetmeli; birbirleriyle etkileşime girmemeli ve verilerin bütünlüğünü bozmamalıdır.

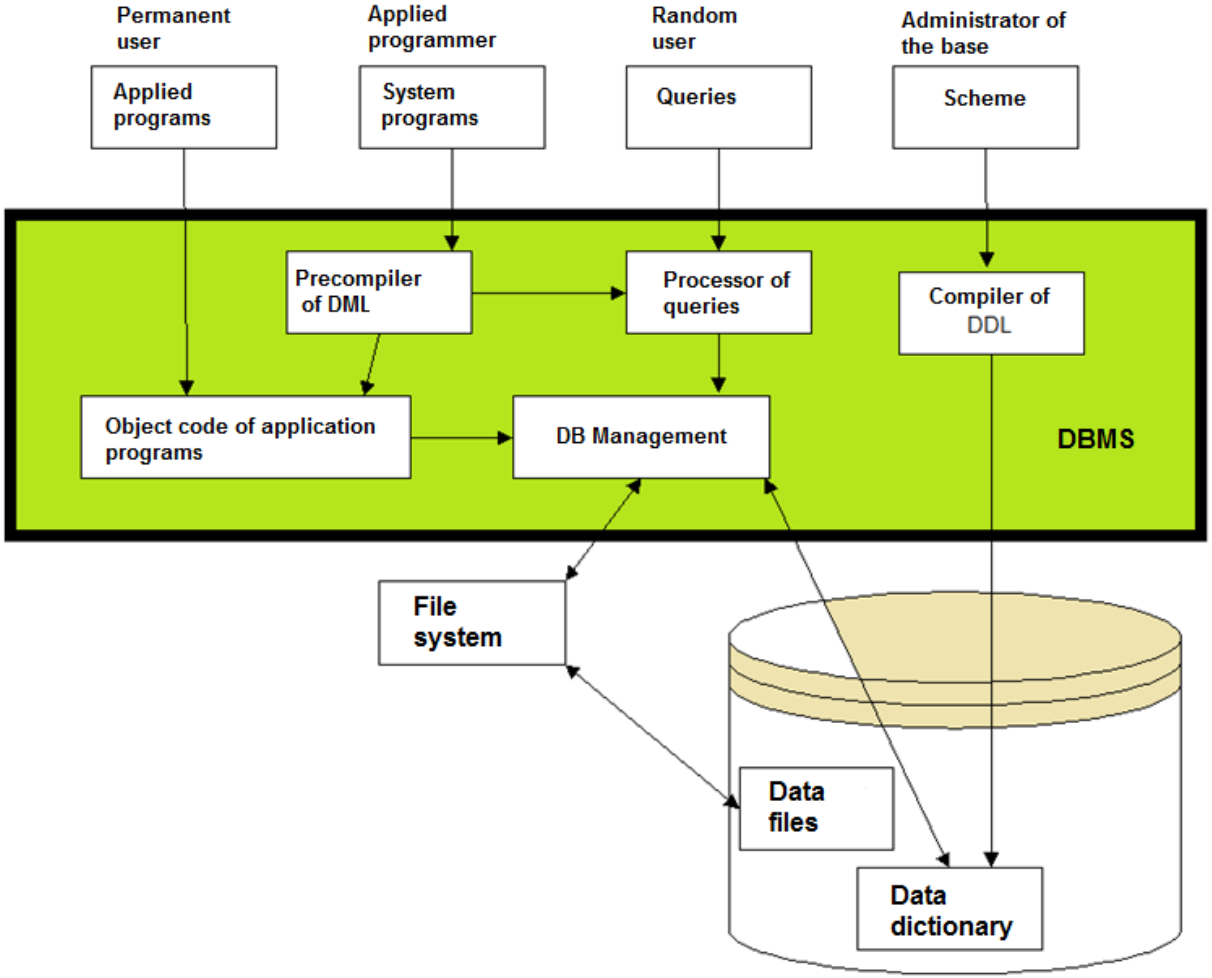


Bir veritabanı sisteminin basitleştirilmiş resmi

Veritabanının performans seviyeleri

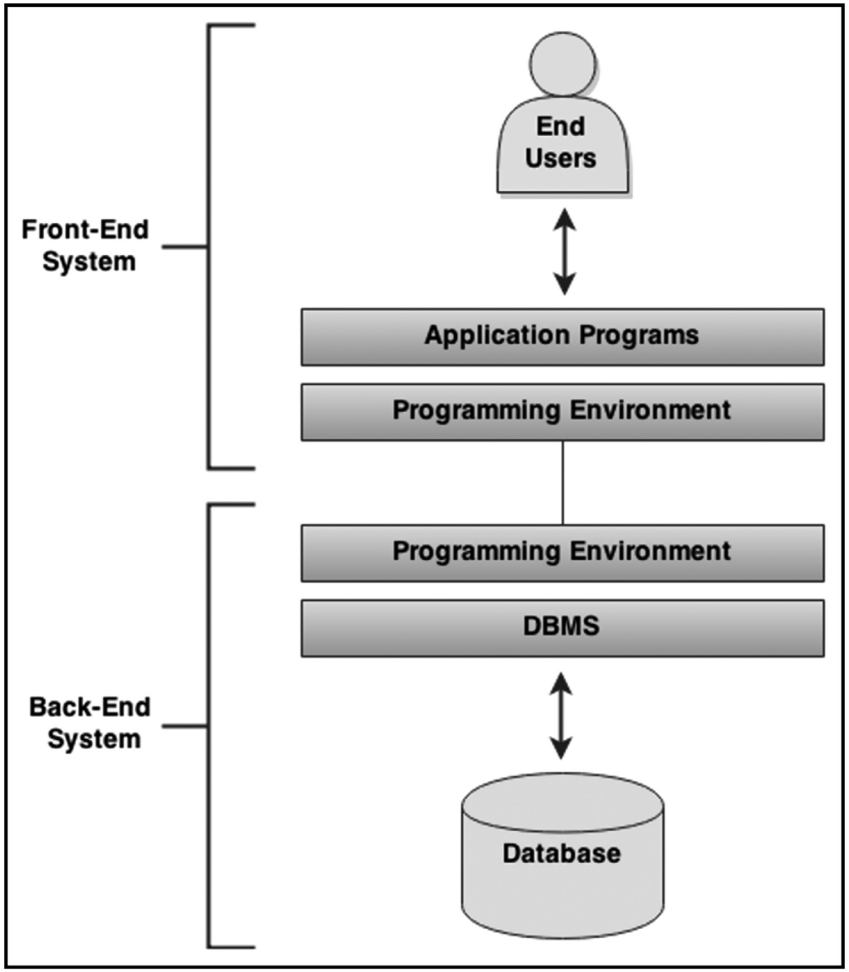
Dahili (fiziksel) seviye
Kavramsal düzey





DBMS'nin işlevsel şeması örneği

Co-funded by the Erasmus+ Programme of the European Union



Ön uç ve arka uç perspektifleri.

VERİTABANI YÖNETİM SİSTEMİ (DBMS)

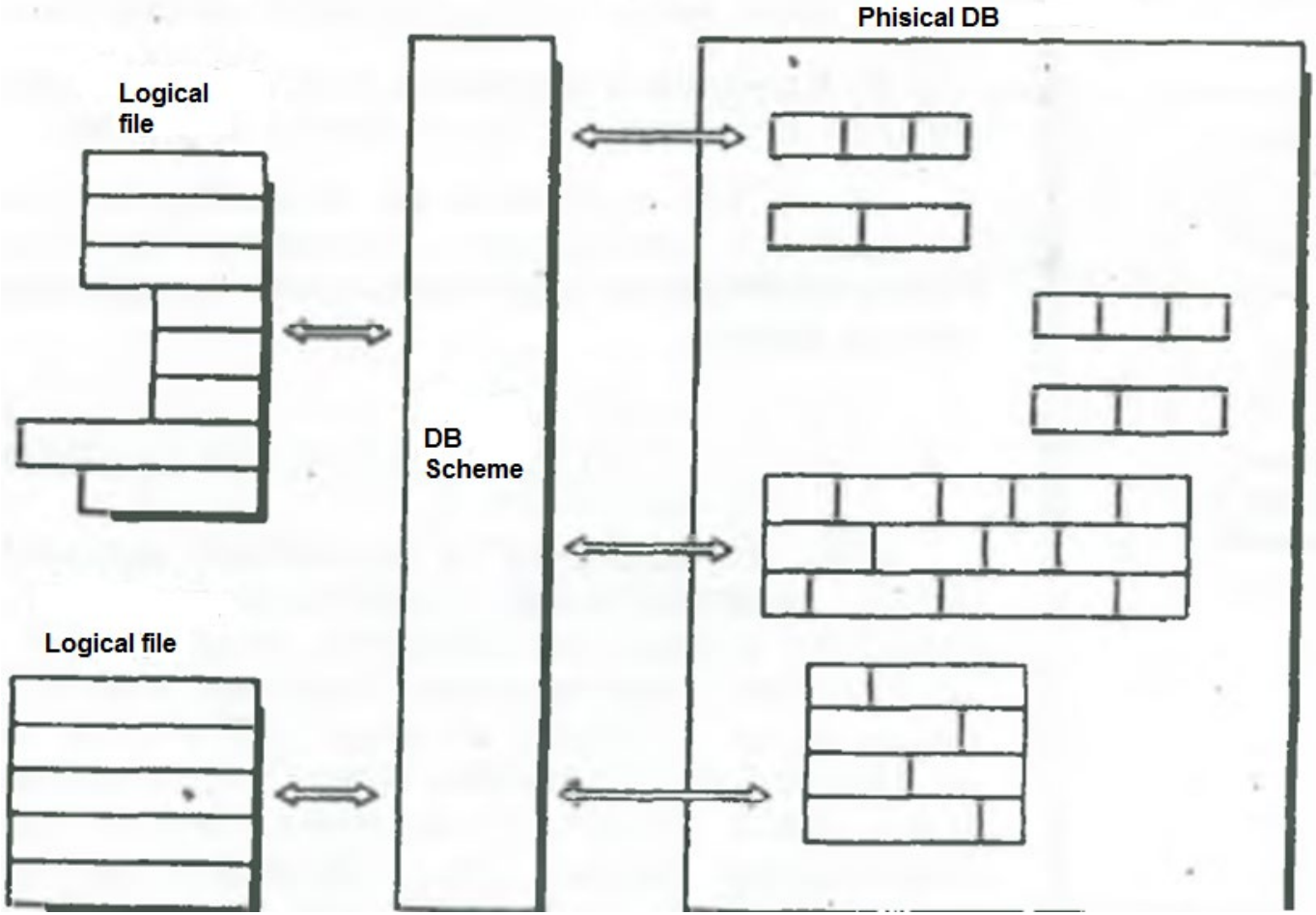
- Dolayısıyla, veritabanının bireysel kullanıcıların mantıksal dosyalarından oluşan bir koleksiyon olduğunu ve veritabanının harici bellekte saklanan bir veri dosyaları koleksiyonu olduğunu varsayabiliriz; Mantıksal ve fiziksel bir veritabanımız var.
- Bu durumda ilginç olan, hem mantıksal hem de fiziksel veri tabanlarının biri diğerini etkilemeden değişebilmesidir.
- İki veritabanının davranışında bu bağımsızlığı mümkün kılmak için, orta düzeyde bir veri gösterimi tanıtılmıştır. Kavramsal model veya kavramsal şema adıyla bilinir ve çoğunlukla yalnızca veritabanının bir şemasıdır.

VERİTABANI YÖNETİM SİSTEMİ (DBMS)

- Şema, veritabanının genel bir mantıksal açıklamasıdır.
- Nesnelerin adlarını, bazı özelliklerinin açıklamalarını ve aralarındaki ilişkileri içerir.
- Veritabanı yöneticisi yaygın adıyla anılan uzmanlar tarafından oluşturulur ve güncel tutulur.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

VERİTABANI YÖNETİM SİSTEMİ (DBMS)



VERİTABANI YÖNETİM SİSTEMİ (DBMS)

- Bir veritabanı şemasının varlığında, mantıksal veritabanında (mantıksal dosyalarda) yapılan değişiklikler veritabanı şemasına yansıtılır, ancak verilerin fiziksel veritabanında veya diğer uygulama programlarında depolanma şeklini etkilemez.

- Benzer şekilde, verilerin fiziksel veritabanında saklanma ve erişilme şeklindeki değişiklikler, mantıksal dosyalarda değişikliklere neden olmayacaktır (uygulama programlarında).

VERİTABANI YÖNETİM SİSTEMİ (DBMS) - Sonuç

- Veritabanı yönetim sistemi (DBMS), oluşturulmasını ve yönetimini kolaylaştıran yazılımdır.
- veritabanının. Bir kullanıcı bazı DSL (tipik olarak SQL) aracılığıyla bir istek gönderdiğinde, bu tür bir isteği yorumlayan, uygun talimatları yürüten ve isteğe yanıt veren DBMS'dir.
- İlk isteğin niteliğine bağlı olarak, yanıt (DBMS tarafından) doğrudan son kullanıcıya veya dolaylı olarak yürütülen bir uygulama programı aracılığıyla son kullanıcıya iletilebilir.

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANI YÖNETİM SİSTEMİ (DBMS) - Sonuç

- DBMS aracılığıyla Ders 1'de belirtilen DBS hedeflerine ulaşılır. Bu çok önemli yazılım sisteminin temel işlevleri şunlardır:
- Veri tanımı (ilişki, bağımlılıklar, bütünlük kısıtlamaları, görünümler vb.)
- Veri manipülasyonu (verilerin eklenmesi, güncellenmesi, silinmesi, alınması, yeniden düzenlenmesi ve toplanması)
- Veri güvenliği ve bütünlüğü kontrolleri
- Veri erişiminin yönetimi (sorgu optimizasyonu dahil), arşivleme ve eşzamanlılık
- Kullanıcı tarafından erişilebilen bir sistem kataloğunun bakımı (veri sözlüğü)

Çeşitli veritabanı dışı işlevlerin desteklenmesi (örneğin, kopyalama gibi yardımcı programlar)

Programlama dili desteği

İşlem yönetimi (ya tüm değişiklikler yapılır ya da hiç yapılmaz)

Yedekleme ve kurtarma hizmetleri

İletişim desteği (DBMS'nin temel iletişim yazılımıyla entegre olmasına izin verin)

Açık veritabanı bağlantısı (ODBC), Java veritabanı bağlantısı (JDBC) ve diğer ilgili sorunlar dahil birlikte çalışabilirlik desteği

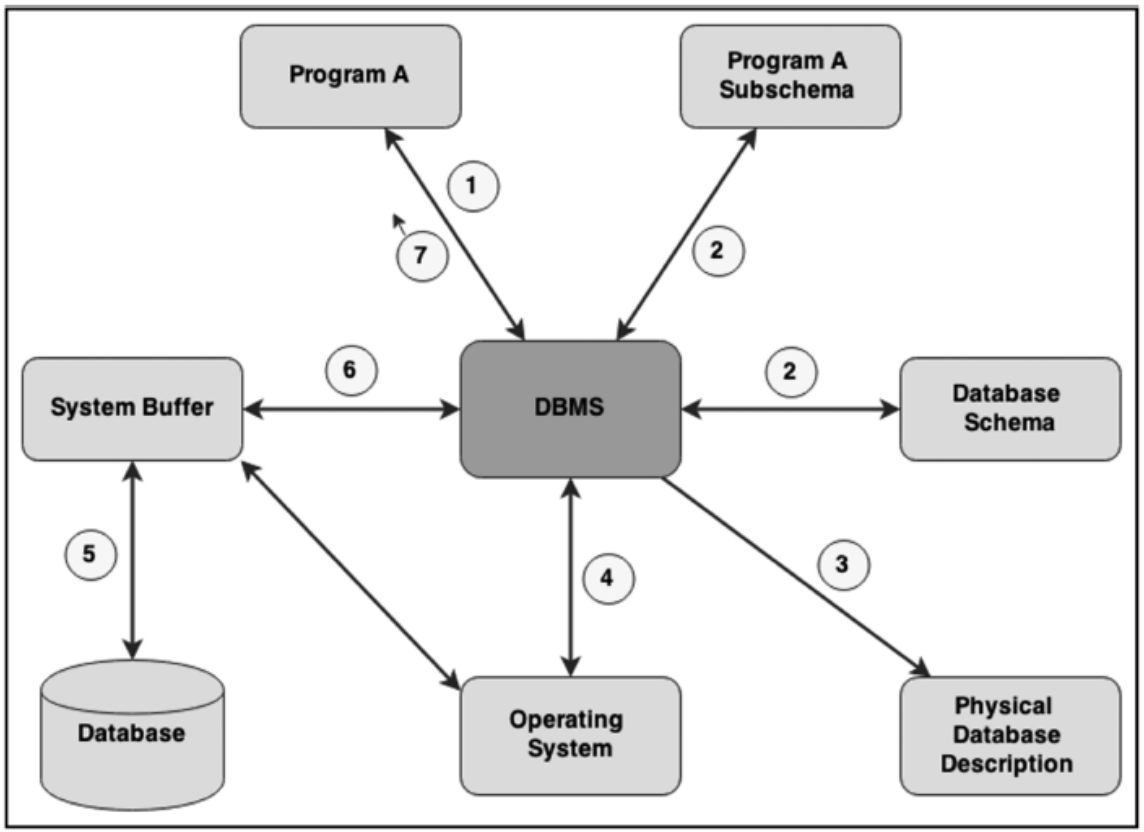
VERİTABANI YÖNETİM SİSTEMİ (DBMS) - Sonuç

Optimum verimlilik ve performans, iyi bir DBMS'nin ayırt edici özellikleridir. DBMS'nin kritik rolünü göstermek için, bir uygulama programı veritabanına eriştiğinde gereken adımları göz önünde bulundurun:

- Program-A, DBMS'ye (alt şema dili cinsinden ifade edilen) bir istek gönderir.
- DBMS, Program-A alt şemasına, şemasına ve fiziksel açıklamasına bakar (bu bilgiler tablolarda saklanır).
- DBMS, hangi dosyalara erişilmesi gerektiğini, dosyalardaki hangi kayıtların gerekli olduğunu belirleyerek verilere erişmenin en uygun yolunu belirler ve DBMS'ye erişmenin en iyi yöntemi, işletim sistemine talimat (lar) (okuma veya yazma) verir.
- İşletim sistemi, disk depolama alanı ile ana bellek arasında veri aktarımına neden olur.
- DBMS sorunları gerekli alanları aktarmak için taşınır.
- DBMS, kontrolü Program-A'ya döndürür (muhtemelen bir tamamlama koduyla).
- Veritabanı Sistemleri.

VERİTABANI YÖNETİM SİSTEMİ (DBMS) - Sonuç

- Aşağıdaki şekil grafiksel bir gösterim sunmaktadır ancak bu adımların kullanıcıya şeffaf olacak şekilde otomatik olarak gerçekleştirildiğini unutmayın:



Steps involved when application programs access a database.

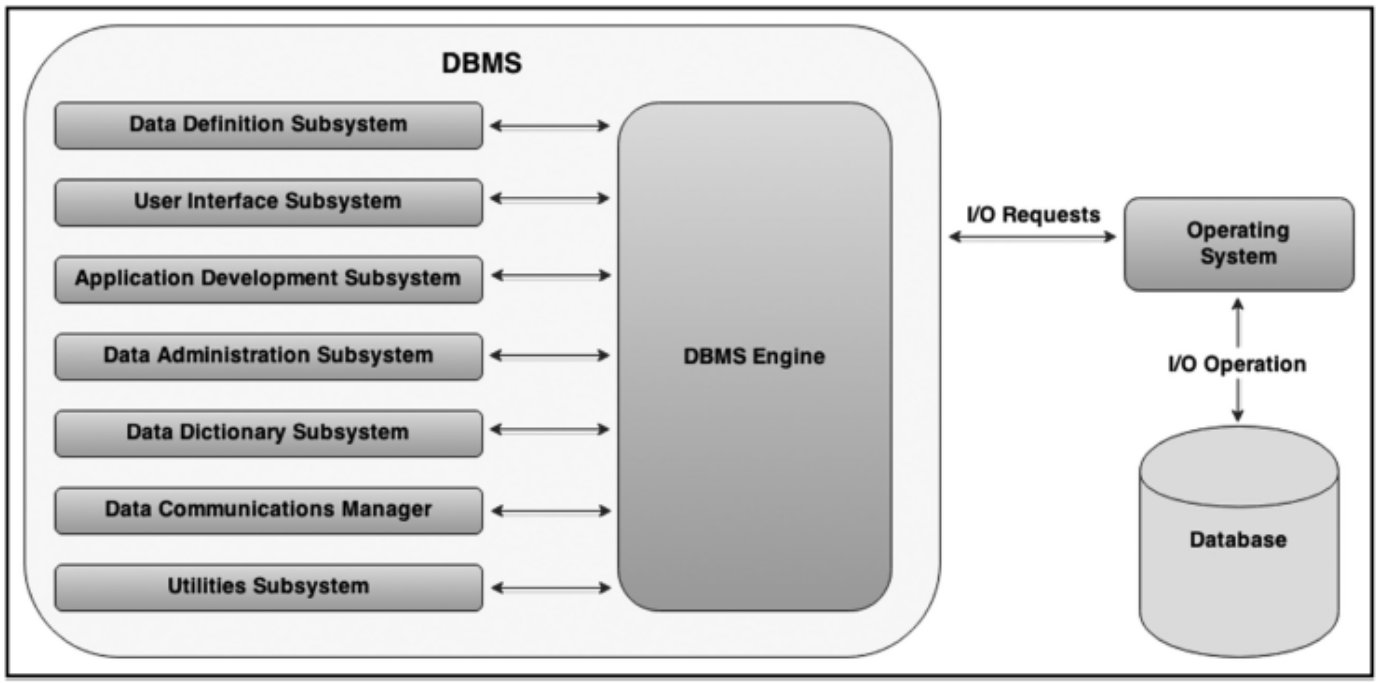
VERİTABANI YÖNETİM SİSTEMİ (DBMS) - Sonuç

DBMS aslında bir dizi ortak amaç için birlikte çalışan yazılım bileşenlerinin karmaşık bir birleşimidir. Açıklama amacıyla DBMS'nin temel bileşenlerini aşağıdaki gibi temsil edebiliriz:

- DBMS Motoru
- Veri Tanımlama Alt Sistemi
- Kullanıcı Arayüzü Alt Sistemi
- Uygulama Geliştirme Alt Sistemi
- Veri Yönetimi Alt Sistemi
- Veri Sözlüğü Alt Sistemi
- Veri İletişim Yöneticisi
- Yardımcı Programlar Alt Sistemi

VERİTABANI YÖNETİM SİSTEMİ (DBMS) - Sonuç

- Bu işlevsel bileşenler mutlaka somut olarak tanımlanabilir olmayabilir ancak DBMS'nin kabul edilebilir performansını sağlamak için mevcutturlar:



Bir DBMS'nin fonksiyonel bileşenleri.

VERİTABANI YÖNETİM SİSTEMİ (DBMS) - Sonuç

- DBMS motoru, işletim sistemi aracılığıyla diğer tüm alt sistemler ile fiziksel cihaz (bilgisayar) arasındaki bağlantıdır. Bazı önemli işlevler aşağıdaki gibidir:

- İşletim sistemi yardımcı programlarına ve programlarına doğrudan erişimin sağlanması (örneğin, giriş/çıkış talepleri, veri sıkıştırma talepleri, iletişim talepleri vb.)

- İşletim sistemi aracılığıyla dosya erişiminin (ve veri yönetiminin) yönetimi

- Kullanıcı isteklerini gerçekleştirmek için bellek ile sistem arabellek(ler)i arasındaki veri aktarımının yönetimi

- Veri sözlüğünde (sistem kataloğu) saklanan genel verilerin ve meta verilerin bakımı

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Veritabanı Yönetim Sistemi (DBMS) nedir ve faydası nedir?

2. Veritabanı Yönetim Sistemi (DBMS)'nin birkaç avantajını açıklayın.

3. Dosya sistemi ile Veritabanı Yönetim Sistemi (DBMS) arasındaki fark nedir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

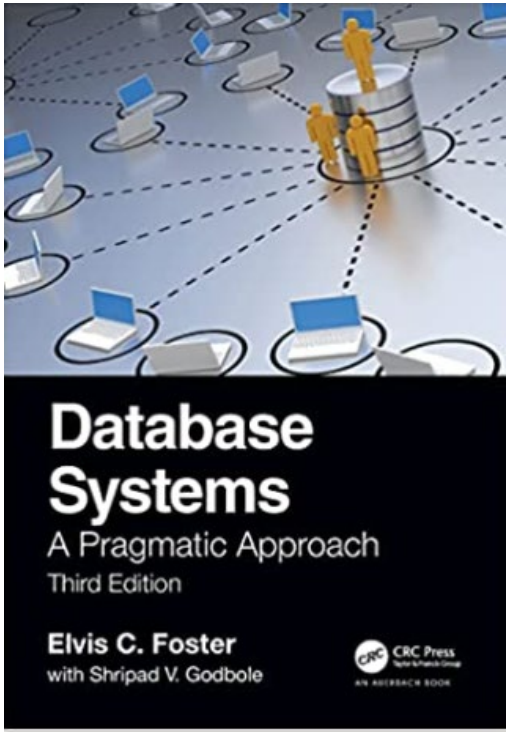
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

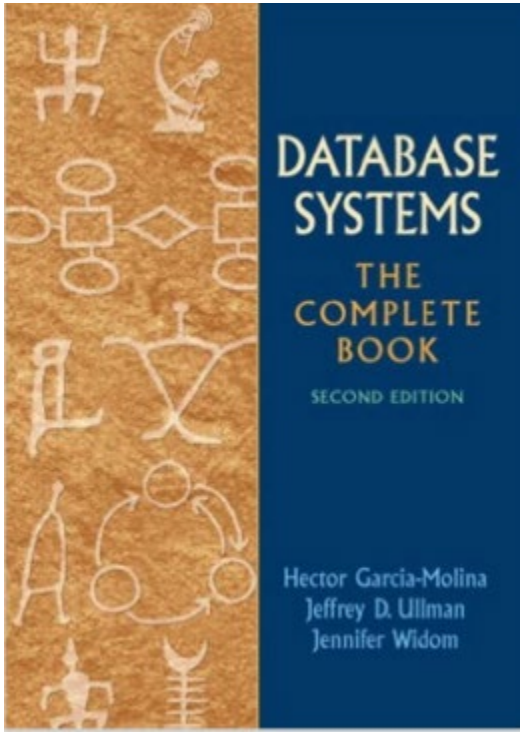
- **Modül 1. Veritabanlarının Teorik Temelleri**
- **Konu 2. Veritabanlarının mantıksal ve fiziksel temelleri**
- Ders 1. Veritabanlarının mantıksal temelleri

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANLARININ MANTIK TEMELLERİ





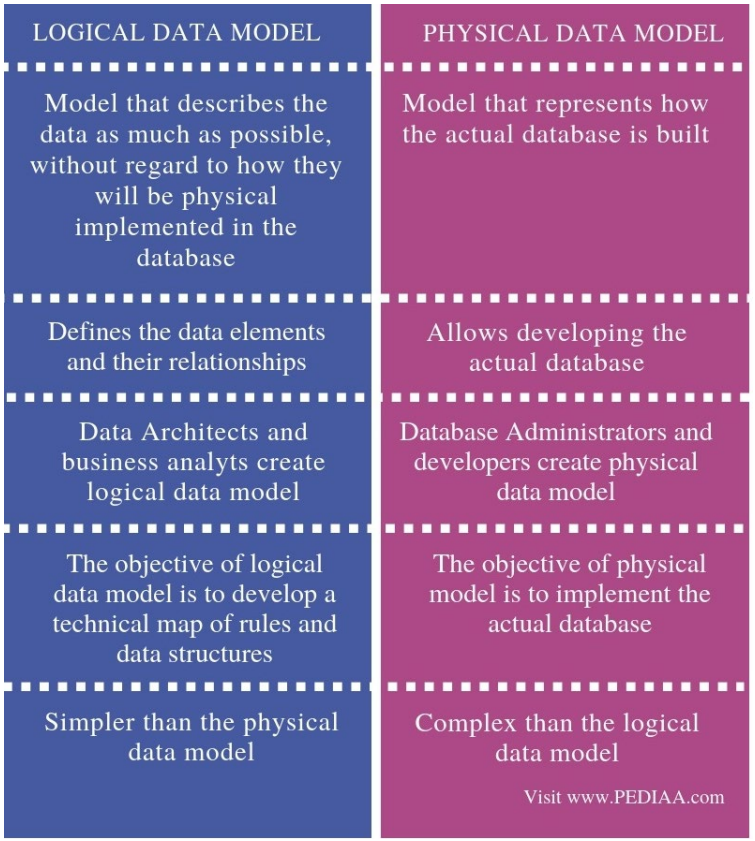
Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.

Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

Azalov, P. (1991) Database, Sofia.

GİRİŞ

- **Veritabanı tasarım sürecinin İKİ ANA aşaması vardır:**
  - **Mantıksal veritabanı tasarımı;**
  - **Fiziksel veritabanı tasarımı.**



Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

Veri tabanı tasarımı

- **Mantıksal veritabanı tasarımı**
- Bir kuruluşta kullanılan, belirli bir veri modeline (örneğin **ilişkisel**) dayalı, ancak belirli bir **Veritabanı Yönetim Sistemi (DMBS)**'nden ve diğer **fiziksel** hususlardan **bağımsız** olarak kullanılan bir bilgi modeli oluşturma süreci.

- **Fiziksel tasarım**
- Veritabanının **ikincil depolamaya** uygulanmasına ilişkin bir **açıklama üretme süreci**

verilere etkin erişim sağlamak için kullanılan **temel ilişkileri**, **dosya** organizasyonlarını ve dizin tasarımını ve ilgili **bütünlük kısıtlamalarını** ve **güvenlik** önlemlerini açıklar.

Purpose of Database Design

- Structure the data in stable **structures**, called **normalized tables**
  - **Not** likely to **change over time**
  - **Minimal** redundancy

- Develop a **logical database design** that reflects **actual data requirements**

  - It forms the **base** for a **physical** database design

Veritabanı Tasarımının Amacı

- İlişkisel bir veritabanı modelini, çeşitli performans faktörlerini dengeleyen bir teknik dosyaya ve veritabanı tasarımına çevirmek

- Veritabanı etkinliklerini verimli, doğru ve güvenli bir şekilde işleyecek veri depolama teknolojilerini seçmek

Veritabanı Tasarımı Süreci

- Mantıksal Tasarım
- **Kavramsal veri modeline dayalı**

4 önemli adım:

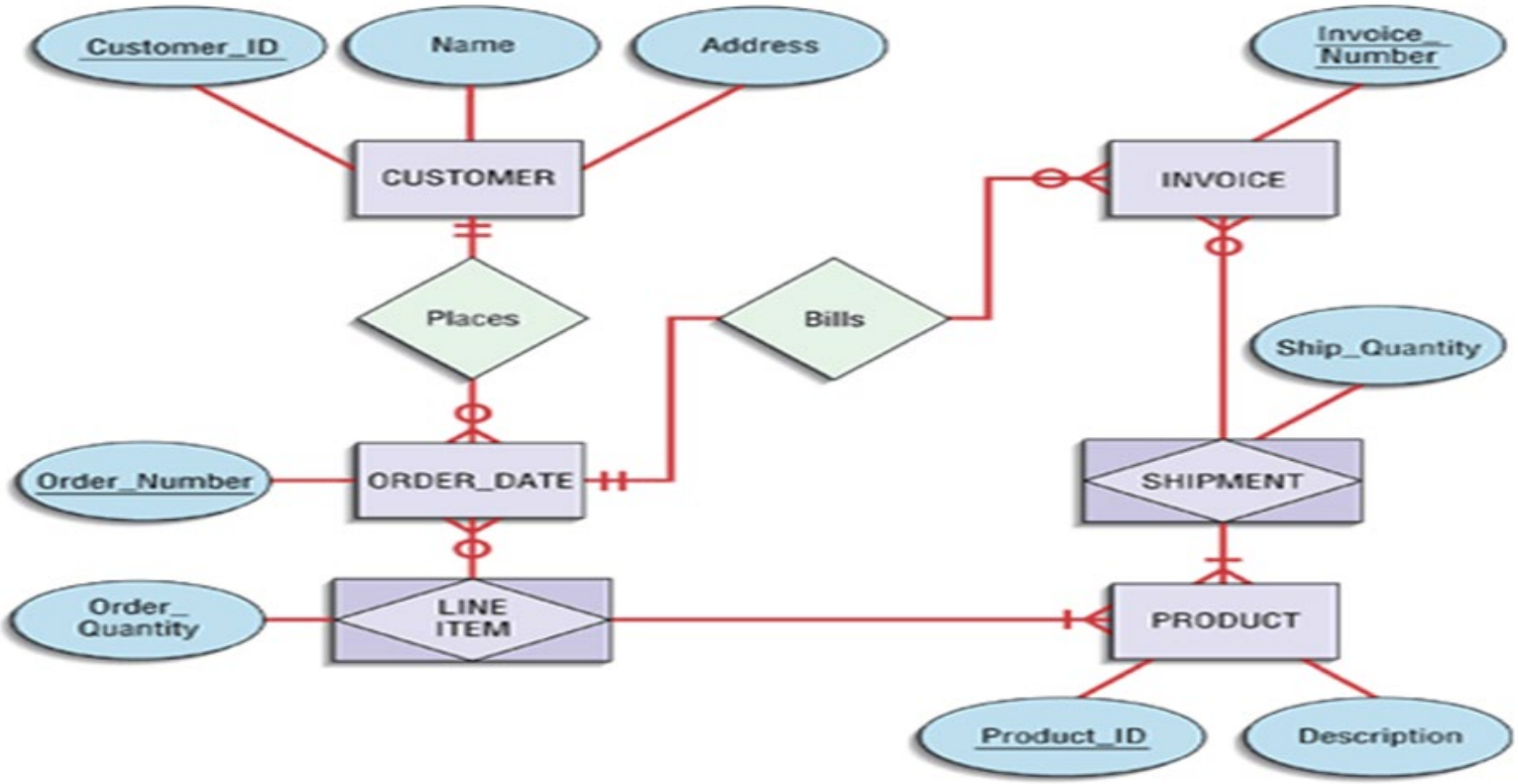
1. Normalleştirme ilkelerini kullanarak uygulama için bilinen her kullanıcı arayüzü için mantıksal bir veri modeli geliştirin.

2. Tüm kullanıcı arayüzlerinden gelen normalleştirilmiş veri gereksinimlerini tek bir birleştirilmiş mantıksal veritabanı modelinde birleştirin.

3. Uygulamaya yönelik kavramsal E-R veri modelini normalleştirilmiş veri gereksinimlerine çevirin.

4. Birleştirilmiş mantıksal veritabanı tasarımını çevrilmiş E-R modeliyle karşılaştırın ve uygulama için son bir mantıksal veritabanı modeli üretin.

Simple Example of Logical Data Modeling — Conceptual Data Model and Transformed Relations



Relations:

CUSTOMER(Customer_ID,Name,Address)
PRODUCT(Product_ID,Description)
ORDER(Order_Number,Customer_ID,Order_Date)
LINE ITEM(Order_Number,Product_ID,Order_Quantity)
INVOICE(Invoice_Number,Order_Number)
SHIPMENT(Invoice_Number,Product_ID,Ship_Quantity)

Veritabanı Tasarımı Süreci

Fiziksel tasarım

- Mantıksal veritabanı tasarımının sonuçlarına dayanarak

Önemli kararlar

1. Mantıksal veritabanı modelinden her öznitelik için depolama biçimini seçme
2. Mantıksal veritabanı modelindeki nitelikleri fiziksel kayıtlar halinde gruplama
4. İkincil bellekteki (sabit diskler ve manyetik bantlar) ilgili kayıtların, kayıtların hızlı bir şekilde saklanması, geri getirilmesi ve güncellenmesi için düzenlenmesi
5. Erişimi daha verimli hale getirmek amacıyla veri depolamak için ortam ve yapıların seçilmesi

Teslimatlar ve Sonuçlar

- **Mantıksal veritabanı tasarımı, sistem girişi veya çıkışındaki her veri öğesini hesaba katmalıdır**
- **Normalleştirilmiş ilişkiler birincil çıktıdır**
- **Fiziksel veritabanı tasarımı, ilişkilerin dosyalara dönüştürülmesini sağlar.**

İlişkisel Veritabanı Modeli

- **İlgili tablolar veya ilişkiler kümesi olarak temsil edilen veriler**

İlişki

- Adlandırılmış, iki boyutlu bir veri tablosu. Her ilişki bir dizi adlandırılmış sütundan ve isteğe bağlı sayıda adsız satırdan oluşur

Özellikler

- Hücrelerdeki girişler basittir
- Sütunlardaki girişler aynı değer grubundandır
- Her satır benzersizdir
- Sütunların sırası, ilişkinin anlamı veya kullanımı değiştirilmeden değiştirilebilir
- Satırlar herhangi bir sırayla değiştirilebilir veya saklanabilir

Form ve Rapor Tasarlama

Analiz aşamasının sonunda sistem **giriş ve çıkışları** üretilir.

- Bu aşamada **kesin görünüm tanımlanmadı**

- Formlar ve raporlar **DFD (Varlık İlişkisi Diyagramı) (Veri Akış Diyagramı)** ve **E-R diyagramlarıyla** bütünleşiktir.
  - *DFD ve E-R diyagramı, uygulamanızdaki bilgi akışlarının görsel bir temsilidir. Bilginin uygulamaya nasıl girip çıktığını, bilgiyi nelerin değiştirdiğini ve bilginin nerede saklandığını gösterir.*

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

Form ve Rapor Tasarlama: Temel Kavramlar

- **Biçim**
- Önceden tanımlanmış bazı verileri içeren ve ek verilerin doldurulacağı bazı alanları içerebilen **bir iş belgesi**
- Bir formun örneği genellikle **bir veritabanı kaydına** dayanır

Vs

- **Rapor**
- Yalnızca önceden tanımlanmış verileri içeren **bir iş belgesi**
- Verileri okumak veya görüntülemek için **pasif** bir belge
- Genellikle **birçok veritabanı kaydından veya işleminden** veri içerir

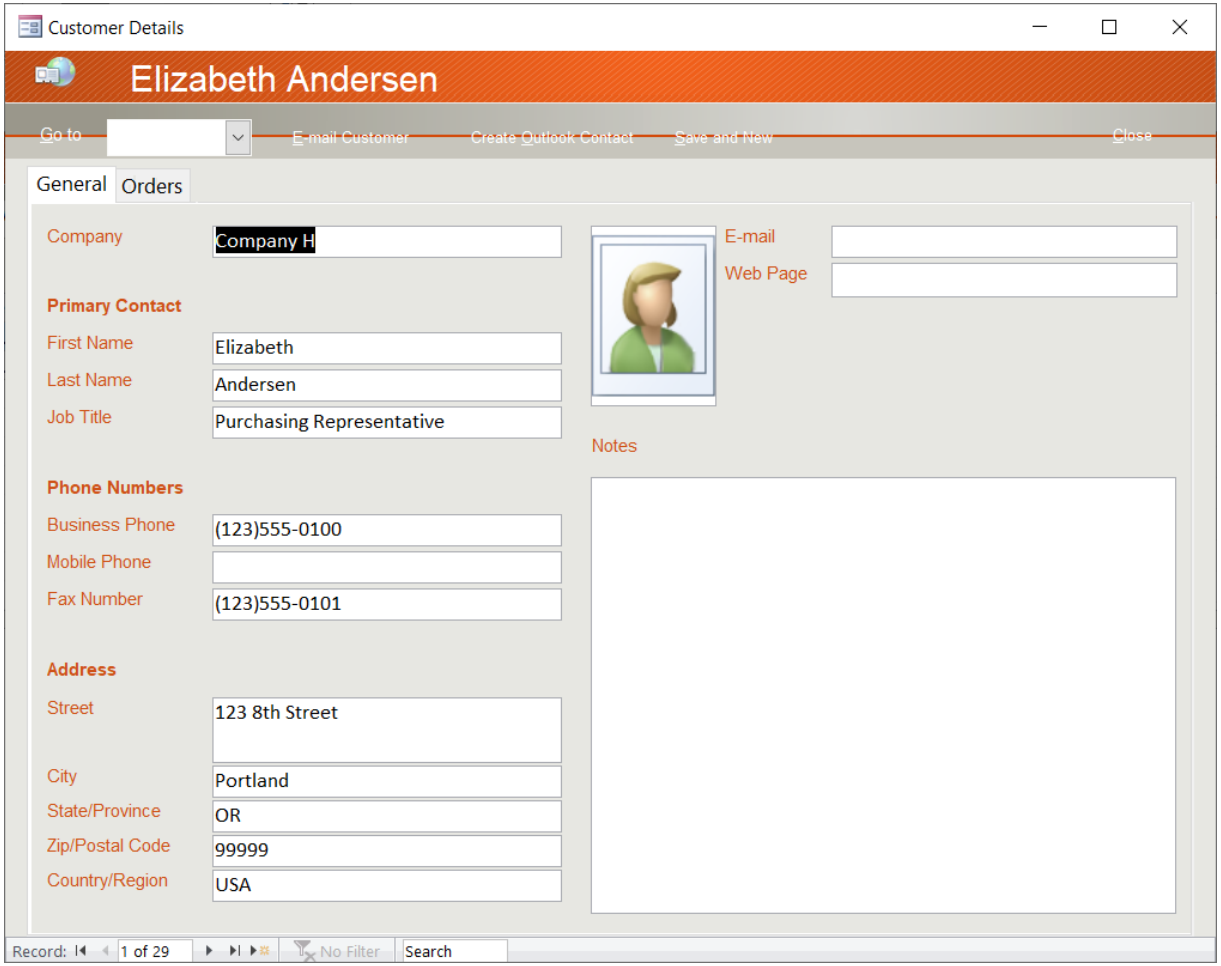
Co-funded by the Erasmus+ Programme of the European Union

Form ve Rapor Tasarlama Süreci

- **Kullanıcı odaklı** etkinlik
- **Prototip** oluşturma yaklaşımını takip eder.
- **Gereksinimlerin** belirlenmesi
- Formu veya raporu **kim kullanacak?**
- Formun veya raporun **amacı nedir?**
- **Rapora ne zaman** ihtiyaç duyulur veya kullanılır?
- Formun veya raporun **nereye teslim edilmesi** ve kullanılması gerekiyor?
- Formu veya raporu **kaç kişinin kullanması veya görüntülemesi** gerekiyor?

Form ve Rapor Tasarlama Süreci

- **Prototipleme**
- İlk **prototip** **gereksinimlere** göre tasarlandı.
- Kullanıcılar prototip tasarımını inceler ve tasarımı kabul eder veya değişiklik talebinde bulunur
- Değişiklik talep edilmesi durumunda yapım-değerlendirme-talep döngüsü tasarım kabul edilene kadar tekrarlanır.



Microsoft Access'te tasarlanmış bir veri giriş ekranı

Co-funded by the Erasmus+ Programme of the European Union

Teslimatlar ve Sonuçlar

- **Tasarım spesifikasyonları ana çıktılardır ve üç bölüm içerir**

1. Anlatı
2. Ekran Tasarımı
3. Test ve kullanılabilirlik değerlendirmesi

Formlar ve Raporlar için Genel Biçimlendirme Yönergeleri

- **Vurgulama**
- Kullanıcıyı belirli bilgilere çekmek veya belirli bilgilerden uzaklaştırmak için tedbirli/dikkatli kullanın
- Yanıp sönen ve duyulabilir tonlar yalnızca kullanıcının hemen ilgilenmesini gerektiren kritik bilgileri vurgulamak için kullanılmalıdır.
- Yöntemler, vurgulanan bilgilerin önem düzeyine göre tutarlı bir şekilde seçilmeli ve kullanılmalıdır.

Co-funded by the Erasmus+ Programme of the European Union

Methods of Highlighting

- Blinking and audible tones
- Color differences
- Intensity differences
- Size differences
- Font differences
- Reverse video
- Boxing
- Underlining
- All capital letters
- Offsetting the position of nonstandard information

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Formlar ve Raporlar için Genel Biçimlendirme Yönergeleri

Guidelines for Displaying Text

Case	Display text in mixed upper- and lowercase and use conventional punctuation.
Spacing	Use double spacing if space permits. If not, place a blank line between paragraphs.
Justification	Left-justify text and leave a ragged right margin.
Hyphenation	Do not hyphenate words between lines.
Abbreviations	Use abbreviations and acronyms only when they are widely understood by users and are significantly shorter than the full text.

Formlar ve Raporlar için Genel Biçimlendirme Yönergeleri

- Tablo ve listelerin görüntülenmesi
- **Etiketler**
- Tüm sütun ve satırların anlamlı etiketleri olmalıdır
- Etiketler vurgulama yöntemiyle diğer bilgilerden ayrılmalıdır.
- Veriler tek bir ekranın veya sayfanın ötesine geçtiğinde etiketleri yeniden görüntüleme

Formlar ve Raporlar için Genel Biçimlendirme Yönergeleri

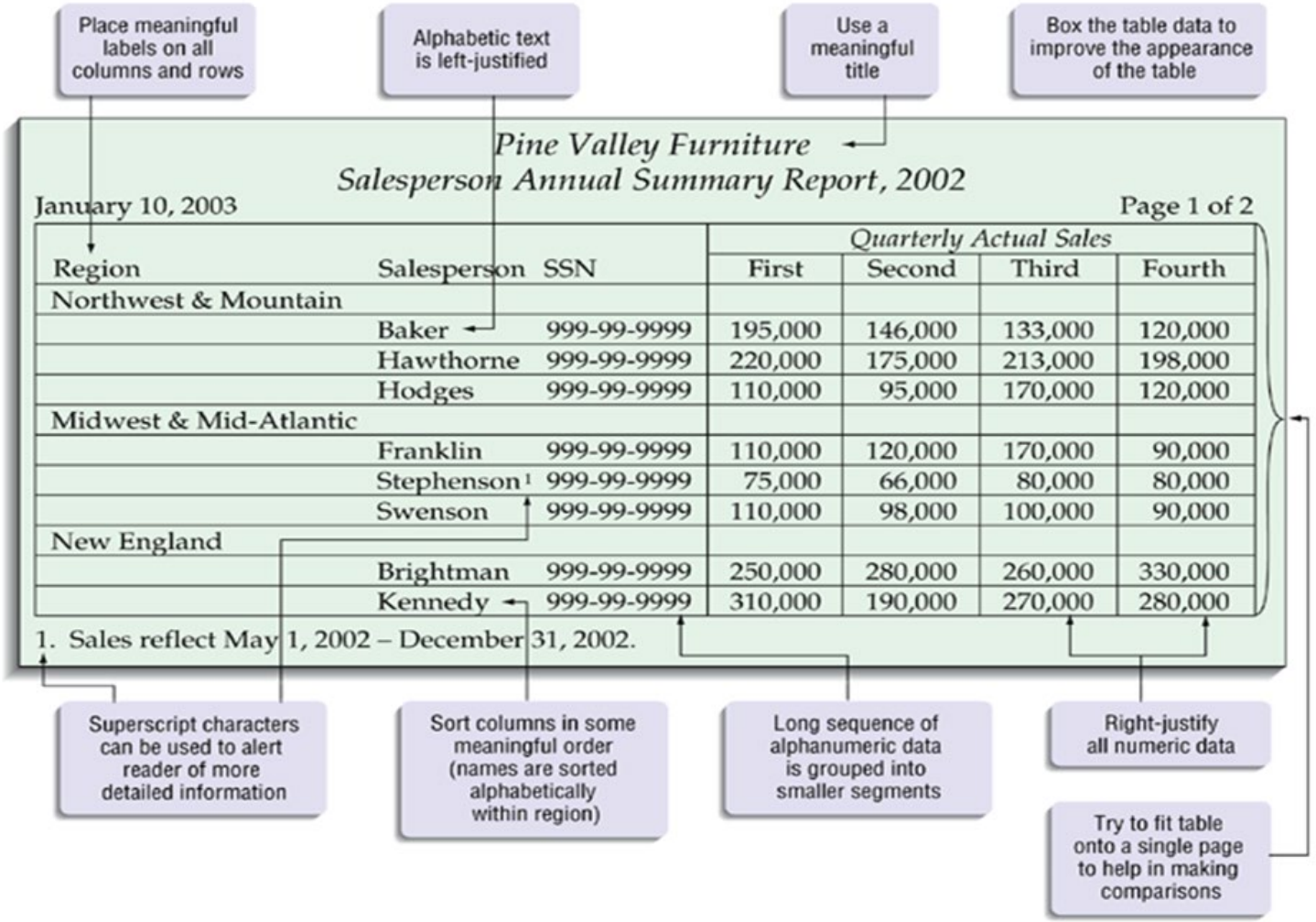
- Tabloların ve listelerin görüntülenmesi (devamı)
- Sütunları, satırları ve metni biçimlendirme
- Anlamlı bir sıraya göre sıralayın
- Uzun sütunlarda her 5 satırın arasına boş bir satır yerleştirin
- Birden çok sütunda görüntülenen benzer bilgiler dikey olarak sıralanmalıdır
- Sütunların aralarında en az iki boşluk olmalıdır
- Kullanıcının not yazması için yazdırılan raporlarda beyaz alana izin verin
- Vurgu dışında tek bir yazı tipi kullanın
- Ekranlar ve raporlar içinde ve genelinde aynı yazı tipi ailesini kullanın
- Aşırı süslü yazı tiplerinden kaçının

Co-funded by the Erasmus+ Programme of the European Union

Formlar ve Raporlar için Genel Biçimlendirme Yönergeleri

- Tabloların ve listelerin görüntülenmesi (devamı)
  - Sayısal, metinsel ve alfasayısal verileri biçimlendirme
- Sayısal verileri sağa yaslayın ve sütunları ondalık noktalara veya diğer sınırlayıcılara göre hizalayın
- Metin verilerini sola yasla. Kısa satır uzunluğu kullanın; genellikle satır başına 30 ila 40 karakter
- Uzun alfasayısal veri dizilerini her biri üç ila dört karakterden oluşan küçük gruplara bölün

Tabular Report Illustrating Good Report Design



Arayüzler ve Diyaloglar Tasarlamak

- **Bilgilerin kullanıcılara nasıl sağlandığı ve kullanıcılardan nasıl alındığına odaklanın**
- **Diyaloglar iki kişi arasındaki konuşmaya benzer**
- **İyi bir insan-bilgisayar arayüzü, bir sistemin farklı bileşenlerini bulmak, görüntülemek ve çağırmak için tek tip bir yapı sağlar.**

Arayüzler ve Diyaloglar Tasarlama Süreci

- **Kullanıcı odaklı etkinlik**
- Form ve rapor tasarlama sürecine benzer
- Prototipleme metodolojisini kullanır
- Bilgi toplamak
- Prototip oluştur
- Kullanılabilirliği değerlendirin
- Ayrıntılandırmalar yapın

Arayüzler ve Diyaloglar Tasarlama Süreci

- **Teslimat**
- Tasarım özellikleri
- Anlatı
- Örnek Tasarım
- Test ve kullanılabilirlik değerlendirmesi

Arayüz Tasarımı

- **Arayüz Tasarımı**
  1. Düzen tasarlamak,
  2. Yapılanma ve
  3. Veri giriş alanının kontrol edilmesi,
  4. Geri bildirim sağlamak,
  5. Çevrimiçi yardım tasarlama

Düzenleri Tasarlama

- **Düzenleri** Tasarlama
  - **Kağıt üzerindeki** form ve raporlara benzer **standart** formatlar kullanılmalıdır
  - Veri giriş ekranlarında **ekran gezinmesi** kağıt formlarda olduğu **gibi soldan sağa, yukarıdan aşağıya** doğru olmalıdır.

Düzenleri Tasarlama

- **Esneklik** ve **tutarlılık** birincil tasarım hedefleridir.
- Kullanıcılar alanlar arasında serbestçe hareket edebilmelidir (örneğin ileri ve geri)
- Kullanıcı açıkça talep edene kadar veriler kalıcı olarak kaydedilmemelidir
- Her tuş ve komut bir fonksiyona atanmalıdır

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

Veri Girişinin Yapılandırılması

Giriş	**Zaten çevrimiçi** olan veya hesaplanabilen verilere **(tarih, yaş, toplam satış…) asla** gerek duymazsınız.
Varsayılanlar	Uygun olduğunda her zaman varsayılan değerleri sağlayın **(ürün fiyatı)**
Birimler	Giriş için talep edilen veri birimlerinin türünü netleştirin
Başlıklama	Her zaman alanların yanına bir başlık yerleştirin **(dd/ mm/yyyy)**
Biçim	Biçimlendirme örnekleri sağlayın (ondalık basamak, virgül, $)
Gerekçelendirme	Veri girişlerini otomatik olarak gerekçelendirin
Yardım	Uygun olduğunda **bağlama duyarlı** yardım sağlayın

Veri Girişini Kontrol Etme

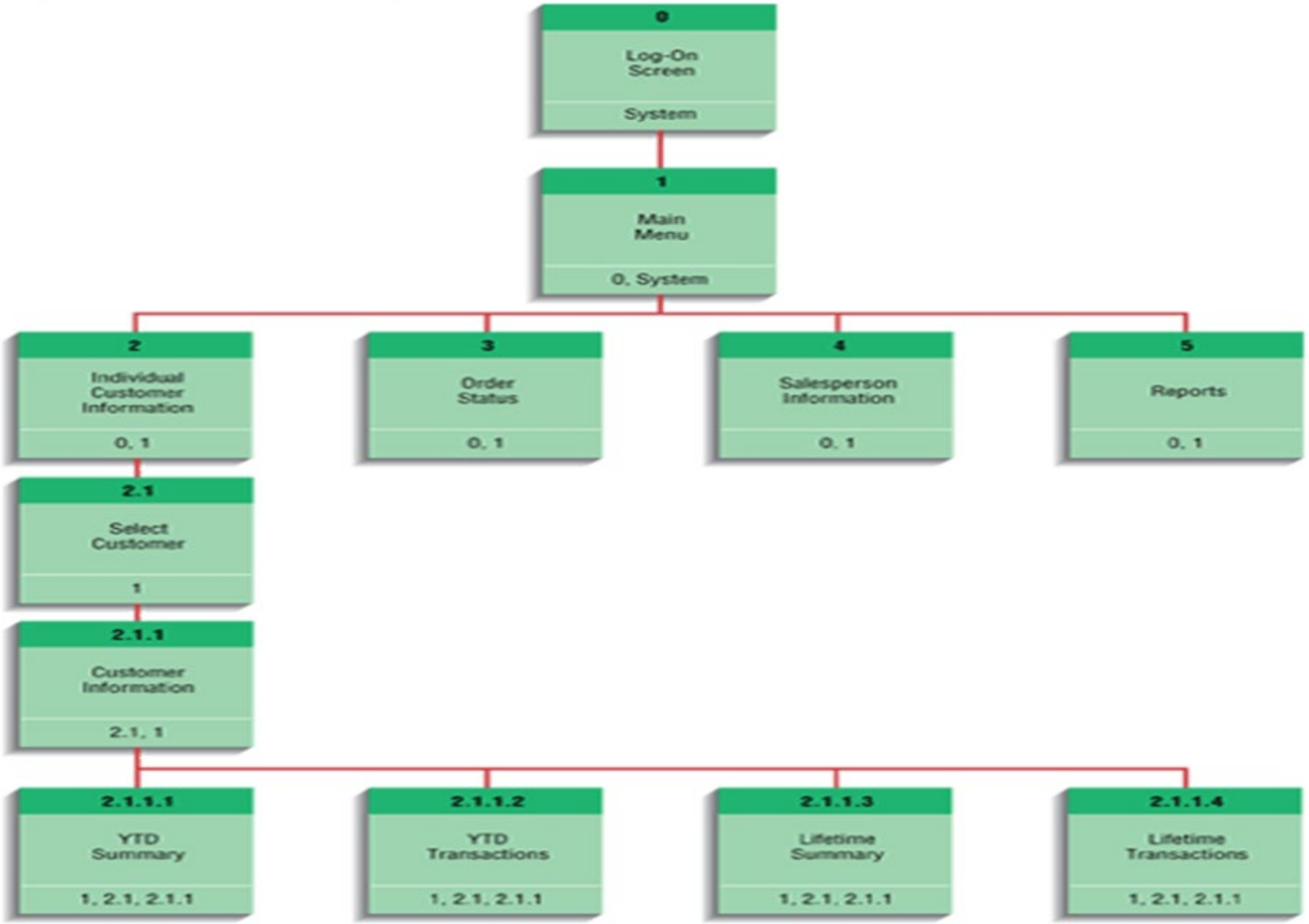
- Arayüz tasarımının amaçlarından biri veri girişi hatalarını azaltmaktır
- Sistem analistinin rolü, veri girişi hatalarını önlemek, tespit etmek ve düzeltmek için kullanıcı hatalarını tahmin etmek ve sistemin arayüzlerine yönelik özellikler tasarlamaktır.
- **Tablo 8-9** veri girişi hata türlerini açıklamaktadır
- **Tablo 8-10** sistem tasarımcıları tarafından hataları tespit etmek için kullanılan teknikleri listelemektedir

ÖRNEK

- Pine Valley Furniture'deki (PVF) pazarlama müdürünün, satış ve pazarlama personelinin herhangi bir PVF müşterisinin yılbaşından bugüne işlem etkinliğini inceleyebilmesini istediğini varsayalım.
- **Diyalog dizisi**
- 1. Bireysel müşteri bilgilerini görüntüleme isteği
- 2. İlgilendiğiniz müşteriyi belirtin/müşteriyi seçin
- 3. Yılbaşından bugüne işlem özeti ekranını seçin
- 4. Müşteri bilgilerini inceleyin
- 5. Sistemden ayrılın

Co-funded by the Erasmus+ Programme of the European Union

Dialogue Diagram for the Customer Information System at Pine Valley Furniture



Sonuç

- Mantıksal veritabanı tasarımı, belirli bir iş ortamındaki varlıkların niteliklerinin, ilişkisel bir veritabanının tabloları gibi veritabanı yapılarında nasıl düzenleneceğine karar verme sürecidir.
- Mantıksal veritabanı tasarımının amacı şirketin iş ortamını doğru şekilde yansıtan iyi yapılandırılmış tablolar oluşturmaktır.
- Mantıksal Veritabanı Tasarımı, tanımlanan varlıkların ve bunların birbirleriyle nasıl ilişkili olduğunun ve ne tür verilerin depolanacağının alt düzey bir açıklamasına sahiptir.
- Bu model, işletmenin tüm gereksinimlerinin toplanıp toplanmadığını belirler.
- **Mantıksal veri modeli, bir veritabanının tüm kullanıcı görünümlerini temsil eder.**

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Veritabanının mantıksal tasarımı nedir?
2. Mantıksal veritabanı tasarımının 4 temel adımı kimlerdir?
3. Mantıksal veritabanı tasarımından Arayüz tasarlamanın parçası nedir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

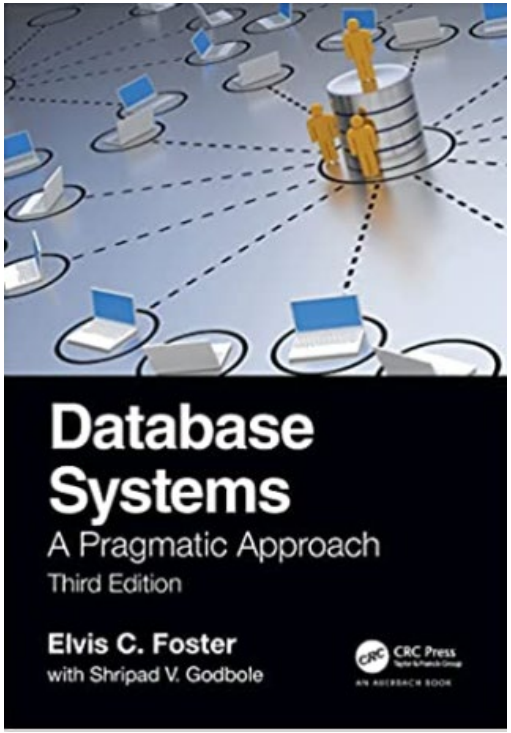
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

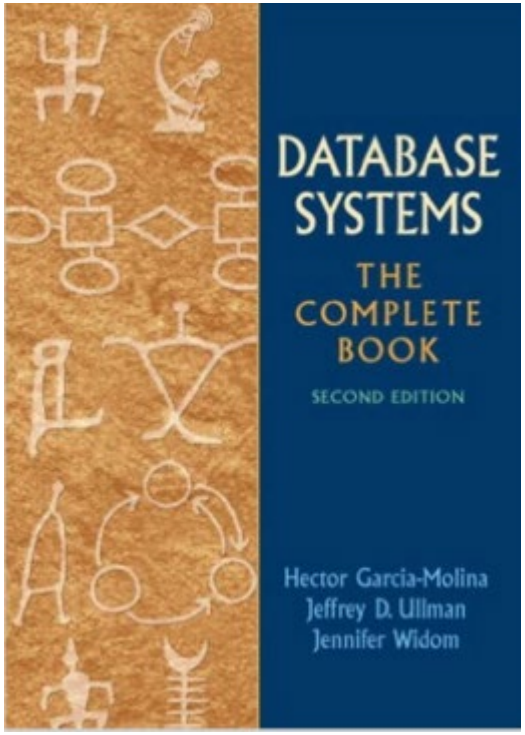
- **Modül 1. Veritabanlarının Teorik Temelleri**
- **Konu 2. Veritabanlarının mantıksal ve fiziksel temelleri**
- Ders 2. Veritabanlarının fiziksel temelleri

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANLARININ FİZİKSEL TEMELLERİ





Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.

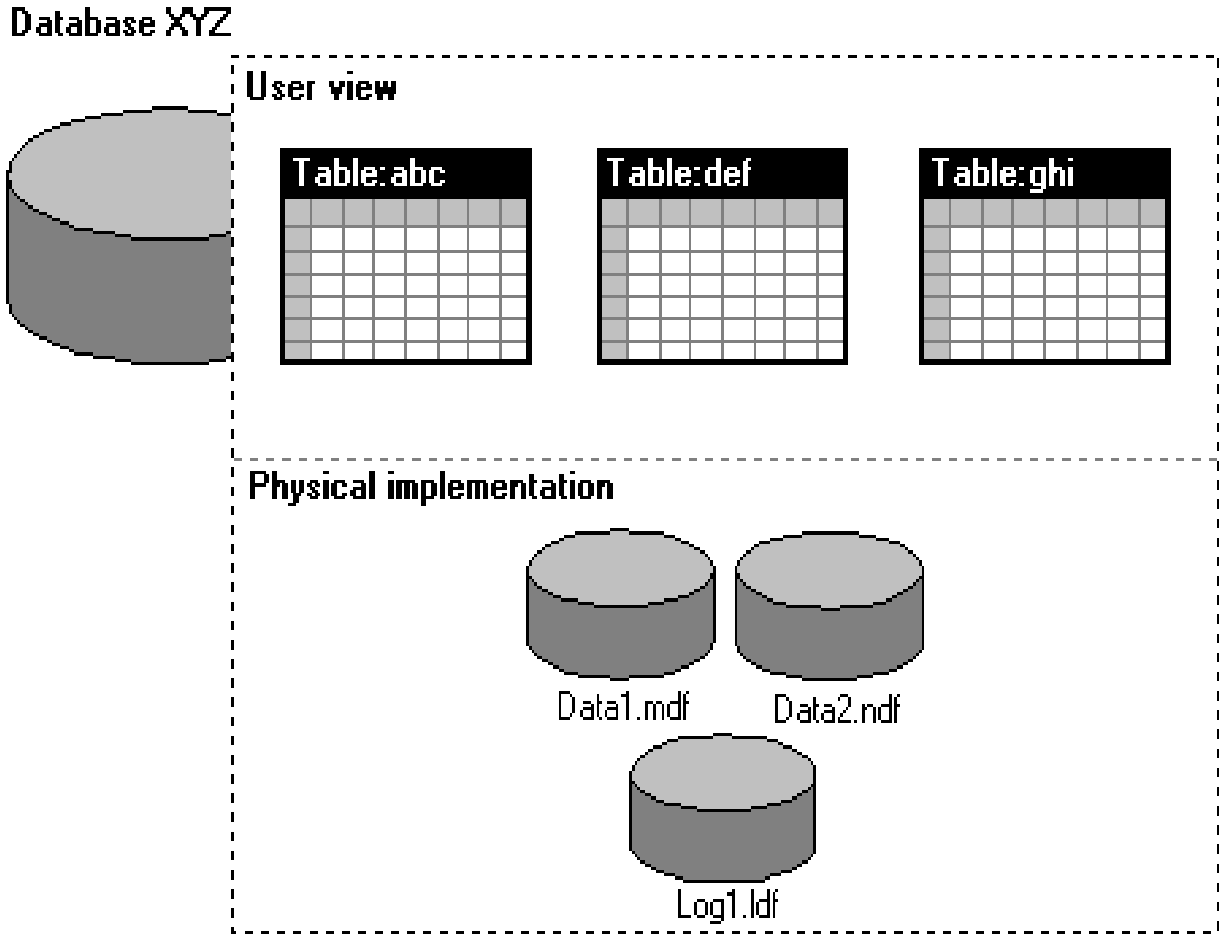
Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

Azalov, P. (1991) Database, Sofia.

GİRİŞ

- Bir veritabanındaki veriler, kullanıcıların görebileceği mantıksal bileşenler halinde düzenlenir.
- Veritabanı ayrıca fiziksel olarak diskte iki veya daha fazla dosya olarak uygulanır.
- Bir veritabanını kullanırken öncelikle tablolar, görünümler, prosedürler ve kullanıcılar gibi mantıksal öğelerle çalışırsınız.
- Dosyaların fiziksel uygulaması çoğunlukla şeffaftır. Genellikle yalnızca veritabanı yöneticisinin fiziksel uygulamayla çalışması gerekir.

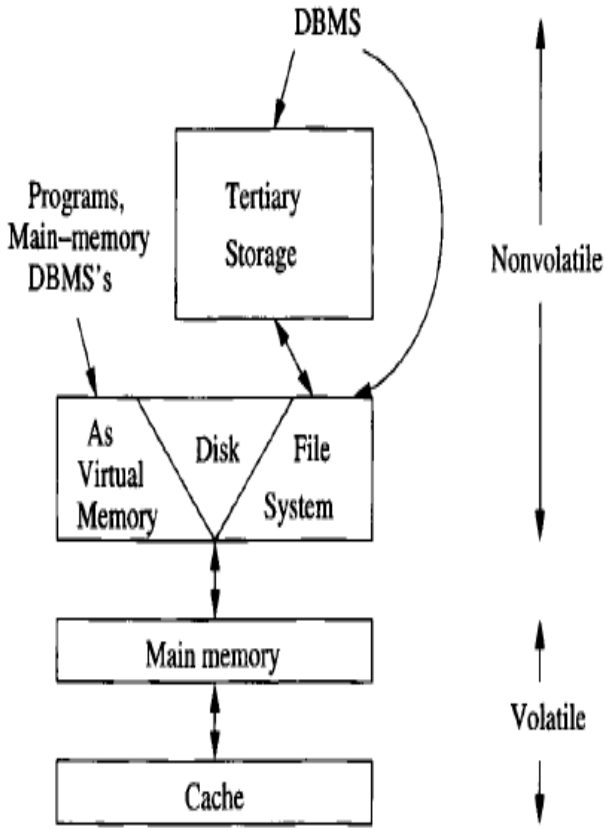
GİRİŞ


Database XYZ
User view
Table:abc
Table:def
Table:ghi
Physical implementation
Data1.mdf
Data2.ndf
Log1.ldf

GİRİŞ

- Tipik bir bilgisayar sistemi, verilerin saklanabileceği çeşitli bileşenlere sahiptir.

- Bu bileşenler, farklı sıralarda değişen veri kapasitesine sahiptir ve ayrıca verilere farklı erişim hızlarına sahiptir.



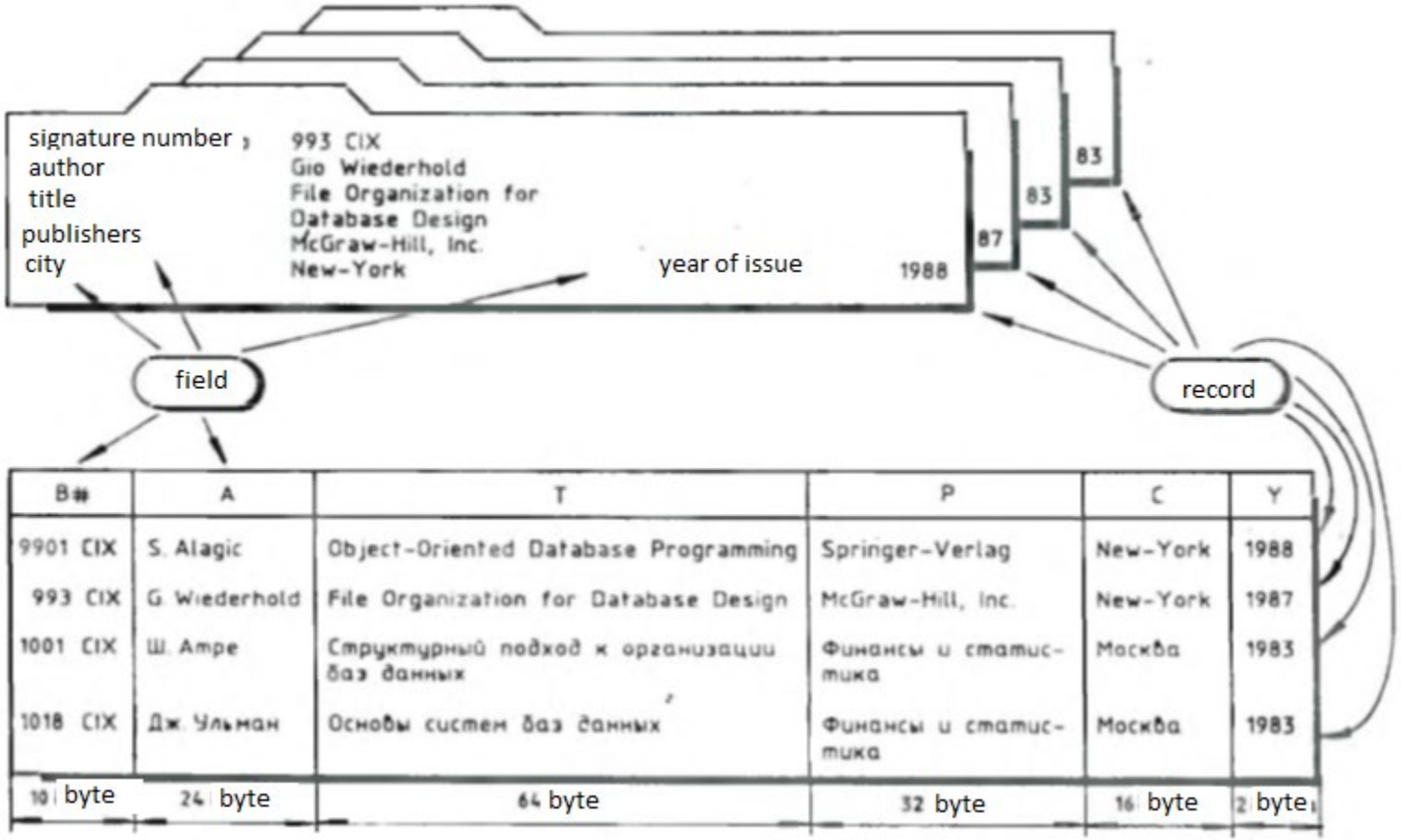
Bellek hiyerarşisi

DOSYALAR. TEMEL KONSEPTLER.

- KAYITLAR
- Aynı sınıftaki nesneler aynı özellikler kümesiyle tanımlanır.
- Her özellik, özelliğin alanı adı verilen kesin olarak tanımlanmış bir değerler kümesinden değerler alabilir.
- Bir sınıfın bireysel niteliklerinin değer kümesi, genellikle kayıt olarak adlandırılan, o sınıfın belirli bir nesne örneğini temsil eder.
- Aşağıdaki şekil, her biri ayrı bir kopyayı (bir kitabı) temsil eden kütüphane indeks fişlerinin örneklerini vermektedir.

DOSYALAR. TEMEL KONSEPTLER.

➢ KAYITLAR



DOSYALAR. TEMEL KONSEPTLER.

➢ KAYITLAR

- Kayıtları oluşturan veri birimlerine (öznitelik değerlerine) alan da denir.
- Her alanın özelliği uzunluğudur. Kaydın harici gösteriminde bu uzunluk karakter sayısıyla, dahili gösteriminde ise bayt sayısıyla ölçülür.
- Dolayısıyla kaydın toplam uzunluğu, bireysel alanların uzunluklarının toplamıdır.
- Çoğu zaman, nesne sınıfının farklı kopyalarına ilişkin bireysel kayıtların uzunluğu aynıdır. Bu tür kayıtlara sabit uzunluklu kayıtlar denir.

DOSYALAR. TEMEL KONSEPTLER.

➢ KAYITLAR

- Değişken uzunluklu kayıtların kullanıldığı durumlar da vardır. Bu aşağıdaki durumlarda gereklidir:

а) alanların bir veya daha fazlası değişken uzunluktadır;

б) birkaç kez tekrarlanan bir veya bir grup alan;

в) alanların bazıları değişkendir.

- Her kitap için özet özelliğinin eklenmesine izin verin. Her bir slipin ilgili alanının içeriği, uzunluğu önceden belirlenemeyen bir metin olacaktır. Böyle bir alana değişken uzunluklu alan adı verilir.

Co-funded by the Erasmus+ Programme of the European Union

DOSYALAR. TEMEL KONSEPTLER.

➢ KAYITLAR

- Çoğu zaman bir kitabın veya makalenin içeriğini kısaca anlatmak için özet yerine anahtar kelimeler kullanılır. Bireysel kitaplara atıfta bulunan anahtar kelimelerin sayısı genellikle değişir. Kitabın içeriğini daha kesin olarak karakterize etmek gerektiğinde daha fazla anahtar kelime kullanılır.
- Anahtar kelime özelliğinin değerler için karakter dizilerine sahip olduğu, örneğin 16 karakter uzunluğunda olduğu varsayılabilir, ancak farklı kitaplar için anahtar kelime sayısı farklı olacaktır.

DOSYALAR. TEMEL KONSEPTLER.

➢ KAYITLAR

- Aynı kayıtta bir veya daha fazla kez bulunabilen bir alan veya alan grubuna (genellikle sabit uzunlukta) **yinelenen grup** adı verilir.
- Kütüphanelerde kitapların yanı sıra dergiler de muhafaza edilmektedir. Kitap ve dergilerin bazı nitelikleri ortaktır ancak farklı olan nitelikleri de vardır.
- Gerektiğinde kitap ve dergiler aynı sınıfın nesneleri (kopyaları) olarak kabul edilebilir.

DOSYALAR. TEMEL KONSEPTLER.

➢ KAYITLAR

- Bu, kitap ve dergilere karşılık gelen iki farklı formatta giriş yapar.
- Bu tür kayıtlara değişken formatlı kayıtlar veya değişken kayıtlar da denir.

Bir tablonun her satırına kayıt denir. Kayıtlar, bireysel bilgi parçalarının saklandığı yerdir. Her kayıt bir veya daha fazla alandan oluşur. Alanlar tablodaki sütunlara karşılık gelir.

DOSYALAR. TEMEL KONSEPTLER.

➢ DOSYALAR

Aynı sınıftaki nesnelere karşılık gelen kayıtlar kümesine **dosya** denir.

Ayrıca dosyaların aynı verileri harici bellek setlerinde depolamak için kullanıldığı da söylenir. Her dosya iki boyutlu bir tablo olarak düşünülebilir.

Tablodaki satırlar kayıtları temsil eder ve bunların sayısı dosya boyutlarından birini belirler. Kayıt sayısı birkaç kayıttan birkaç yüz bin kayda kadar değişen dosyalar bulunmaktadır.

DOSYALAR. TEMEL KONSEPTLER.

➢ DOSYALAR

- Dosyadaki bireysel kayıtların birbirine göre düzenlenme şekli **dosyanın organizasyonunu** belirler.
- Bazen, gerçek veriler dışında, tek tek kayıtları bulmayı kolaylaştırmak için dosyaların içinde veya dışında ek bilgiler saklanır.
- Dosyadaki kayıtlar, istenilen kayıtların sahip olması gereken belirli özellikler belirtilerek aranır.
- Bu özellikler formüle edilirken nesne sınıfının tanımlanmasında kullanılan nitelik adları belirtilir.

DOSYALAR. TEMEL KONSEPTLER.

➢ DOSYALAR

- Bir dosya kaydının aranmasını veya sıralanmasını sağlayan özniteliklere anahtar olarak atıfta bulunmak gelenekseldir.
- Değerleri dosyanın tamamındaki kayıtları benzersiz şekilde tanımlayan bir anahtar, birincil anahtardır.
- Çoğu zaman birincil anahtar tek bir öznitelik tarafından tanımlanır, ancak birden fazla öznitelikten oluştuğu durumlar da vardır. Bu durumda birincil anahtara bileşik anahtar denir.

DOSYALAR. TEMEL KONSEPTLER.

➢ DOSYALAR

- Dosyadaki kayıtların aranma ve seçilme şekli **erişim** yöntemini belirler.
- İki ana erişim yöntemi türü vardır:
- tutarlı erişim yöntemi

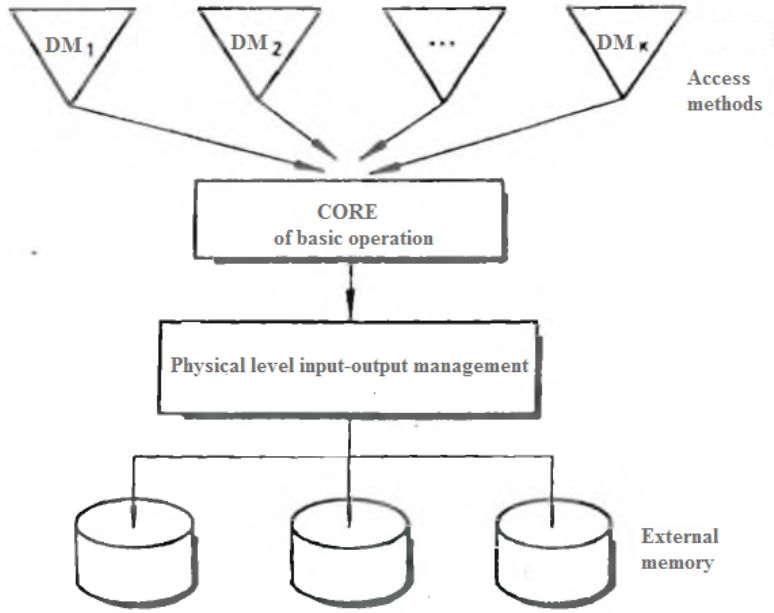
Ve

- anahtar erişim yöntemi.
- İkinci tipte en çok kullanılan yöntemler ise hashleme ve indeksleme yoluyla erişim yöntemleridir.
- Dosya yönetimi, dosya sistemi adı verilen özel bir sistem tarafından yapılır.

DOSYALAR. TEMEL KONSEPTLER.

- DOSYALAR

Her dosya sistemi temel işlemlerden oluşan bir çekirdek ve bir dizi farklı erişim yöntemini içerir:



Bir dosya sistemi mimarisinin genel görünümü

DOSYALAR. TEMEL KONSEPTLER.

➢ DOSYA

- Temel işlemler, dosya oluşturma ve yok etme, disk alanı tahsisi, bir disk paketindeki tüm dosyalar için katalog tutma ve giriş-çıkış arabelleklerini yönetmedir.
- Bu işlemler, çekirdeğin yükseltilmesini temsil eden tüm erişim yöntemleri tarafından kullanılır.
- Erişim yöntemlerinin her biri, fiziksel kayıtları (bloklar) okumak veya yazmak için G/Ç işlemleri aracılığıyla ilgili dosyalardan gelen verilerle etkileşime girer.

DOSYALAR. TEMEL KONSEPTLER.

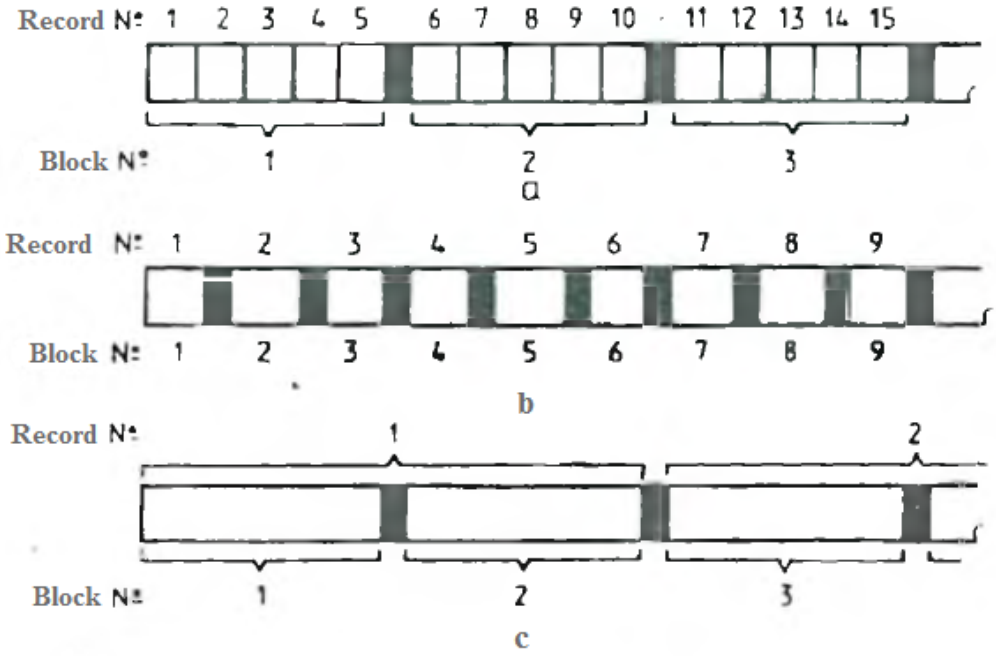
➢ DOSYALAR

- Genellikle bir blok birden fazla (mantıksal) kayıt içerir ve bu durumda kayıt engellemeden bahsederiz.
- Ancak bir kaydın birden fazla bloğa yerleştirildiği durumlar da vardır (sonraki slayttaki şekle bakın)

Modern dosya sistemleri çeşitli erişim yöntemlerini içerir.

DOSYALAR. TEMEL KONSEPTLER.

➢ DOSYALAR



Fiziksel ve mantıksal kayıtlar
a – kayıtları engellemek. Engelleme katsayısı K = 5;
b – engelleme kaydı yok;
c - bir mantıksal kayıt iki blokta bulunur.

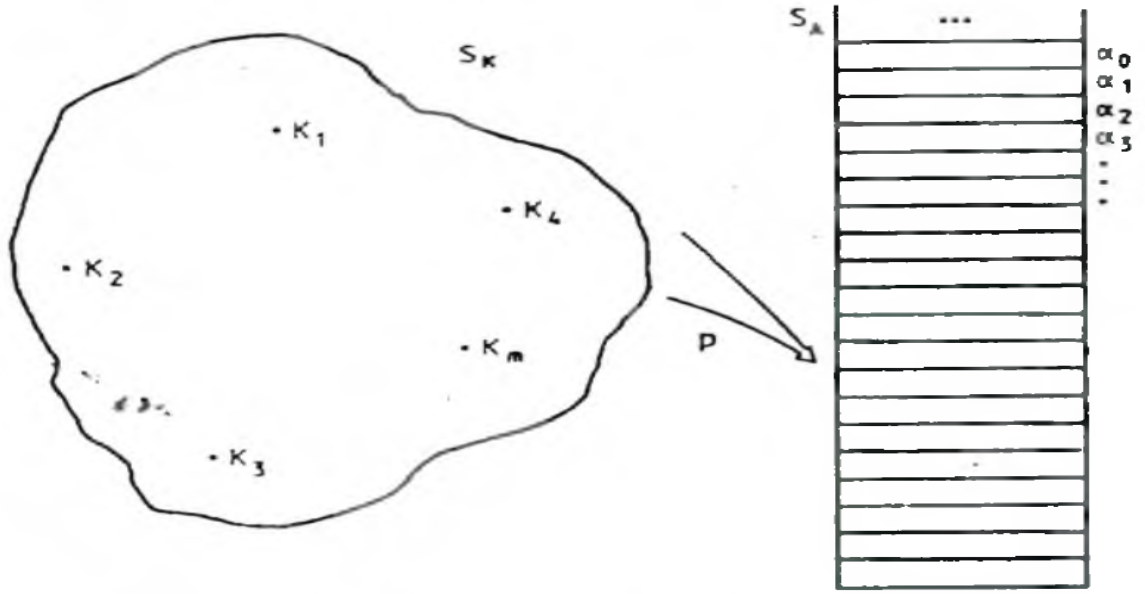
FİZİKSEL VERİ ORGANİZASYONU GÖREVLERİ.

- Somut bir fiziksel veritabanı oluştururken çözülmesi gereken önemli bir sorun, çeşitli veri yapılarının (tablolar, ağaçlar, ızgaralar vb.) bit dizileri halinde fiziksel olarak nasıl temsil edileceğidir.
- Bu görevin bir sonucu olarak, diğer bazı problemler de çözüldü:

1.İlgili fiziksel kaydın konumu belirli bir anahtarla ve muhtemelen minimum sürede nasıl belirlenir?

- Aslında bu görev, belirli bir SK dosyası (anahtar alanı) için izin verilen anahtar değerleri kümesi ile belirli bir harici bellek alanı SA'sındaki mevcut adresler kümesi arasında bir yazışma (kural, prosedür, işlev) kurulmasına yol açar.

FİZİKSEL VERİ ORGANİZASYONU GÖREVLERİ.



Sınıf kümesi ile adres kümesi arasındaki korelasyon

- Korelasyon doğrudan bir fonksiyon tarafından tanımlanabilir, bu durumda buna doğrudan erişim yöntemi denir ve diğer durumlarda indeks adı verilen ek bir tablo kullanılabilir, bu durumda buna indeks erişim yöntemi denir.

FİZİKSEL VERİ ORGANİZASYONU GÖREVLERİ.

- Ana görevin sonucu olan diğer alt görevler kısaca şu şekilde formüle edilebilir:

2. Çok tuşlu aramanın mümkün olması ve verimli bir şekilde gerçekleştirilmesi için veriler nasıl organize edilir?

3. Hiyerarşik ve ağ yapılarının fiziksel temsili nasıl olmalı ve güncelleme işlemi de dahil olmak üzere çeşitli işlemleri gerçekleştirme yöntemleri neler olmalıdır?

4. Minimum belleğin kullanıldığı verilerin fiziksel temsili ne olmalıdır, temsil maksimum yoğunlukta mı olmalı?

FİZİKSEL VERİ ORGANİZASYONU GÖREVLERİ.

- Bu görevlerden bazılarını en iyi şekilde çözmek, diğer bazılarının daha az verimli çözümlerine yol açar.
- Bu, bireysel görevler arasındaki gereksinimlerde var olan çelişkinin bir sonucudur.
- Bu nedenle, fiziksel veritabanının özel tasarımında, bu durumda ilgilendiğimiz sorunları en iyi şekilde çözmemize izin verecek şekilde belirli tavizler vermek zorundayız.

Verilerin fiziksel organizasyonuna yönelik temel gereksinimlerden, depolama yöntemi ve harici bellekte depolanan verilere erişim yönteminin (algoritmasının), bir Veritabanı Yönetim Sistemi (DBMS)'nin performansını etkileyen en önemli faktörler olduğu açıktır.

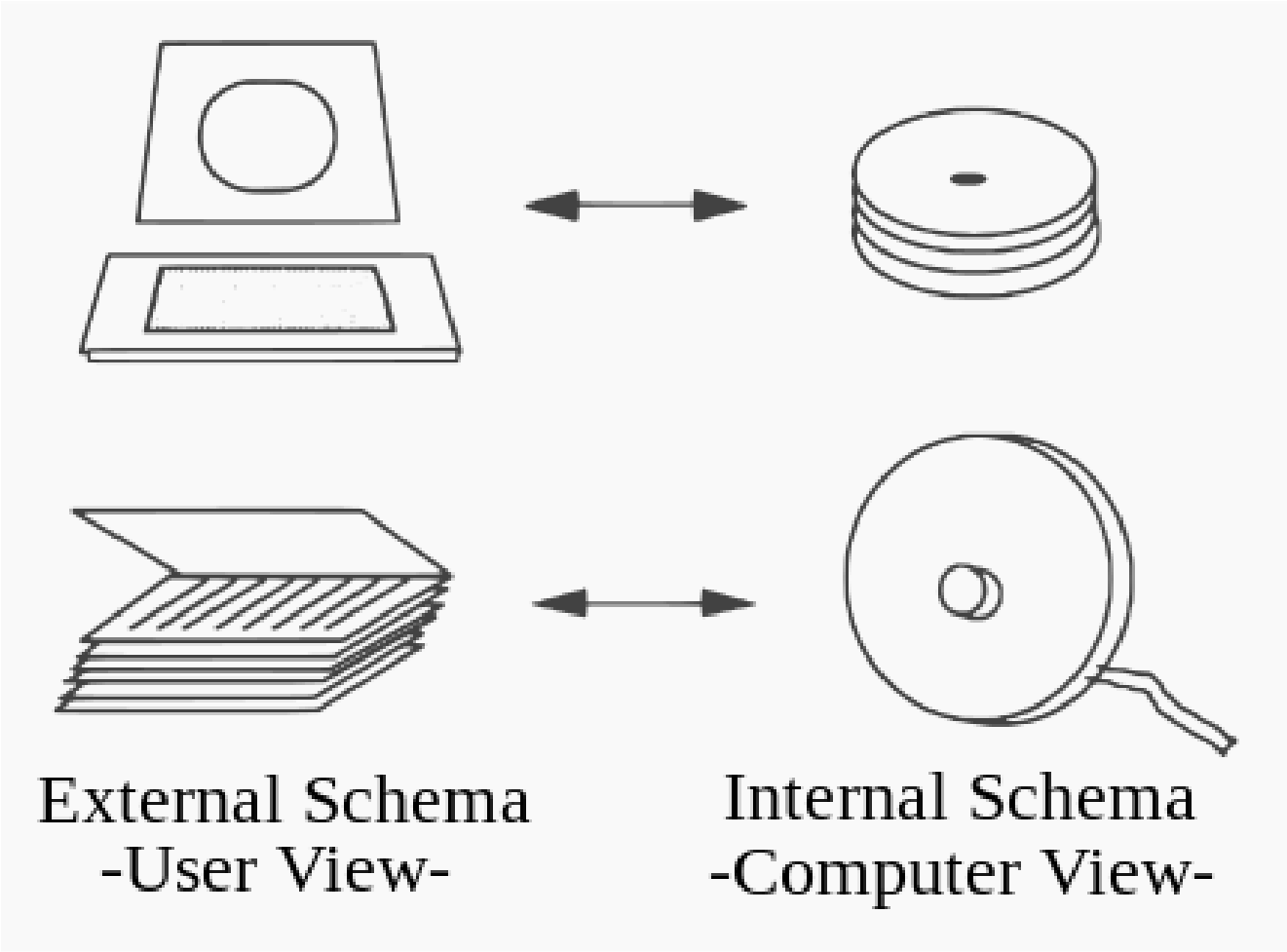
FİZİKSEL VERİ ORGANİZASYONU GÖREVLERİ.

- Dolayısıyla her bir Veritabanı Yönetim Sistemi (DBMS) 'nin iç şemasının bu özellikleri hem erişim verimliliği hem de veri depolama verimliliği açısından değerlendirilir.
- **Erişim verimliliği,** veritabanından mantıksal bir kayıt almak için gereken ortalama G/Ç işlemi sayısının karşılıklı değeri olarak tanımlanır.
- **Depolama verimliliği,** bir baytlık giriş verisini temsil etmek için gereken harici belleğin ortalama bayt miktarının karşılıklı değeri olarak tanımlanır.

FİZİKSEL VERİ ORGANİZASYONU GÖREVLERİ.

- Mantıksal dosyaların fiziksel temsilinin ek tablolar ve kontrol bilgileri kullandığı, o anda ihtiyaç duyulandan daha fazla boş bellek ayırdığı vb. burada akılda tutulmalıdır.
- Modern DBMS'lerde en sık kullanılan erişim yöntemleri s**ıralı, indeksli, ters çevrilmiş ve karma** erişim yöntemleridir ve bunlar bir sonraki derste tartışılacaktır.

Co-funded by the Erasmus+ Programme of the European Union



Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Veritabanındaki dosyaların fiziksel uygulaması nedir?
2. Tipik bir bilgisayar sisteminde verilerin depolanabileceği çeşitli bileşenler nelerdir?
3. Dosyanın organizasyonu nedir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

❑ Modül 1. Veritabanlarının Teorik Temelleri

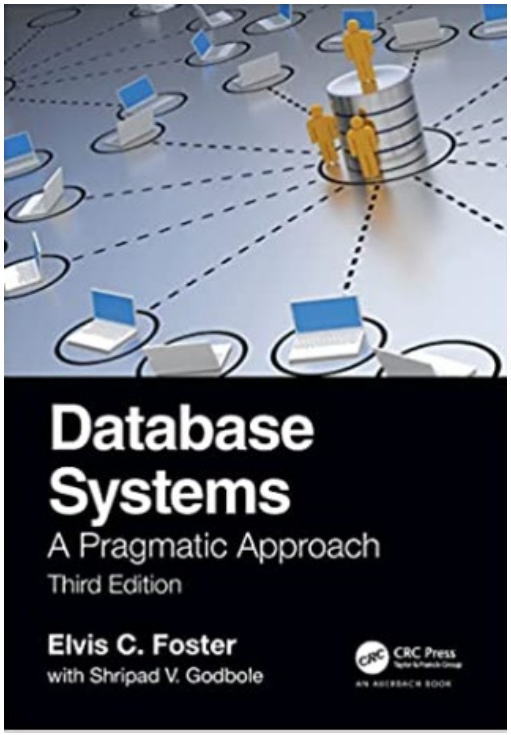
❑ Konu 3. Veri modelleri

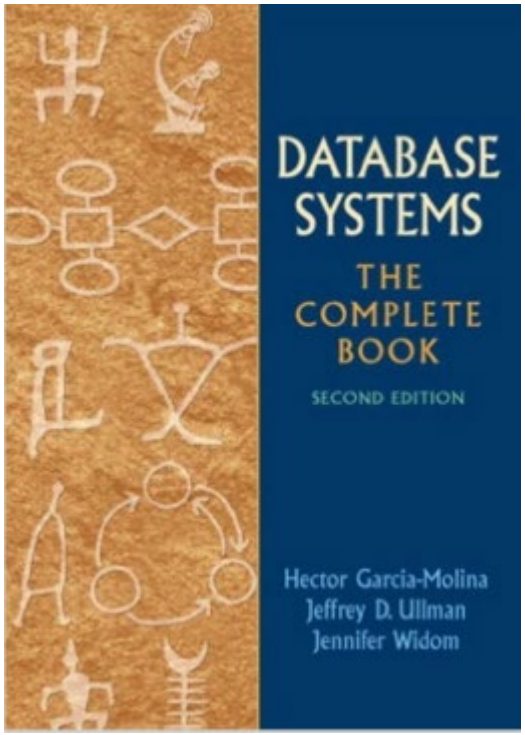
- ❑ Ders 1. Veritabanı Modeli Nedir? Veri modeli türleri.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANI MODELİ NEDİR? VERİ MODELLERİNİN TÜRLERİ.





Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.

Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

Azalov, P. (1991) Database, Sofia.

Co-funded by the Erasmus+ Programme of the European Union

GİRİŞ

- **"Veri modeli"** fikri, veritabanı sistemleri çalışmalarındaki en temel fikirlerden biridir. Bu derste bazı temel terminolojiyi tanımlayacağız ve en önemli veri modellerinden bahsedeceğiz.
- Günümüzde matematiksel, sibernetik, ekonomik, fiziksel ve daha birçok farklı bilgi alanına ait modeller bilinmekte ve üzerinde çalışılmaktadır.
- Modeller çizimler, yapılar, metinler, şemalar, denklemler, algoritmalar vb. şeklinde oluşturulur.

Co-funded by the Erasmus+ Programme of the European Union

GİRİŞ

- Her veritabanının şeması aynı zamanda bir modeldir.
- Belirli bir konu alanının, sınırlı sayıda gerçek dünya nesnesinin ve bunlar arasında var olan ilişkilerin tanımlarıyla temsil edilen bir modelidir.
- Veri modellerinin (DM) daha spesifik bir amacı vardır. Veritabanı şemaları için kullanılırlar, yani DM araçları, verilerin bilgisayarlar tarafından saklandığı ve işlendiği nesnelerin modellerini oluşturmak için kullanılır.

Co-funded by the Erasmus+ Programme of the European Union

GİRİŞ

- Burada DM (Veri Modeli) kavramına ve onun DB (Veritabanı) kavramıyla ilişkisine odaklanacağız.
- Her bir veritabanındaki verilerin organizasyonu en önemli özelliklerinden biridir ve kullanılan modelle doğrudan ilişkilidir.

Veri modeli nedir?

- Veri modeli, verileri veya bilgileri açıklamaya yönelik bir gösterimdir (gösterim sistemi). Açıklama genellikle üç bölümden oluşur:

1. Veri yapısı. Bir programın kullandığı veri yapısını açıklamak için C veya Java gibi programlama dillerindeki araçlara aşina olabilirsiniz: örneğin diziler, yapılar veya nesneler. Bir bilgisayardaki verileri temsil etmek için kullanılan veri yapıları, bazen veri tabanı sistemlerini tartışırken, fiziksel veriler için bir model olarak nitelendirilir; ancak gerçekte bunlar, fiziksel verilerin gerçekte temsil edilme biçiminden çok uzaktır.

Veri modeli nedir?

Veritabanı dünyasında veri modelleri, veri yapılarından biraz daha yüksek seviyededir ve bazen seviye farkını vurgulamak için kavramsal model olarak da anılır.

2. Veriler üzerinde işlemler. Programlama dillerinde veriler üzerinde yapılan işlemler genel olarak programlanabilen bir şeydir. Veritabanlarındaki veri modelleri için genellikle gerçekleştirilebilecek sınırlı sayıda işlem vardır. Genellikle sınırlı sayıda sorgu (bilgiyi getiren işlemler) ve değişiklikleri (veritabanındaki bilgileri değiştiren işlemler) gerçekleştirebiliriz. Bu sınırlama bir zayıflık değil, bir güçtür. İşlemlerin kısıtlanmasıyla programcıların veritabanı işlemlerini çok yüksek düzeyde anlatabilmesi mümkün olup, işlemleri verimli bir şekilde uygulayacak veritabanı yönetim sistemleri bilinmektedir.

Veri modeli nedir?

3. Veri kısıtlamaları. Veritabanlarındaki veri modelleri genellikle verilerin ne olabileceğine ilişkin kısıtlamaları açıklamanın bir yoluna sahiptir. Bu kısıtlamalar basitten (örneğin, "haftanın bir günü 1 ile 7 arasında bir tam sayıdır" veya "bir filmin en fazla bir başlığı vardır") çok karmaşık kısıtlamalara kadar değişebilir.

Önemli veri modeli türleri

- Günümüzde iki veri modeli veritabanı sistemleri için temel öneme sahiptir:
- **1. İlişkisel model,** nesne ilişkisel uzantıları içerir.
- **XML ve ilgili standartları içeren yarı yapılandırılmış model**.

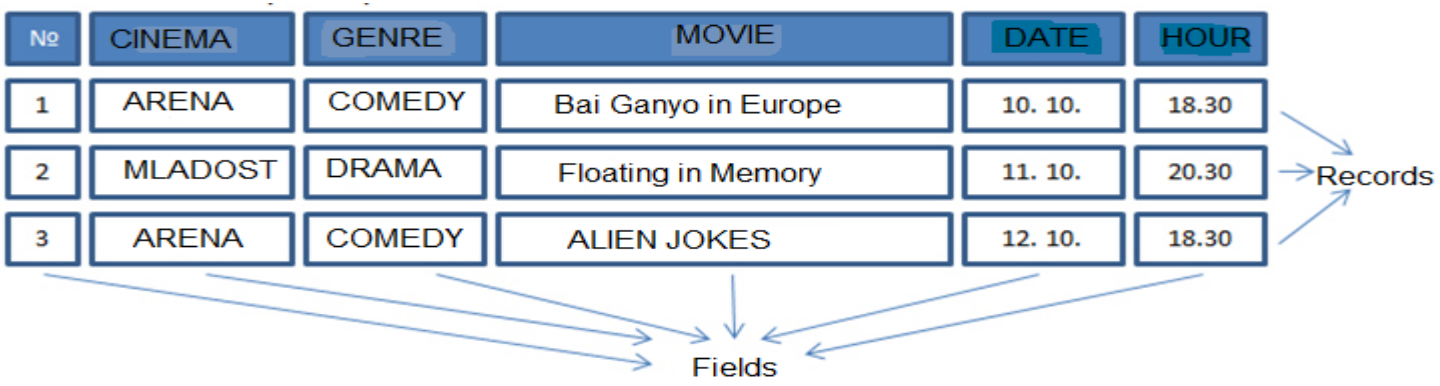
✓ *Tüm ticari veri tabanı yönetim sistemlerinde mevcut olan ilki (ilişkisel model) bu dersin konusudur ve ayrı bir derste detaylı olarak ele alınacaktır.*

✓ *XML'in ana prosedür olduğu yarı yapılandırılmış model, çoğu ilişkisel DBMS'nin ek bir özelliğidir ve bir dizi başka bağlamda da ortaya çıkar.*

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

NESNELER VE AÇIKLAMALARI

- Her veritabanı birbirine bağlı **nesnelerden** oluşur. Bu ilişkiler, nesneler ve onların özellikleri arasındaki ilişkiler belirtilerek veritabanında tasvir edilir.
- Aynı türdeki nesnelere ilişkin veriler **tablolara** yerleştirilir. Tablonun her satırı bir **nesneye (kayıt)** ilişkin verileri içerir ve her **sütun (alan)** karşılık gelen nesne özelliklerini içerir.



NESNELER VE AÇIKLAMALARI

- **"Nesne"** terimi tanıma tabi değildir. Nesneler vardır ve ayırt edilebilirler. Nesneler örneğin arabalar, kitaplar, insanlar, evler, konut projeleri vb.'dir.
- Ayrıca maddi olmayan nesneleri de dikkate alıyoruz - olaylar, fenomenler, duygular. Her nesnenin onu karakterize eden ve diğerlerinden ayıran özellikleri vardır.
- Örneğin, her kitabın karakteristik özelliği başlığı, yazarı, yayıncısı, yayın yılı vb.'dir.
- Kabaca, bir nesneyi ne kadar ayrıntılı tanımlarsak tanımlayalım, onun tam ve doğru tanımını vermek zordur.

NESNELER VE AÇIKLAMALARI

- Bununla birlikte, bireysel nesnelerin temel olmayan özelliklerinden soyutlarsak, her nesne için onu yeterince doğru bir şekilde temsil eden ortak bir karakteristik özellikler kümesi tanımlanabilir.
- Bu şekilde, bazı ortak özelliklere sahip olan aynı (benzer) nesne kümeleri düşünülebilir.
- Bu tür kümelerin her biri, karakteristik özelliklerinin tamamının veya bir kısmının değerlerinde birbirinden farklı olan nesneler içerecektir.
- Örneğin, her arabayı karakterize eden özelliklerin plaka numarası, sahibi ve üretim yılı olduğunu varsayarak "kişisel arabalar" nesne kümelerini ele alalım.

NESNELER VE AÇIKLAMALARI

- Bu kümedeki nesnelerin birkaç örneği tabloda verilmiştir:

Plate number	Owner	Year of manufacture
C 1584	Ivan Petrov	1985
BH5047	Todor Ivanov	1988
C36382	Hristo Ganchev	1979
ПА4766	Todor Ivanov	1989

- Aynı tür nesne sınıflarına ait nesne kümelerini ve bunları karakterize eden özellikleri - nitelikleri çağırmak gelenekseldir.
- Her nitelik, bir adla ve bir dizi izin verilen değerle veya daha yaygın olarak adlandırıldığı gibi bir nitelik alanıyla tanımlanan bir miktardır.

Co-funded by the Erasmus+ Programme of the European Union

NESNELER VE AÇIKLAMALARI

- Farklı niteliklerin aynı alana sahip olmasına izin verilir. Örneğin, "Fakülte Numarası" ve "Ofis Numarası" farklı nesne sınıflarının nitelikleri olabilir, ancak aynı alanla - pozitif tamsayılar kümesiyle.
- "Öğrencinin adı", "Yönetmenin adı", "Yazarın adı" da ortak bir alana sahip olan diğer niteliklerin örnekleridir - en fazla 40 karakterden oluşan alfabetik dizeler kümesi.
- Her nesne sınıfı için, sınıfın her nesnesini doğrudan tanımlayan **bir veya birkaç nitelik grubu** tanımlanabilir.

NESNELER VE AÇIKLAMALARI

- Bu öznitelik veya öznitelikler grubuna nesne sınıfı anahtarı adı verilir.
- Her nesne sınıfının en az bir anahtarı vardır. Bu, nesne sınıflarının yalnızca birbirinden farklı nesneler içermesi gerçeğinden kaynaklanır.
- *Örnekler:*

1. “ÇALIŞAN” nesnelerinin sınıfı

- Sınıf nitelikleri: **ofis numarası**; adres; doğum tarihi; eğitim; ders; departman; maaş bordrosu.

- "ÇALIŞAN" sınıf anahtarı "**ofis numarası**" özelliğidir.

NESNELER VE AÇIKLAMALARI

- *Örnekler*:

2. “SATIN ALMA” nesnelerinin sınıfı

Nesnelerin sınıfı nitelikleri: fatura numarası; satın almak; sayı; tarih; alıcı.

Bir faturayla birden fazla ürün satın alınabildiğinden "fatura numarası" özelliğinin "SATIN ALMA" sınıfındaki ürünleri benzersiz bir şekilde tanımlamadığı anlaşılmaktadır.

Ancak ilk iki özelliğin birlikte bu sınıftaki nesneleri benzersiz bir şekilde tanımladığı görülebilir.

NESNELER VE AÇIKLAMALARI

- *Örnekler*:

Attribute name	No.	Name	Unit	Marital status	Position	Commencement of work
Description of one emplooyee	2867	Angel Hristov	05	1	lathe operator	150681
	3704	Stefan Toshev	08	1	caster	280475
	5663	Boris Kostov	05	1	lathe operator	170580
	8457	Kiril Penev	05	0	lathe operator	261182
	9680	Nikola Velev	12	1	miller	190979
	Values from one and the same field					

NESNELER VE AÇIKLAMALARI

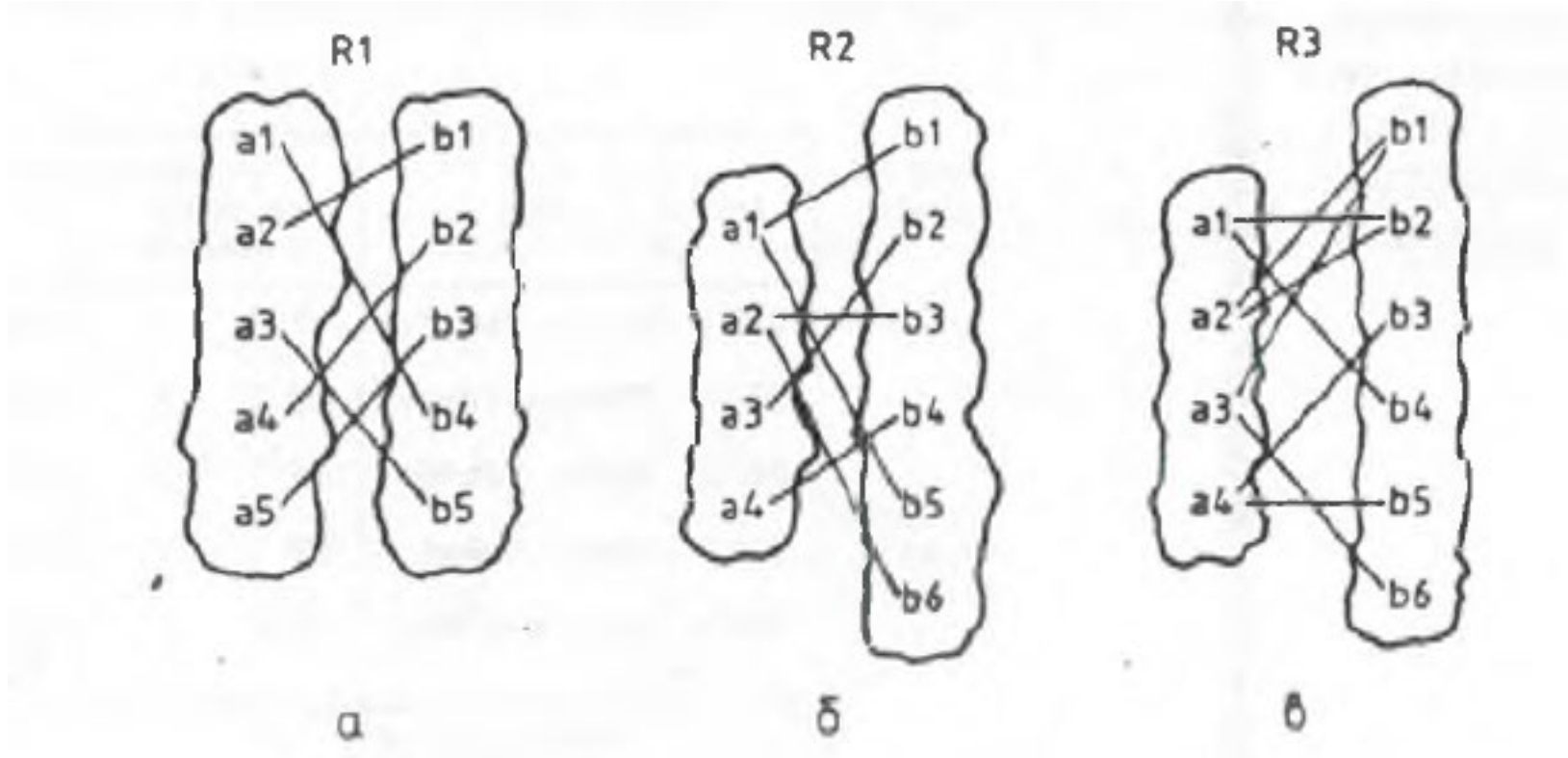
- Belirli bir konu alanını modelleyen herhangi bir DB'nin tasarımındaki önemli aşamalardan biri, nesne sınıflarının, bunların ilgili nitelik kümelerinin ve ayrıca çeşitli sınıfların nesneleri arasında var olan ilişkilerin seçilmesidir.
- Nesne modelleri oluşturulurken her bir sınıfı tanımlayan nitelikler arasındaki ilişkilerin incelenmesi gerekmektedir.
- Her bir özelliğin sahip olabileceği değerler kendi başına bir anlam ifade etmeyebilir. Bu, "Plaka numarası" özelliğinin C 1584, Вн5048, Сз6382 ve Пл4766 değerlerini ve "Üretim yılı" özelliğinin 1985, 1988, 1979 ve 1989 değerlerini ayrı ayrı ele aldığımızda görülebilir.

NESNELER VE AÇIKLAMALARI

- Bununla birlikte, C 1584 plaka numaralı bir otomobilin 1985 yılında üretildiğini ve Вн5047, Сз6382 ve Пл4766 plaka numaralı otomobillerin sırasıyla 1988, 1979 ve 1989 yıllarında üretildiğini öğrenirsek, bu iki değer kümesi arasındaki ilişki (uygunluk) şunu verir: onlara bir anlam
- Bu durumda "Plaka numarası" ve "Üretim yılı" niteliklerinin değerleri arasında bir ilişki kurulduğunu belirtmek için karşılık gelen öğeler aynı sıraya yerleştirilir.
- A ve B iki küme olsun ve R, A ile B arasında iki parçalı bir ilişki olsun, $R \subseteq x\ B$. Eğer $<a,b> \in R$ ise a'nın b ile R ilişkisi içinde olduğu da söylenir.

NESNELER VE AÇIKLAMALARI

- Aşağıdaki şekil, ilişkiye ait eleman çiftlerinin bir çizgiyle birleştirildiği üç ilişki örneğini göstermektedir.



NESNELER VE AÇIKLAMALARI

- Belirli bir R, $R \subseteq A \times B$ ilişkisinde, aşağıdaki şekilde tanımlanan R^{-1} ilişkisini ele alıyoruz:

$$\langle b,a \rangle \in R^{-1} \longleftrightarrow \langle a,b \rangle \in R$$

- R^{-1} ilişkisine R ilişkisinin tersi denir.
- Özel bir ilişkiler sınıfı, aşağıdaki durumlarda işlevsel ilişkiler (veya sadece işlev) olarak adlandırılır:

$$\langle a,b \rangle \in R \quad u \quad \langle a,c \rangle \in R \longrightarrow b=c$$

- Bu özellik, a ∈ A'nın her elemanının B'nin bir ve yalnızca bir elemanı ile R ilişkisi içinde olduğu gerçeğini ifade eder.

NESNELER VE AÇIKLAMALARI

- Yukarıdaki şekildeki örnekleri düşünün. Tanıma göre R1 ilişkisi fonksiyonel bir ilişkidir:

$$R1 = \{ \langle a1,b4\rangle, \langle a2,b1\rangle, \langle a3,b5\rangle, \langle a4,b2\rangle, \langle a5,b3\rangle \}$$

- R1 ilişkisinin tersi $R1^{-1}$ aynı zamanda fonksiyonel bir ilişkidir.
- R2 ilişkisi işlevsel değildir çünkü:

$$\langle a1,b1\rangle \in R2$$
$$\langle a1,b5\rangle \in R2 \quad \text{u} \quad b1 \neq b5$$

Co-funded by the Erasmus+ Programme of the European Union

NESNELER VE AÇIKLAMALARI

- İki nesne kümesi üzerinde tanımlanan ilişkiler kullanılarak 1:1, 1:N veya M:N biçiminde ilişkiler oluşturulabilir.
- *Bir ilişki, ters R^{-1} ilişkisi gibi işlevsel olan bir R ilişkisi tarafından kuruluyorsa 1:1 biçimindedir.*
- *R işlevsel değilse ve R^{-1} işlevselse ilişki türü 1:N'dir.*
- *Ve son olarak, eğer R ve R^{-1} işlevsel değilse ilişki türü M:N'dir.*
- Farklı ilişki türlerine birkaç örnek aşağıda verilmiştir:

NESNELER VE AÇIKLAMALARI

- *Örnekler:*

1. Her kulüp futbol takımının tek bir son sınıf öğrencisi vardır.

koç.

2. Her futbolcu yalnızca bir futbol takımının üyesidir ancak bir takım birçok futbolcudan oluşur.

NESNELER VE AÇIKLAMALARI

- *Örnekler:*

3. Bir futbolcu, kulüp takımı dışında milli takımda da oynayabilir. Böylece her futbolcu eşleştirilir

bir veya iki takım ve her takımda çok sayıda oyuncu var.

- Bir kümenin bir elemanı diğer kümenin 0, 1 veya daha fazla elemanıyla eşleştirilebiliyorsa bu durum grafiksel olarak çift okla gösterilir. Örnek 2 ve 3'ün durumları böyledir.

NESNELER VE AÇIKLAMALARI

- Nesne sınıfları arasındaki ilişkilerin temsili her DM'ye özeldir.
- Nitelikler arasındaki ilişkiler tipik olarak dikkate alınan şekilde temsil edilirken, bireysel sınıfların nesneleri arasındaki ilişkilerin temsilinde daha fazla çeşitlilik vardır.
- En popüler yaklaşımlar, nesne sınıfları arasındaki ilişkilerin ilişkiler veya grafikler (ağlar) ile temsil edildiği yaklaşımlardır.
- Bunlar en yaygın modellerin (ilişkisel ve grafik DM'ler) temelini oluşturur.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Veritabanı Yönetim Sistemi (DBMS)'deki veri modeli nedir ve ana veri modeli türleri nelerdir?
2. Veri modeli, verileri veya bilgileri açıklamaya yönelik bir gösterimdir (gösterim sistemi). Açıklama genellikle formdan oluşur? …
3. Veritabanı Yönetim Sistemi (DBMS)'deki terimler nesnesini açıklayın.

Co-funded by the Erasmus+ Programme of the European Union

Thank you for your attention!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

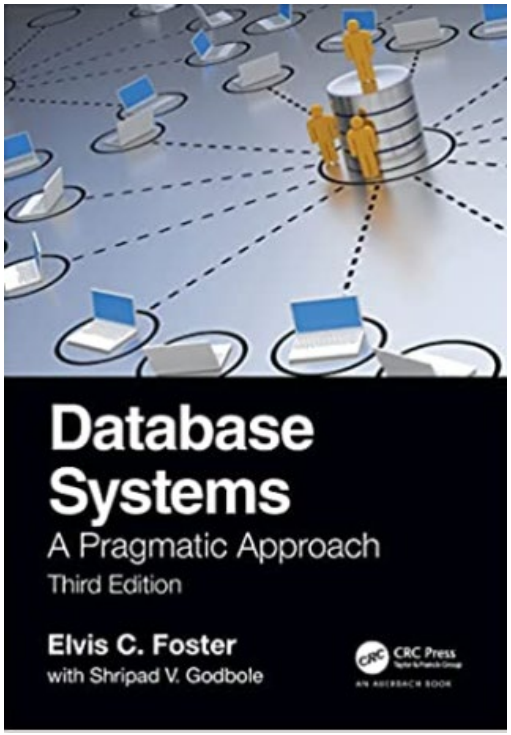
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

- **Modül 1. Veritabanlarının Teorik Temelleri**
- **Konu 3. Veri modelleri**
- Ders 2. Veritabanı Yönetim Sistemi (DBMS)'de Veritabanı Dilleri: Veri tanımlama dilleri. Veri işleme dilleri.

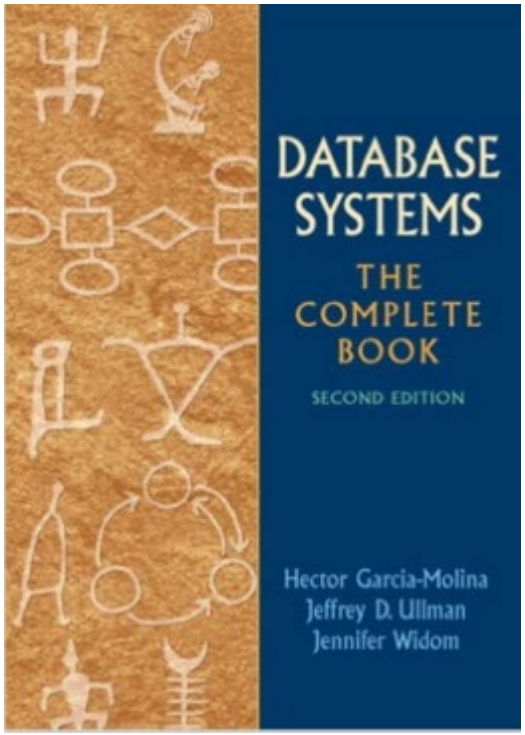
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Veritabanı Yönetim Sistemi (DBMS)'de Veritabanı Dilleri: Veri tanımlama dilleri. Veri işleme dilleri.



Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.



Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

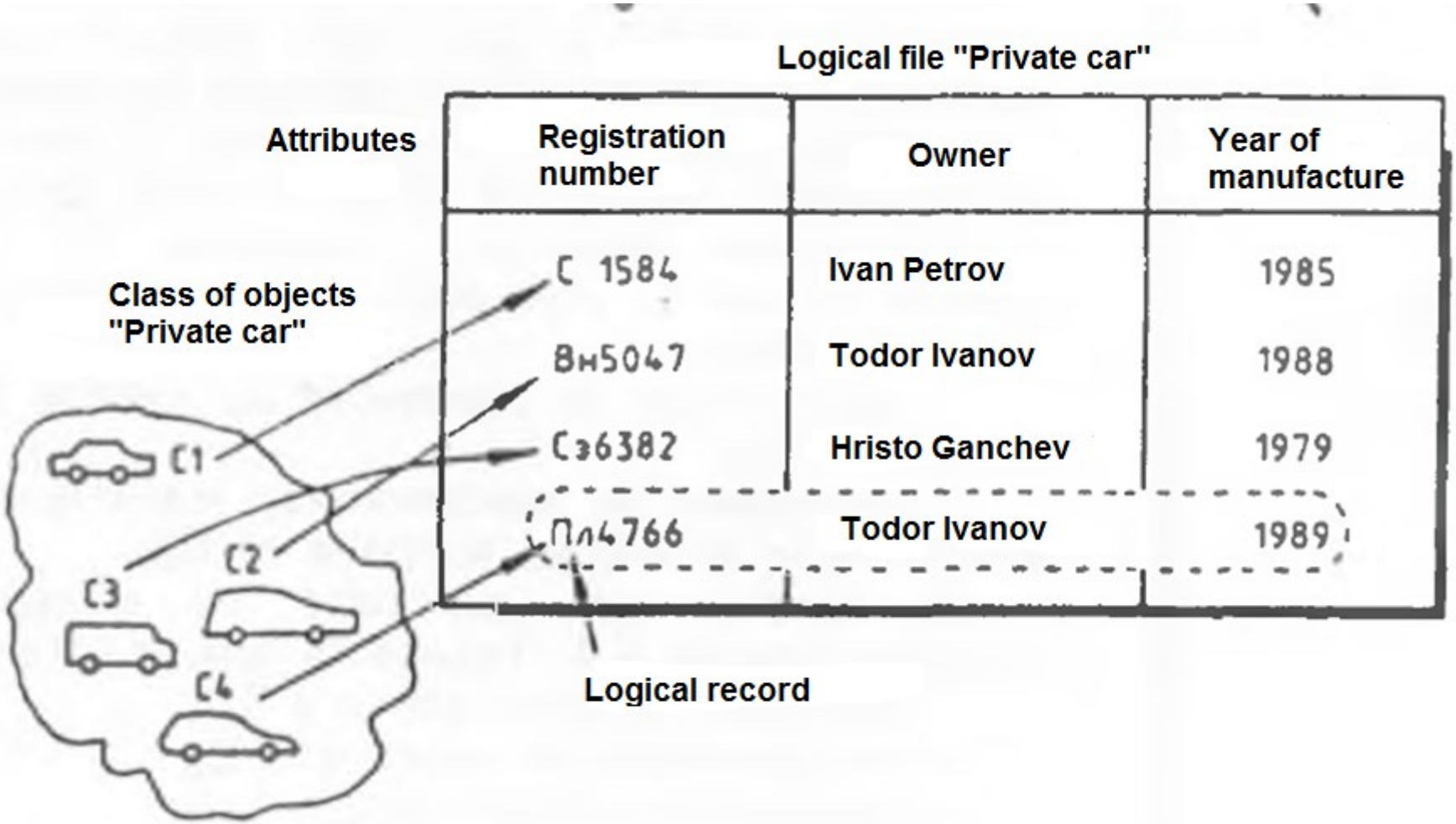
Azalov, P. (1991) Database, Sofia.

Co-funded by the Erasmus+ Programme of the European Union

Veri tanımlama dili

- Her veritabanı, bir veya daha fazla nesne sınıfı hakkındaki verileri ve ayrıca bunlar arasındaki mevcut ilişkileri temsil eden verileri depolar.
- Her nesne sınıfı sınırlı sayıda öznitelik ile temsil edilir.
- Çoğu zaman nesne sınıflarına ait veriler ayrı mantıksal dosyalara ayrılır.
- Bu, her nesne sınıfının bir tür mantıksal kayıtla temsil edildiği ve her sınıfın ayrı ayrı nesnelerinin açıklamalarının mantıksal kayıt örnekleri olduğu anlamına gelir:

Veri tanımlama dili



Co-funded by the Erasmus+ Programme of the European Union

Veri tanımlama dili

- Verilerin yüksek derecede mantıksal ve fiziksel bağımsızlığını sağlamak için, sırasıyla dış şema, kavramsal şema ve iç şema olarak adlandırılan üç düzeyde veri açıklaması dikkate alınır.

- **Dış şema** belirli bir kullanıcı programında yer alan veritabanının bir kısmının açıklamasıdır ve belirli kullanıcının veritabanı yapısına ilişkin görüşünü temsil eder. Aynı zamanda belirli bir kullanıcının eylem çevresini de tanımlar.

Veri tanımlama dili

- Veritabanı verileri, çoğunlukla manyetik diskler ve bantlar üzerinde harici bellekte depolanır ve bu, dosyalar, kayıtlar, alanlar, dizinler, karma yöntemleri vb. gibi depolama yapılarının açıklanmasını gerektirir. Verinin kendisinin fiziksel olarak depolanmasına yönelik olan bu açıklama, iç şemadır.
- **Dahili şema** harici bellekte saklanan verilere yönelik depolama yapılarının ve erişim yöntemlerinin bir spesifikasyonunu içeren, veritabanı verilerinin fiziksel parametrelerinin bir açıklamasıdır. Her veritabanının tek bir dahili ve bir veya daha fazla harici şeması vardır.

Veri tanımlama dili

- **Kavramsal bir şema** Nesnelerin sınıflarına ve bunların söz konusu konu alanındaki ilişkilerine karşılık gelen genel bir mantıksal açıklamadır.
- Her sınıf ve her ilişki, karşılık gelen bir dizi nitelikle temsil edilir.
- Ayrıca kavramsal şemanın, ilgili DM cinsinden ifade edilen, ele alınan konu alanının bir açıklaması olduğu da eklenebilir.
- **Örneğin**, ilişkisel modeldeki kavramsal şema, tabloların açıklamalarından ve bu tablolarda yer alan veriler üzerindeki mantıksal kısıtlamalardan oluşur.

Veri tanımlama dili

- Üç düzeyde veri tanımına sahip olmak, bir düzeydeki veri yapılarının başka bir düzeydeki veri yapılarına görüntülenebilmesini sağlayan bir araç gerektirir.
- Bu araçlar, prosedürler ve işlevlerle uygulanan ve veri temsilleri olarak adlandırılan algoritmalardır.
- Verilerin yapısal özelliklerinin yanı sıra, **veri tanımlama dilleri (DDL'ler)** nitelikler arasında bulunabilecek belirli bağımlılıkları ve kısıtlamaları da tanımlar.
- **Örneğin AE (çalışanın yaşı) özelliği için aşağıdaki koşul tanımlanabilir:** $17 < BC < 61$

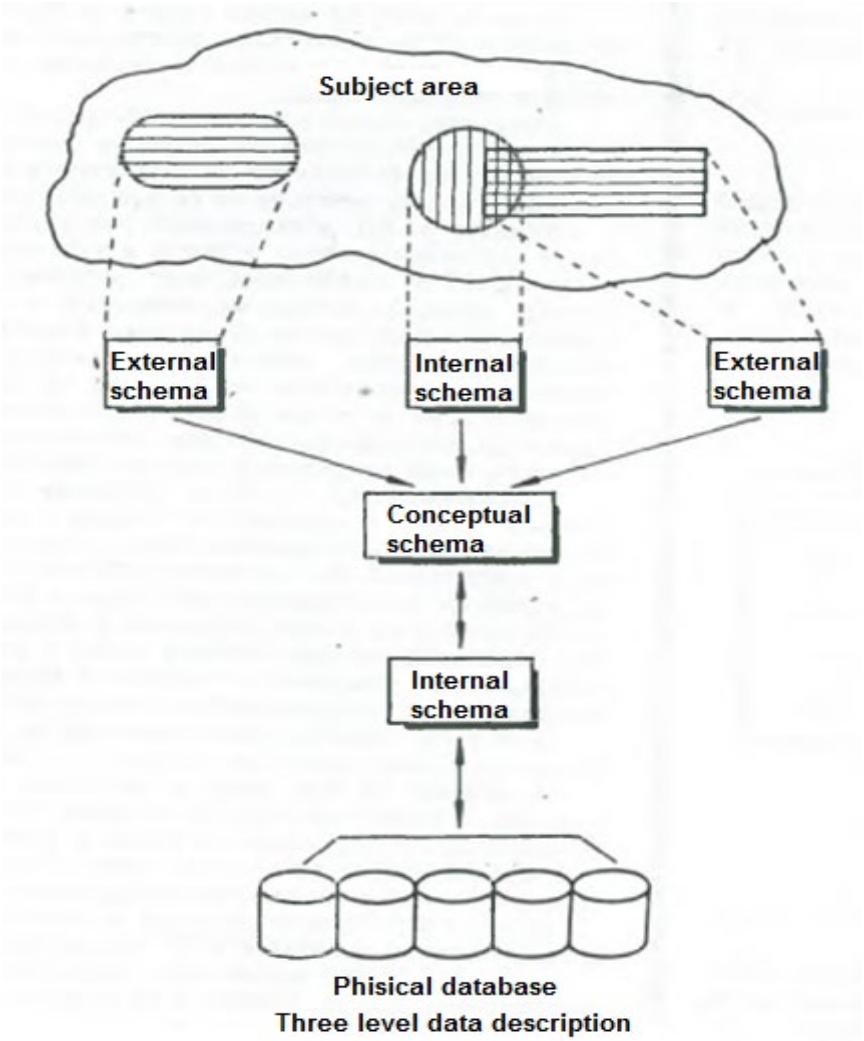
Co-funded by the Erasmus+ Programme of the European Union

Veri tanımlama dili

- Veritabanı şemasında tanımlanan çeşitli özniteliklerin değerleri üzerindeki bu tür mantıksal koşullara bütünlük kısıtlamaları olarak atıfta bulunmak gelenekseldir.
- Genel olarak bütünlük kısıtlamaları, bir veritabanında bulunan verilerde çelişkisizliğin korunmasına yönelik bir dizi kuraldır.
- Bütünlük kısıtlamaları hem veri değerlerine hem de bunlar arasında var olan ilişkilere kısıtlamalar getirir.

Co-funded by the Erasmus+ Programme of the European Union

Veri tanımlama dili



Three level data description

Data manipulation languages

- Çoğu zaman, MD'de tanımlanan işlemlere **veri işleme dili** denir. **Veri işleme dili (DML)** terimi de bazı literatürde kullanılmıştır.
- DML, PL/1, PASCAL, FORTRAN, ASEMBLER gibi belirli bir programlama dilinin uzantısı olabilir, bu durumda buna veri alt dili denir.
- Tamamen kendi kendine yeten ve diğer dillerden bağımsız olan çok sayıda DML de vardır. Bunlar, örneğin sorgu dilleri olarak adlandırılanlardır.

Veri işleme dilleri

Generic DDL Statements	
Statement	*Explanation*
Create-Database	Creates a database. An Oracle database is a complex object that stores various other database objects and resources.
Alter-Database	Modifies the database.
Drop-Database	Removes the database from the system catalog.
Create-Tablespace:	Creates a tablespace. A tablespace is a logical container within a database. It can contain several datafiles that contain the actual database objects.
Alter-Tablespace	Modifies a tablespace
Drop-Tablespace	Removes a tablespace from the system catalog.
Create-Table	Creates a database table. The table may be a relational table, an object table, or an XML table; the default is relational.
Alter-Table	Alters the physical and/or logical structure of a table.
Drop-Table	Removes a table from the system catalog. The table and all its data are deleted. Due to referential integrity, the typical DBMS forbids you to delete a table that contains referenced tuples.

Yaygın Kullanılan DDL İfadeleri

Veri işleme dilleri

- Tipik olarak, DML işlemlerinin kapsamı veritabanının bir kısmıdır ve bu, veri seçimiyle gerçekleştirilir.
- Dolayısıyla bu durumda DML işlemleri, bir dizi veriyi ve bu veriler üzerinde izin verilen eylemleri içerir.
- Örneğin operasyonla:

```
select  EMPLOYEE - NAME , EMPLOYEE - NUMBER
from    PERSONNEL
where CITY  =  '  VARNA  '
```

- PERSONEL dosyasında bilgileri saklanan Varna şehrinden çalışanların isim ve numaraları seçilecektir.

Veri işleme dilleri

- Veritabanı verileriyle yapılan ana eylemler iki türdendir. Bunlardan biriyle verilerin fiili işlenmesi şu şekilde gerçekleştirilir:

- *İfadelerin değerlerinin hesaplanması (içlerindeki argümanlar DB'den gelen sayılardır)*
- *DB'den seçilen sayıların maksimum, minimum, ortalama veya toplamını bulma.*

- Diğer eylem türü DB'nin durumunda değişiklik yapar. Bu tür eylemler şunlardır:

- DB'ye yeni veriler ekleme;
- DB'den tarihin kaldırılması;
- DB'deki verileri değiştirme.

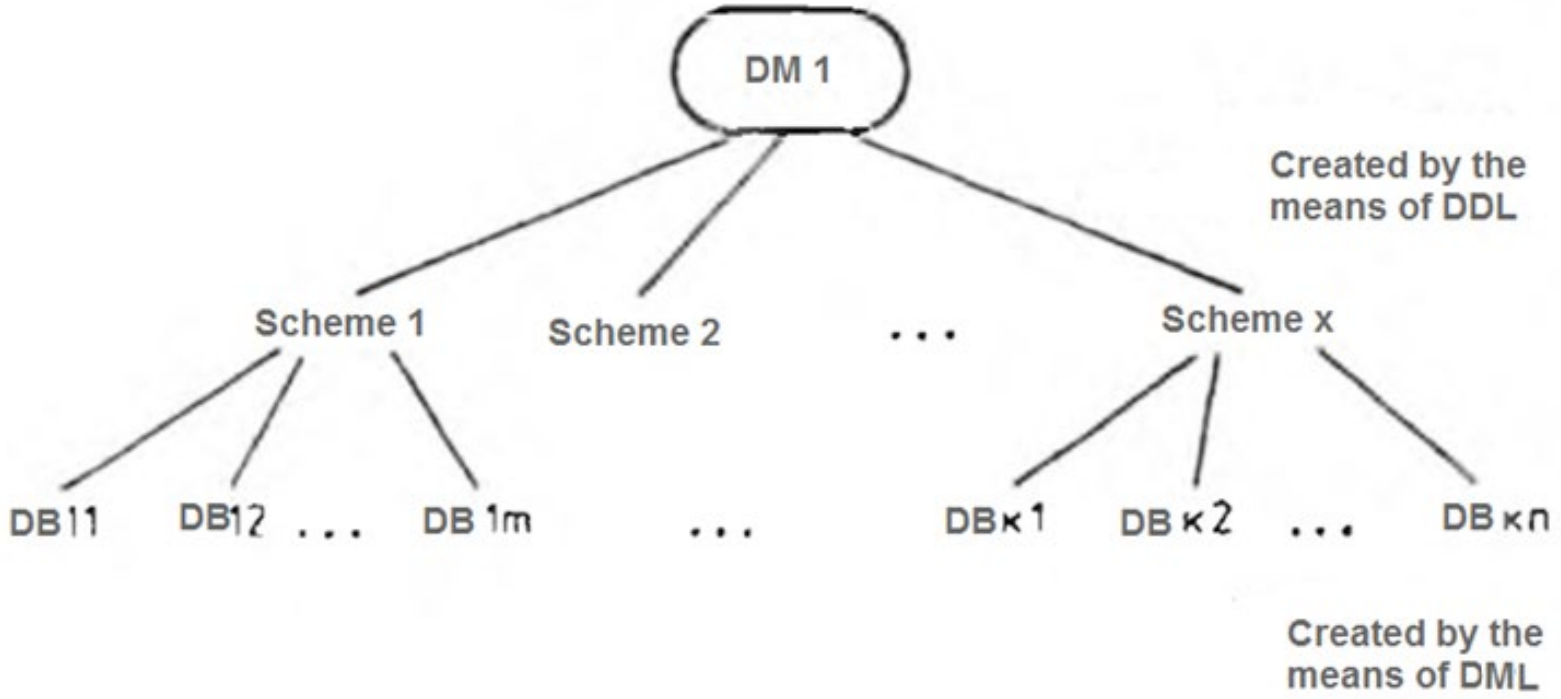
Co-funded by the Erasmus+ Programme of the European Union

Veri işleme dilleri

- İkinci türden eylemleri içeren herhangi bir işlem, DB'yi eski durumu gibi şemasına karşılık gelmesi gereken yeni bir duruma getirir.
- Bu, DML işlemlerinin gereksinimlerinden birinin, DML tarafından tanımlanan verilerin yapısal özelliklerini bozmaması olduğu anlamına gelir; e. Veritabanında saklanan verilerin şemada verilen gereksinimlere uygun olup olmadığı
- Her DM'nin DML aracılığıyla çok sayıda şema oluşturma olanağı sağladığı açıkça ortaya çıkıyor:

Co-funded by the Erasmus+ Programme of the European Union

Veri işleme ve veri tanımlama dilleri



Generating multiple schemes of the same model

Co-funded by the Erasmus+ Programme of the European Union

Veri işleme ve veri tanımlama dilleri

- Bu tür şemaların her birinden, DML işlemlerini kullanarak, şemada açıklanan bütünlük kısıtlamaları dahilinde içeriğini değiştirebilen bir DB oluşturulur.
- Böylece DM1 veri modeliyle Şema 1 oluşturulur.
- DML'deki işlemler, daha sonra durumunu değiştiren ve sürekli olarak DB12'yi elde eden DB11 veritabanını oluşturdu. . . , DB1m.

Co-funded by the Erasmus+ Programme of the European Union

Grafik veri modelleri

Veri tanımlama dili kavramı ve veritabanı modeliyle ilgili adı; burada veritabanı şeması, kullanıcı veri modelinin kayıtlarını, alanlarını ve kümelerini açıklayan bir dil sözdiziminde yazılmıştır. DDL bir veri modelinden oluşturulabilir.

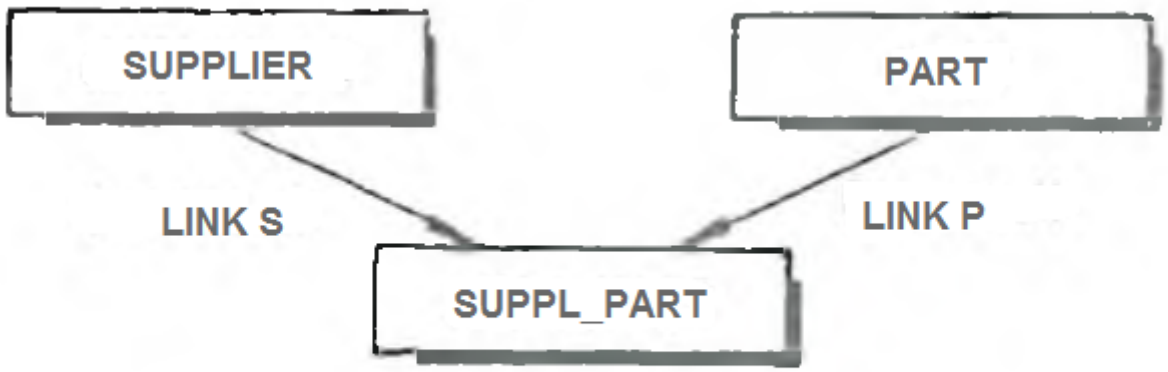
- Grafik veri modelleri ilk kez veritabanı uygulamalarında tanıtıldı ve kullanıldı.
- Bunlar, nesne sınıfları arasındaki ilişkilerin grafiklerle temsil edildiği modellerdir.
- Grafik modellerinin tipik temsilcileri **ağ** ve **hiyerarşik** veri modelleridir.
- Grafik veri modelleri aynı zamanda **"Varlık - İlişki"** veri modelini de ifade eder. Literatürde Varlık İlişkisi anlamına gelen ER veri modeli olarak bilinmektedir.

AĞ VERİ MODELLERİ

- Genel olarak ağ modelleri, verileri ve aralarındaki bağlantıları, köşeleri kayıt türleri ve yayları bağlantı türleri olan bir grafikle temsil etme olanağı sunar.
- Bir grafiğin grafiksel temsiline aynı zamanda veri yapısı grafiği de denir. Aşağıdaki şekil "SUPPLIES" veritabanının yapı şemasını göstermektedir. Üç tür kayıt ve iki tür bağlantı vardır.

Co-funded by the Erasmus+ Programme of the European Union

AĞ VERİ MODELLERİ



STRUCTURAL CHART OF "SUPPLIES" DATABASE

R1 ve R2 iki kayıt tipi olsun. R1 ve R2 arasındaki L12 bağlantı türü, R1 ve R2'nin öğeleri (kayıtları) arasındaki yazışmayı ayarlar. Bu, R1'in her r1 kaydı için, R2'den r1'e karşılık gelen bir kayıt kümesinin L12 bağlantısıyla belirlenebileceği anlamına gelir.

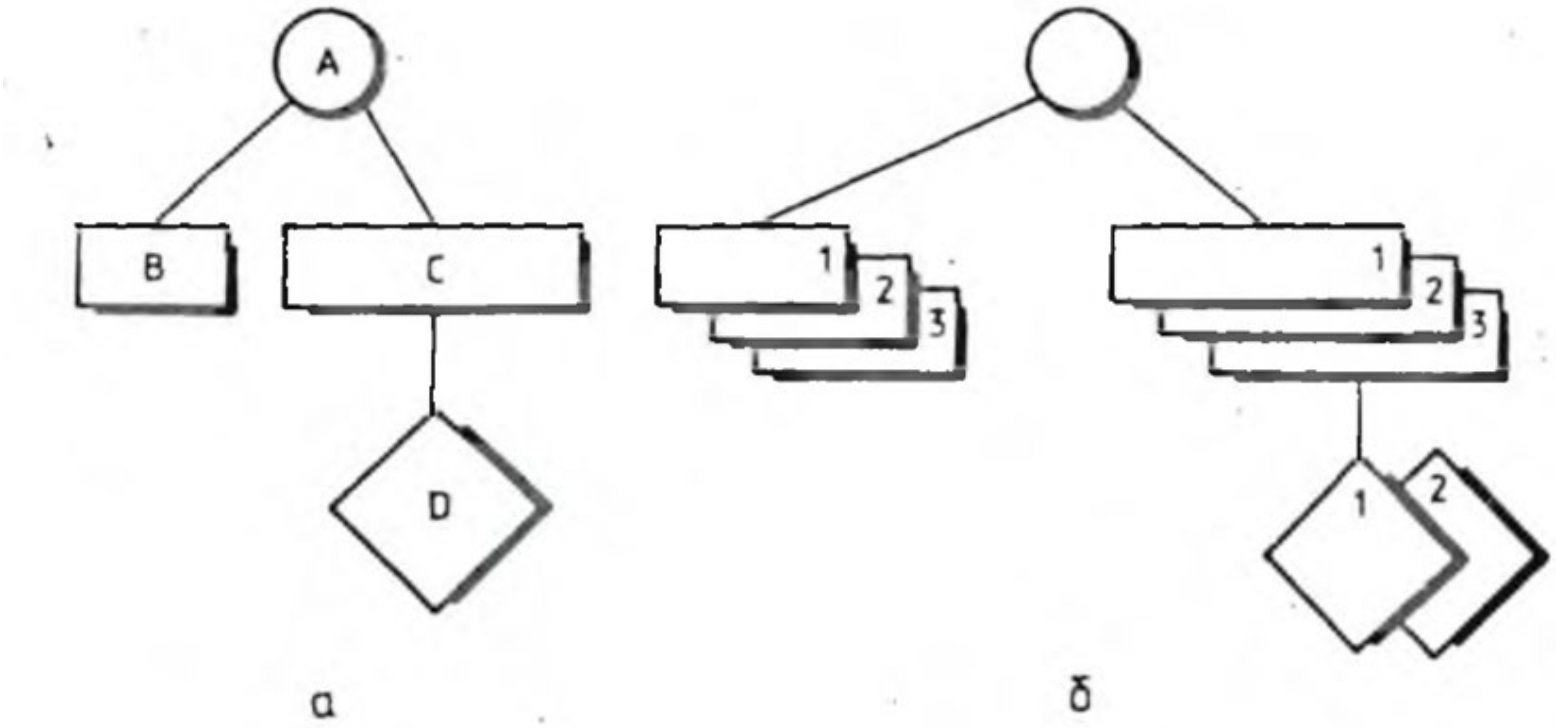
HİYERARŞİK VERİ MODELLERİ

- Hem ağ hem de hiyerarşik veri modelleri grafik modelleridir. Ancak hiyerarşik modellerde nesne sınıfları arasındaki ilişkilere daha güçlü kısıtlamalar uygulanır:
- 1. Bağlantılar işlevseldir ve isimlendirilmez.
- 2. Bir DB'nin yapı diyagramı iyi düzenlenmiş bir ağaçtır, i. e. bireysel alt ağaçların sırasının ilgili olduğu bir ağaç.
- Tanım ağacı olarak da adlandırılan yapı diyagramındaki her düğüm bir kayıt türüdür.
- Hiyerarşik veritabanı, tanım ağacının örneklerinin bir koleksiyonudur.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

HİYERARŞİK VERİ MODELLERİ

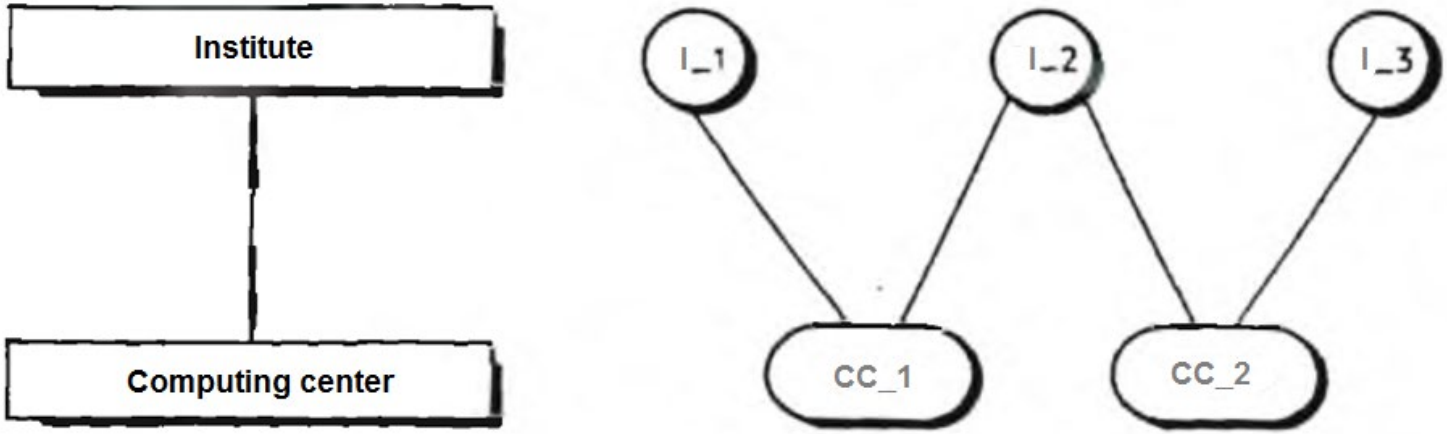


Hierarchical database
а - Definition tree
б - A sample of the definition tree

HİYERARŞİK VERİ MODELLERİ

- *Ağ yapısı diyagramları ağaçlarla nasıl temsil edilir?*
- Aşağıdaki şekildeki yapı şeması, bir enstitünün birden fazla bilgi işlem merkezinin bilgi işlem olanaklarıyla hizmet verebileceğini ve bir bilgi işlem merkezinin birden fazla enstitüye hizmet verebileceğini ifade etmektedir; yani, Enstitü (I) ve Bilgi İşlem Merkezi (CC) nesne sınıfları arasında M:N formunda bir ilişki kurulur.
- Hiyerarşik modelde böyle bir ilişki desteklenmediğinden aşağıdaki yaklaşım izlenebilir:

HİYERARŞİK VERİ MODELLERİ



Structure diagrams two classes of objects

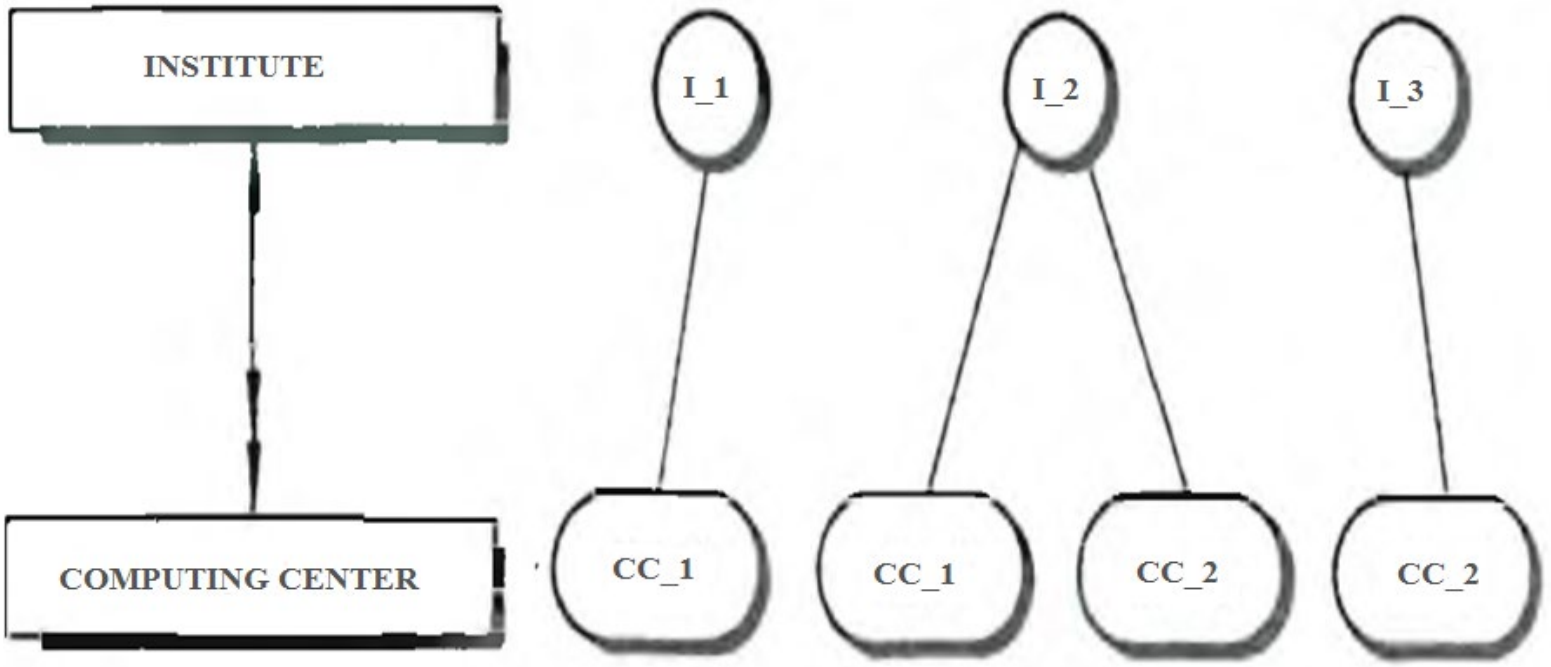
Sınıflardan birinin temel, diğerinin ise ikincil olduğu varsayılır. Aşağıdaki şeklin varsayımları altında temel sınıf "Enstitü"dür ve şuna benzer bir soru vardır:

- I-2 enstitüsüne hangi bilgi işlem merkezleri hizmet veriyor? kolaylıkla cevaplanabilmektedir. Ancak tam tersi soruyu yanıtlamak daha zordur:

- CC-2 bilgi işlem merkezi hangi kurumlara hizmet vermektedir?

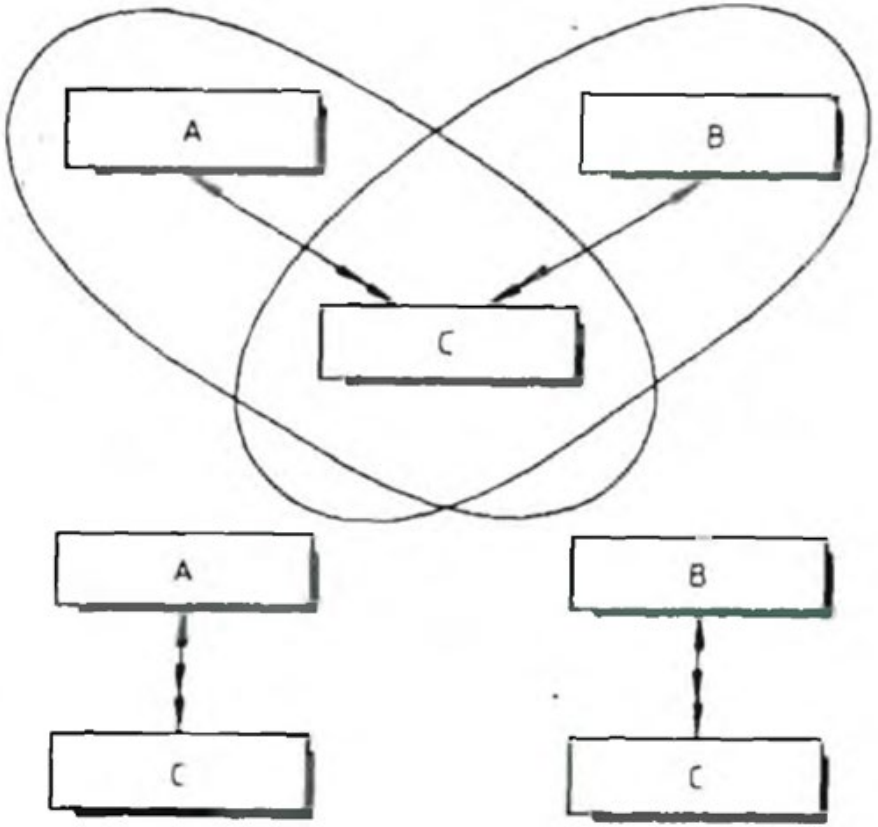
Co-funded by the Erasmus+ Programme of the European Union

HİYERARŞİK VERİ MODELLERİ



Definition tree with a basic class "Institute"

HİYERARŞİK VERİ MODELLERİ

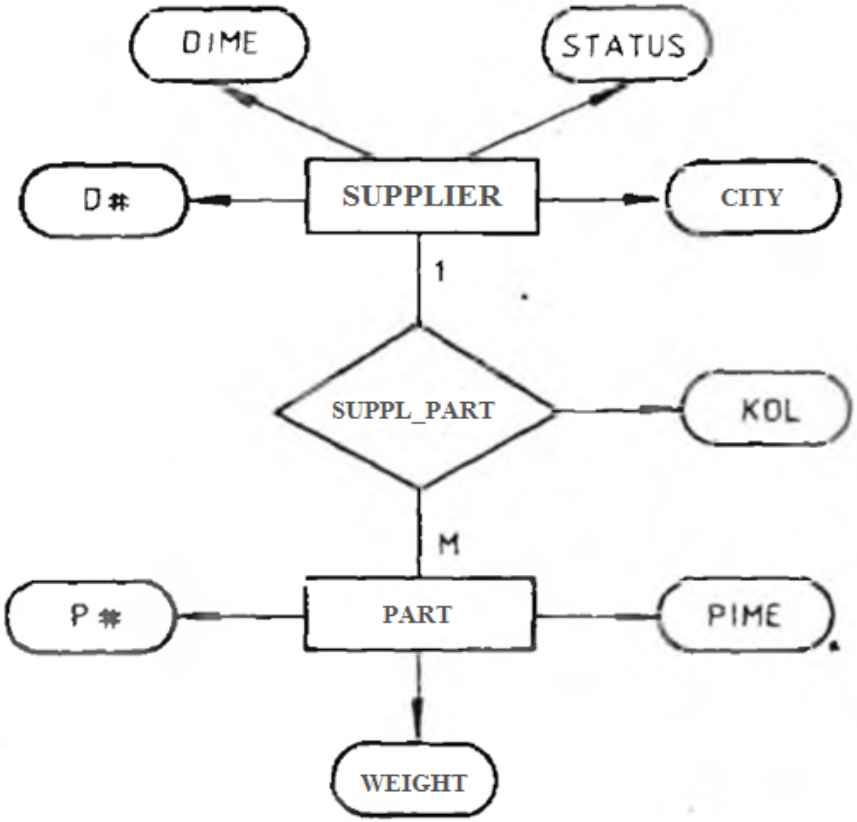


Representation of a network structure diagram as a set of two hierarchies

"Varlık – İlişki" Veri modelleri

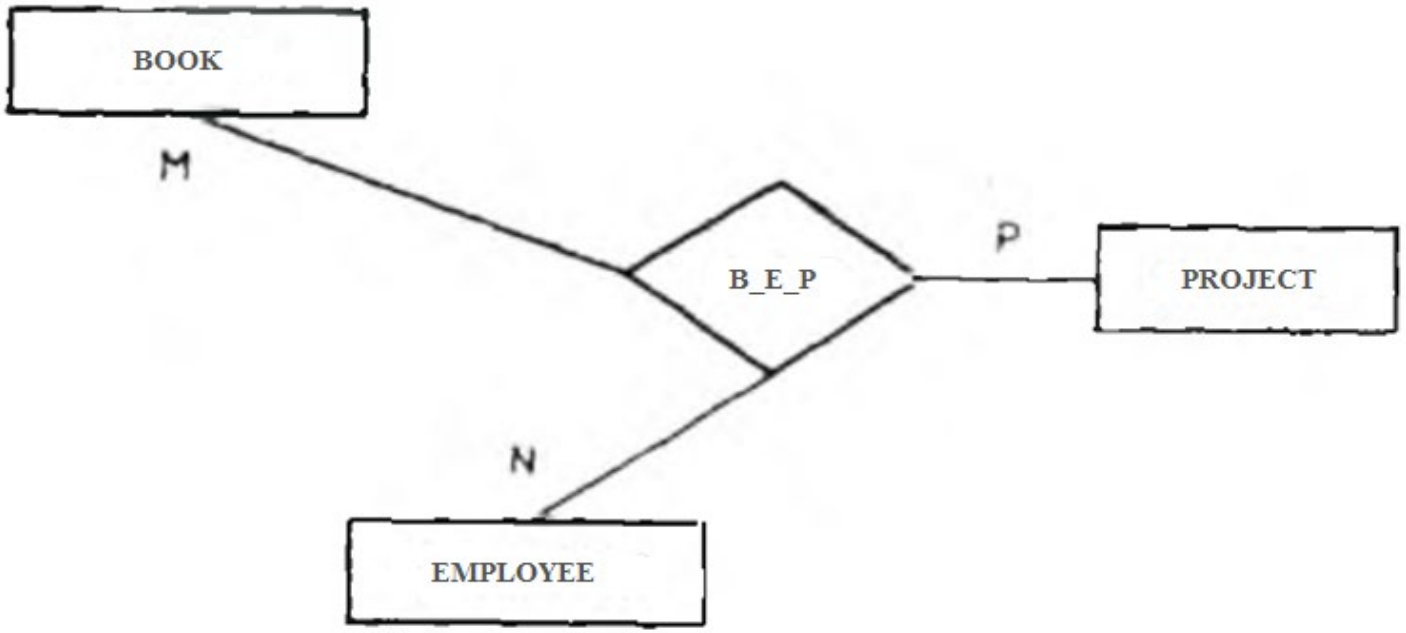
- Varlık-İlişki (ER) veri modeli bir bakıma yukarıda sunulan ağ ve hiyerarşik veri modellerinin özetidir.
- Modelin temel özellikleri ifade gücü ve karmaşık konu alanlarının tanımlanabileceği genelliktir.
- ER modelindeki ana yapılar varlık kümeleri (varlık türü) ve ilişki kümeleridir (ilişki türü).
- "Varlık" kümeleri nesne sınıflarının yapısını, "ilişki" kümeleri ise bu nesne sınıfları arasındaki ilişkilerin yapısını temsil eder.

"Varlık – İlişki" Veri modelleri



ER-DIAGRAM FOR "SUPPLIES" DB

“Varlık – İlişki” Veri modelleri



Realtionships of M:N type between some classes

"Varlık – İlişki" Veri modelleri

- ER modeli aracılığıyla, bir DB'nin yapısı ER diyagramları adı verilen diyagramlarla temsil edilir.
- Bu diyagramlar dikdörtgen, eşkenar dörtgen ve oval olmak üzere üç tür geometrik şekil kullanılarak oluşturulmuştur.
- Nesne sınıfları dikdörtgenlerle, nitelikleri ise karşılık gelen dikdörtgenlere bağlı ayrı ovallerle temsil edilir.
- Nesne sınıfları arasındaki ilişkiler, niteliklere de sahip olabilen eşkenar dörtgen ile temsil edilir.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. DBMS'de bulunan farklı dilleri açıklayın?
2. DBMS'de DDL ve DML arasındaki fark nedir?
3. Bir DBMS'deki farklı veri soyutlama düzeylerini açıklayın.

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

❑ Modül 1. Veritabanlarının Teorik Temelleri

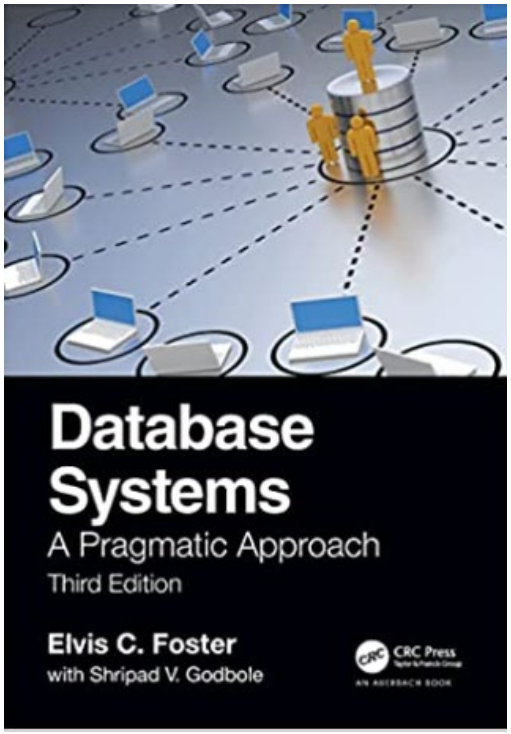
❑ Konu 4. Temel arama yöntemleri. Dosya türleri.

❑ Ders 1. Temel arama yöntemleri.

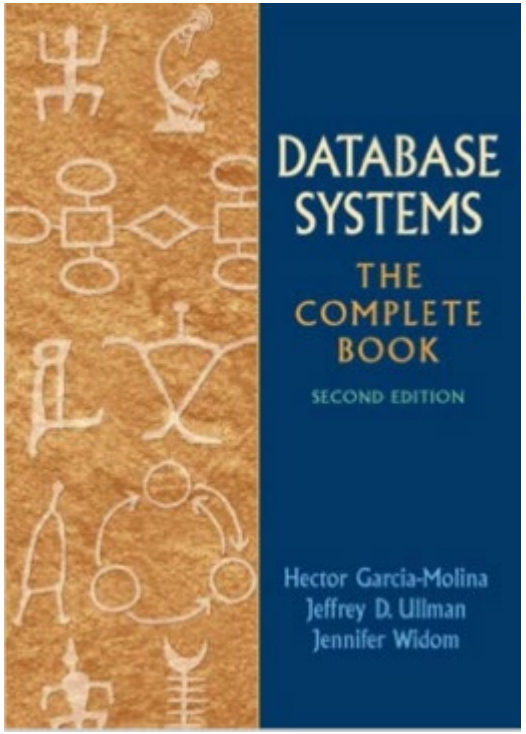
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

TEMEL ARAMA YÖNTEMLERİ



Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.



Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

Azalov, P. (1991) Database, Sofia.

Co-funded by the Erasmus+ Programme of the European Union

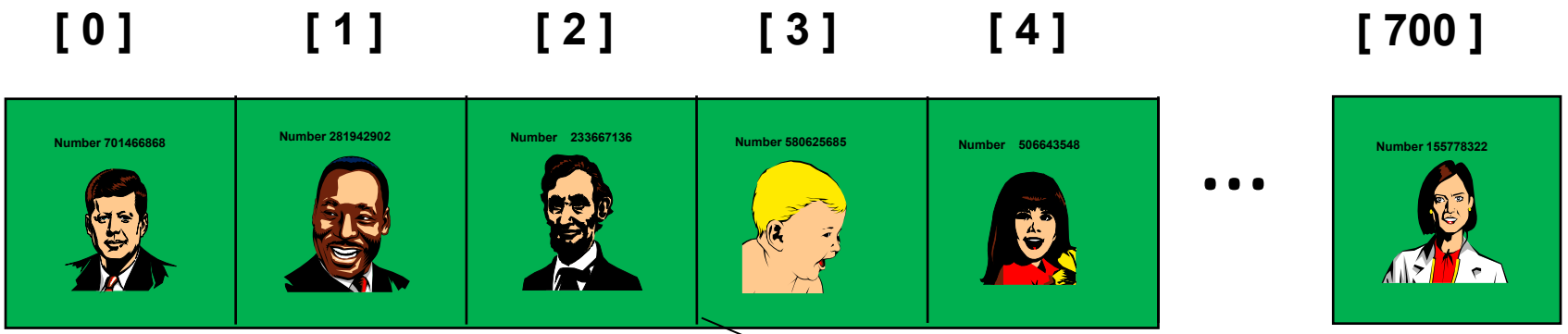
Sorun: Arama

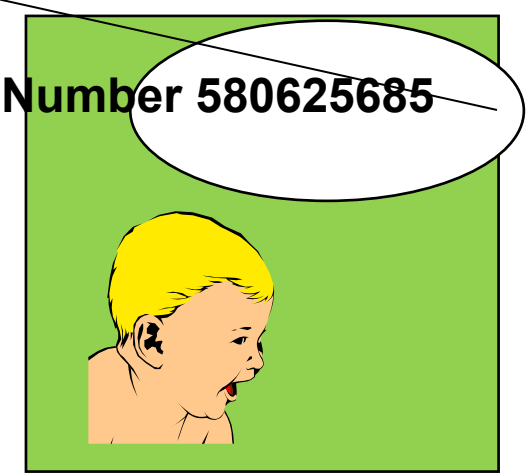
- Bize bir kayıt listesi veriliyor.
- Her kaydın ilişkili bir anahtarı vardır.
- Belirli bir anahtarı içeren bir kaydı aramak için etkili bir algoritma verin.
- Verimlilik, bir öğeyi almak için ortalama süre analizi (karşılaştırma sayısı) cinsinden ölçülür.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Arama





Listedeki her kaydın ilişkili bir anahtarı vardır.
Bu örnekte anahtarlar kimlik numaralarıdır.

Belirli bir anahtar verildiğinde, kaydı listeden verimli bir şekilde nasıl alabiliriz?

TEMEL ARAMA YÖNTEMLERİ

- Önceki derslerde de belirtildiği gibi veri organizasyonundaki asıl görev, verilen bir anahtara göre o anahtarın bulunduğu kaydın konumunu belirlemektir.
- Bu bölümde, bireysel yöntemlerin arama hızına özellikle dikkat edilerek, bu göreve bazı klasik çözümler ele alınmaktadır.

SIRALI ARAMA (Doğrusal Arama)

F'nin n kayıt içeren bir dosya olduğunu, n >= 1 olduğunu varsayıyoruz. Eğer F'nin kayıtları anahtar değerlerine göre sıralanmıyorsa, yapılacak en doğal şey belirli bir kaydı aramak, dosyada sırayla ilerlemektir. bir uçtan diğer uca kayıt ederek.

TEMEL ARAMA YÖNTEMLERİ

- Aranmakta olan her kaydın olasılığının aynı olduğunu önerin, i. e. Her i=1,2 için P = 1/n. . . ,n, dosyadan bir kayıt bulmak için gereken toplam ortalama kontrol sayısı:

$$E = \sum_{i=1}^{n} iP_i = \sum_{i=1}^{n} i(1/n) = (1/n) \sum_{i=1}^{n} i = (n + 1)/2$$

- Açıkçası, n'nin büyük değerleri için E tahmini çok büyük olacak ve belirli bir kaydın aranması yavaş olacaktır.
- Bu nedenle sıralı arama yalnızca dosya sırasızsa ve/veya dosya kayıtlarının tamamının veya çoğunun gözden geçirilmesi ve işlenmesi gerekiyorsa uygulanır.

TEMEL ARAMA YÖNTEMLERİ

➢ *ENGEL ARAMA*

- Dosyanın anahtar değerlerine göre sıralanması durumunda blok araması gerçekleştirilebilir.
- Yine F dosyasının n kayda sahip olduğunu ve kayıtlarının her blokta x kaydından oluşan ayrı gruplar (bloklar) halinde koşullu olarak gruplandırıldığını varsayıyoruz.
- Son blokta x'ten daha az sayıda giriş bulunabilir.
- Bu varsayımlara göre tüm blokların sayısı [n/x] olacaktır.
- [a] işareti a'dan küçük olmayan en küçük tam sayıyı gösterir.

TEMEL ARAMA YÖNTEMLERİ

➢ *ENGEL ARAMA*

- Kayıtları ardı ardına incelemek yerine, her bloktaki yalnızca en yeni kayıtları incelemek elbette daha iyidir. Daha sonra incelenen blokların ortalama miktarı şöyledir:

$$M_1 = (\lceil n/x \rceil + 1)/2$$

- Böyle bir blok araması, muhtemelen aranmakta olan kaydı içeren bir blok bulursa, o blokta, ortalama olarak M2 kontrollerinin yapılacağı x'ten fazla kontrol olmadan sıralı bir arama gerçekleştirilebilir:

$$M_2 = \sum_{i=1}^{x-1} i(1/x) = (1/x) \sum_{i=1}^{x-1} i = (x - 1)/2$$

TEMEL ARAMA YÖNTEMLERİ

➢ *ENGEL ARAMA*

- Böylece blok aramadaki toplam ortalama kontrol sayısı E şöyle olur: $$E = M_1 + M_2 = (\lceil n/x \rceil + 1)/2 + (x - 1)/2 = (\lceil n/x \rceil + x)/2.$$

- Bu tahmin, bir bloktaki kayıtların x sayısına bağlıdır. E'nin minimum değere sahip olduğu x değerini belirleyerek bu sayıyı en aza indirebiliriz: $$E'(x) \approx (-n/x^2 + 1)/2 = 0$$

- Bu nedenle belirliyoruz, $x \approx \sqrt{n}$, ve E'nin minimum değeri için şunu elde ederiz: $$E(\sqrt{n}) = (n/\sqrt{n} + \sqrt{n})/2 = \sqrt{n}$$

Doğrusal arama özeti

- Kayıt dizisinde birer birer adım atın.
- Eşleşen anahtara sahip kaydı arayın.
- Arama ne zaman durur?

-eşleşen anahtara sahip kayıt bulunduğunda

-veya arama tüm kayıtları başarıyla incelediğinde.

Doğrusal arama için sözde kod

```
// n dizi elemanında istenen öğeyi arayın
// a[first]'ten başlıyoruz.
// Bulunursa işaretçiyi istenen kayda döndürür.
// Aksi halde NULL değerini döndür
…
for(i = ilk; i < n; ++i )
if(a[ilk+i] istenen öğeyse)
return &a[ilk+i];

// eğer döngüden geçersek istenilen öğe bulunamadı
NULL'u döndür;
```

Doğrusal arama analizi

- Seri arama için en kötü ve ortalama çalışma süreleri nelerdir?
- Aramada gerekli işlem sayısı için O-notasyonunu belirlememiz gerekir.
- İşlem sayısı listedeki giriş sayısı olan n'ye bağlıdır.

Doğrusal arama için en kötü durum zamanı

- N öğeden oluşan bir dizi için, seri aramanın en kötü durum süresi, n dizi erişimini gerektirir: O(n).
- Tüm n kayıt üzerinde döngü yapmamız gereken durumları düşünün:

-İstenilen kayıt dizinin son konumunda görünür

-istenilen kayıt dizide hiç görünmüyor

Doğrusal arama için ortalama durum

Varsayımlar:

1. Bir aramada tüm anahtarların eşit olasılığa sahip olması
2. Her zaman dizideki bir anahtarı ararız

Örnek:

-10 kayıttan oluşan bir dizimiz var.

-İlk kaydı ararsanız 1 dizi erişimi gerekir; ikincisi ise 2 diziye erişim sağlanır. vesaire.

Tüm bu aramaların ortalaması:

(1+2+3+4+5+6+7+8+9+10)/10 = 5,5

Doğrusal arama için ortalama vaka süresi

Dizi boyutu n için genelleştirin.

Ortalama durum çalışma süresi için ifade:

$(1+2+\ldots+n)/n = n(n+1)/2n = (n+1)/2$

Bu nedenle seri arama için ortalama vaka süresi karmaşıklığı O(n)'dir.

TEMEL ARAMA YÖNTEMLERİ

➢ *IKİLİ ARAMA*

- İkili arama ve blok arama yalnızca dosya sıralandığında uygulanır.
- Eşitsizliklerin geçerli olduğu pozitif bir tam sayı k olsun:

$$2^{k-1} <= n < 2^k$$

- Dosyayı iki "eşit" parçaya bölüyoruz. Aranan kayıt ya birinci ya da ikinci yarıdadır ya da hiçbirinde değildir.
- Bu işlem, kayıt bulunana veya dosyada eksik olduğu tespit edilene kadar devam eder. Kontrol sayısına ilişkin tahmin şu şekildedir:

TEMEL ARAMA YÖNTEMLERİ

➢ *IKİLİ ARAMA*

- 0 dosya bölmesinde n kayıt vardır.
- 1 dosya bölünmesinde n/2 kayıt kalır.
- 2 dosya bölünmesinde n/(2^2) kayıt kalır.
- …

- *k-1* dosya bölünmesinde $n/(2^{k-1})$ kayıt kalır.
- Ama $1 <= n/(2^{k-1}) < 2$. Bu yüzden: $0 <= \log_2 n - \log_2 2^{k-1}$
- Sonunda şunu elde ederiz: $k_{max} = \lfloor \log_2 n \rfloor + 1$
- E ikili aramasının ortalama tahmınının turetılmesı üzerinde durmadan, yalnızca şunu belirteceğiz: $E = \log_2 n - 1$

TEMEL ARAMA YÖNTEMLERİ

➢ *IKİLİ ARAMA*

- Sıralı, blok ve ikili aramadaki ortalama puanların davranışına ilişkin bazı örnekleri burada bulabilirsiniz..

n	$\sqrt{n}$	$\log_2 n - 1$
16	4	3
256	16	7
1024	32	9
2^{2n}	2^n	$2n-1$

TEMEL ARAMA YÖNTEMLERİ

➢ *IKİLİ ARAMA*

- Blok ve ikili aramada daha hızlı aramanın maliyeti, dosya sıralamasını sürdürmek için gereken süre kadar ödenir. Bu nedenle bu yöntemler genellikle zayıf değişkenliğe sahip dosyalar için özellikle uygundur.
- Tartışılan blok ve ikili arama yöntemleri, tamamen RAM'e sığan dosyaları aramak için kullanılan tipik yöntemlerdir. Bu tür dosyalar sıklıkla indeks dosyalarındaki indekslerdir.

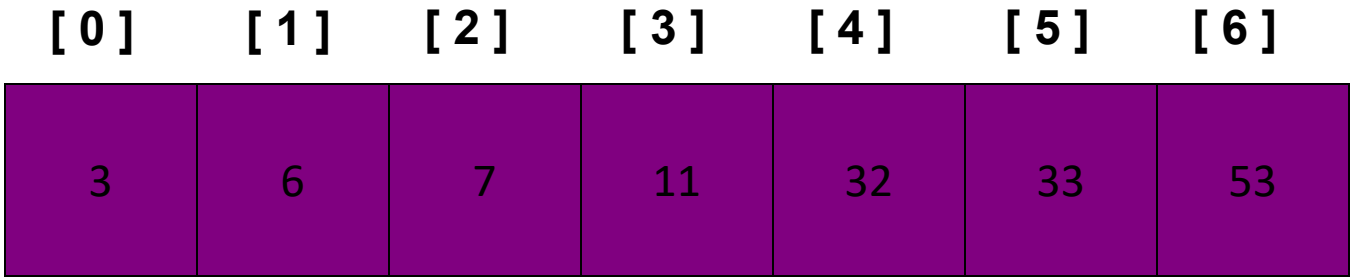
İkili Arama Sözde Kodu

```
…
if(boyut == 0)
bulunan = yanlış;
başka {
orta = dizi bölümünün yaklaşık orta noktasının dizini;
if(hedef == a[orta])
hedef bulundu!
else if(hedef < a[orta])
orta noktadan önceki alanda hedef arayın;
başka
orta noktadan sonraki alanda hedefi arayın;
}
…
```

Co-funded by the Erasmus+ Programme of the European Union

IKİLİ ARAMA

Örnek: tam sayı anahtarlarının sıralanmış dizisi. Hedef=7.

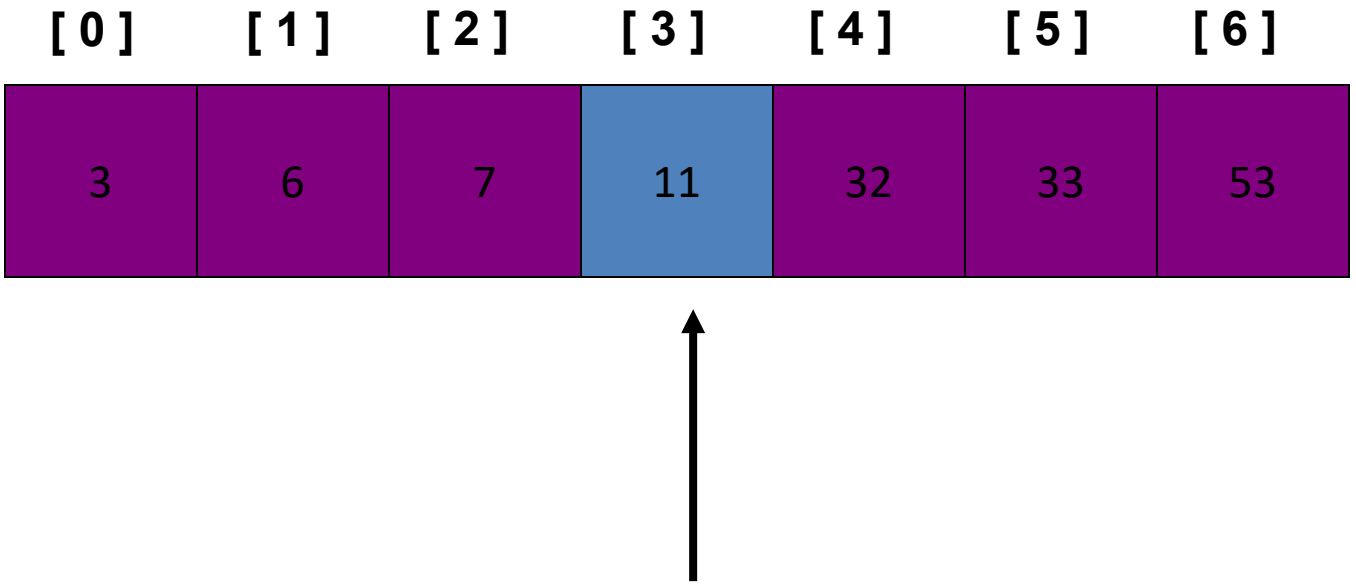


Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

IKİLİ ARAMA

Örnek: tam sayı anahtarlarının sıralanmış dizisi. Hedef=7.

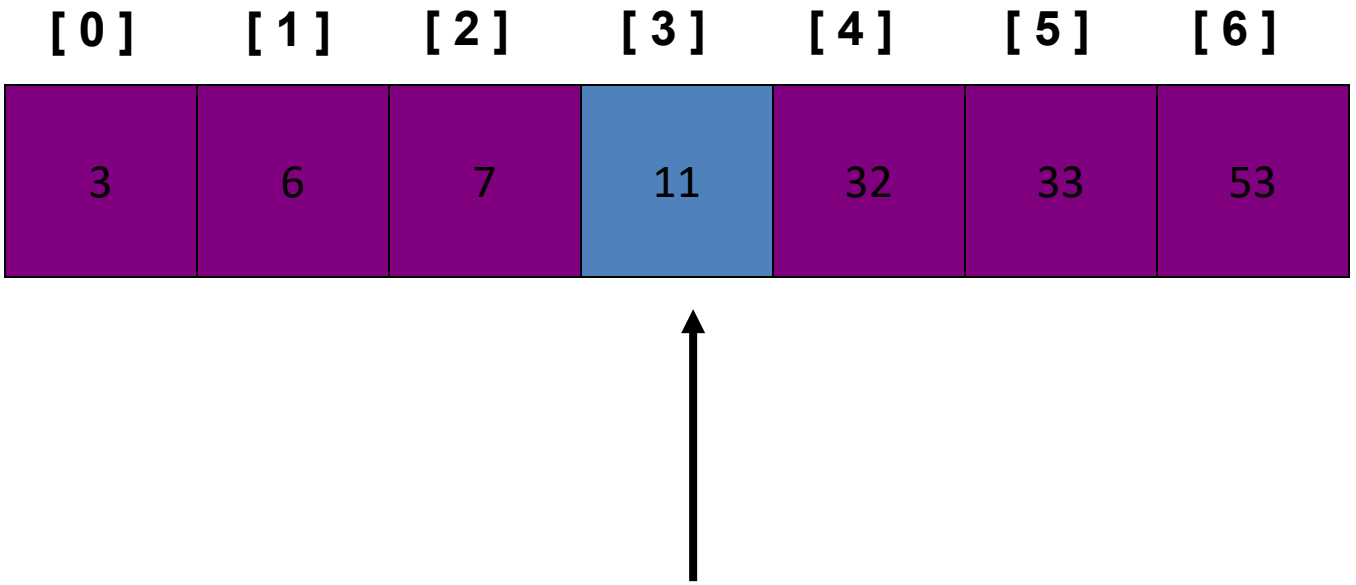


Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

IKİLİ ARAMA

Örnek: tam sayı anahtarlarının sıralanmış dizisi. Hedef=7.



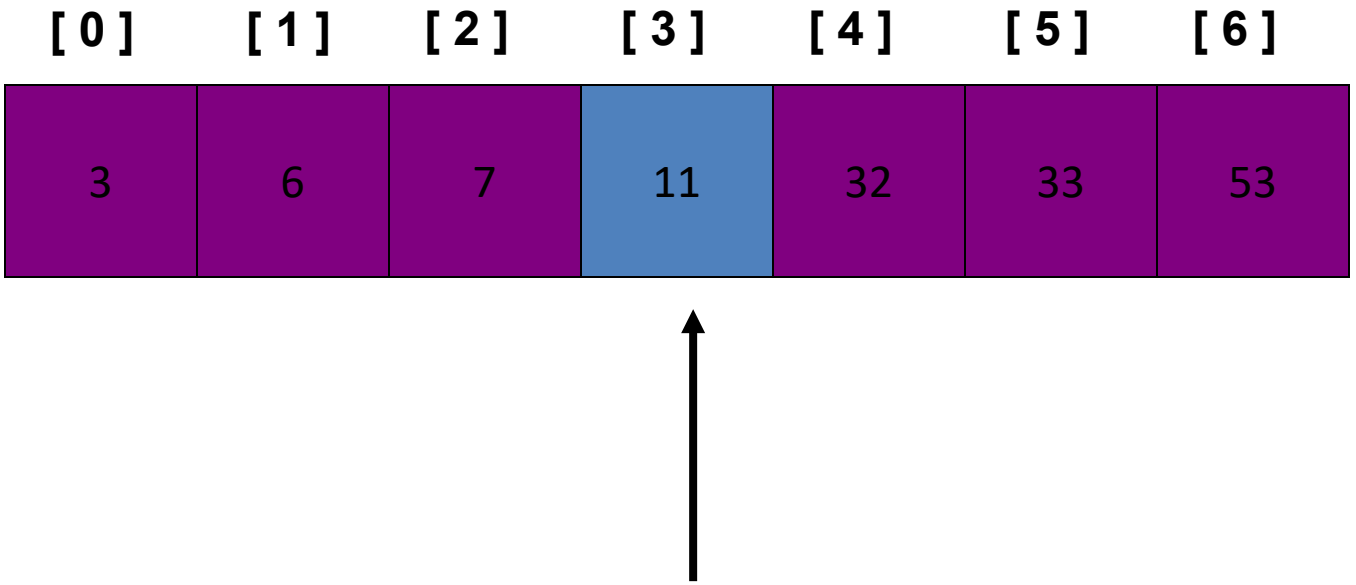
7 = orta nokta anahtarı mı? HAYIR.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

IKİLİ ARAMA

Örnek: tam sayı anahtarlarının sıralanmış dizisi. Hedef=7.

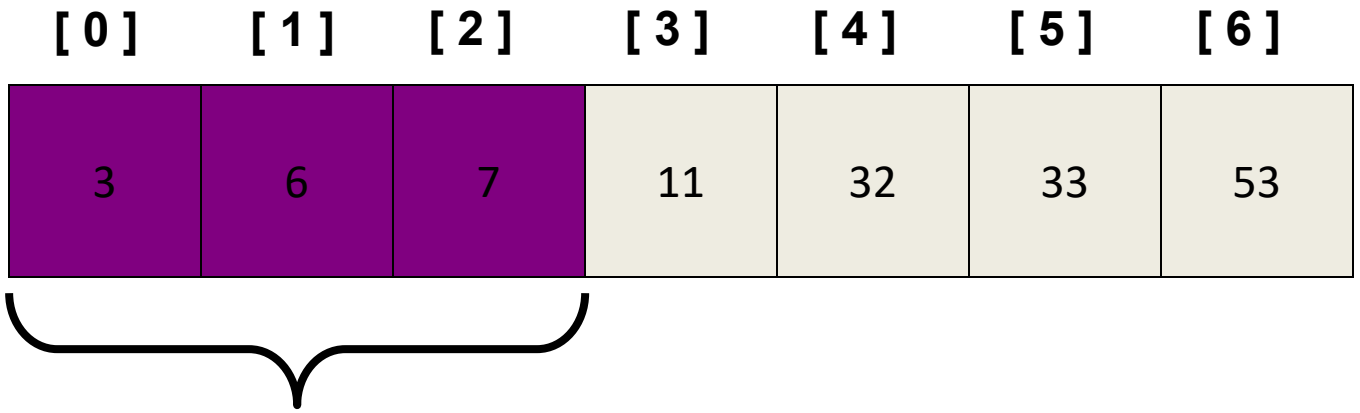


Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

IKİLİ ARAMA

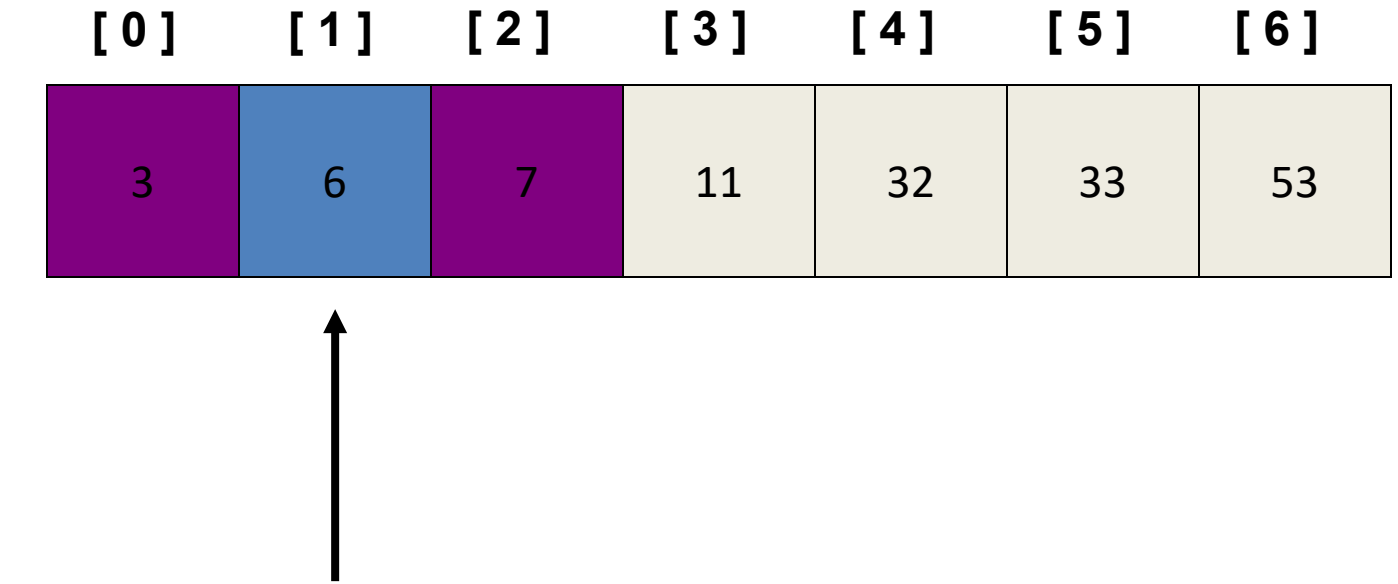
Örnek: tam sayı anahtarlarının sıralanmış dizisi. Hedef=7.



Hedefi orta noktadan önceki alanda arayın.

IKİLİ ARAMA

Örnek: tam sayı anahtarlarının sıralanmış dizisi. Hedef=7.

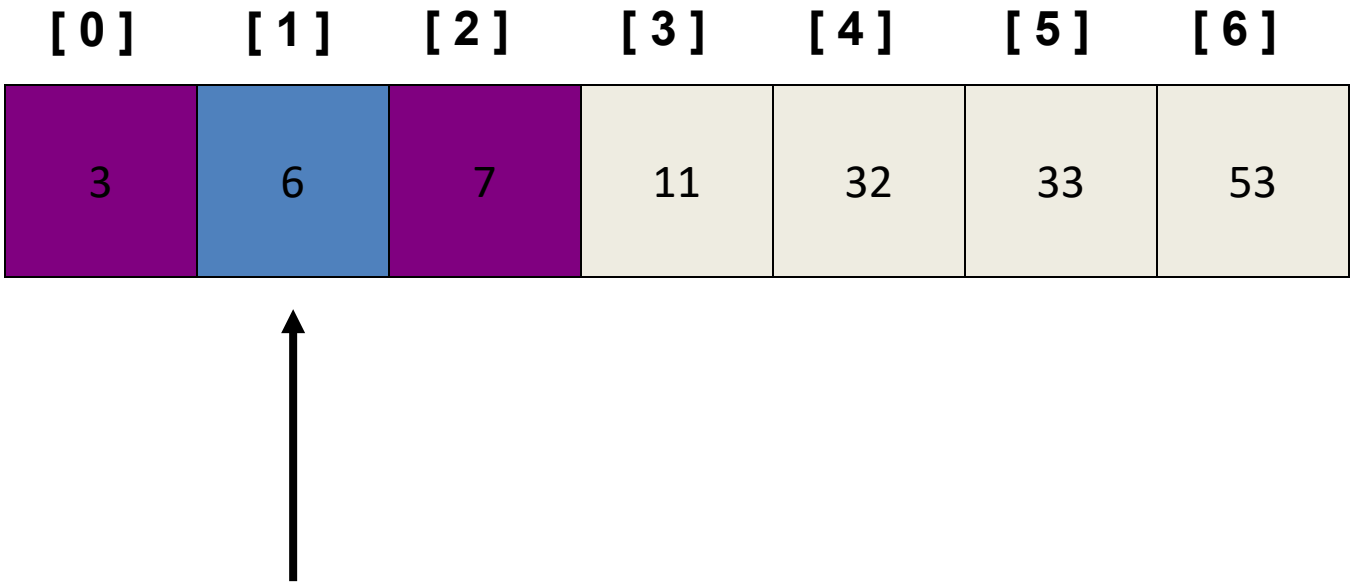


Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

IKİLİ ARAMA

Örnek: tam sayı anahtarlarının sıralanmış dizisi. Hedef=7.



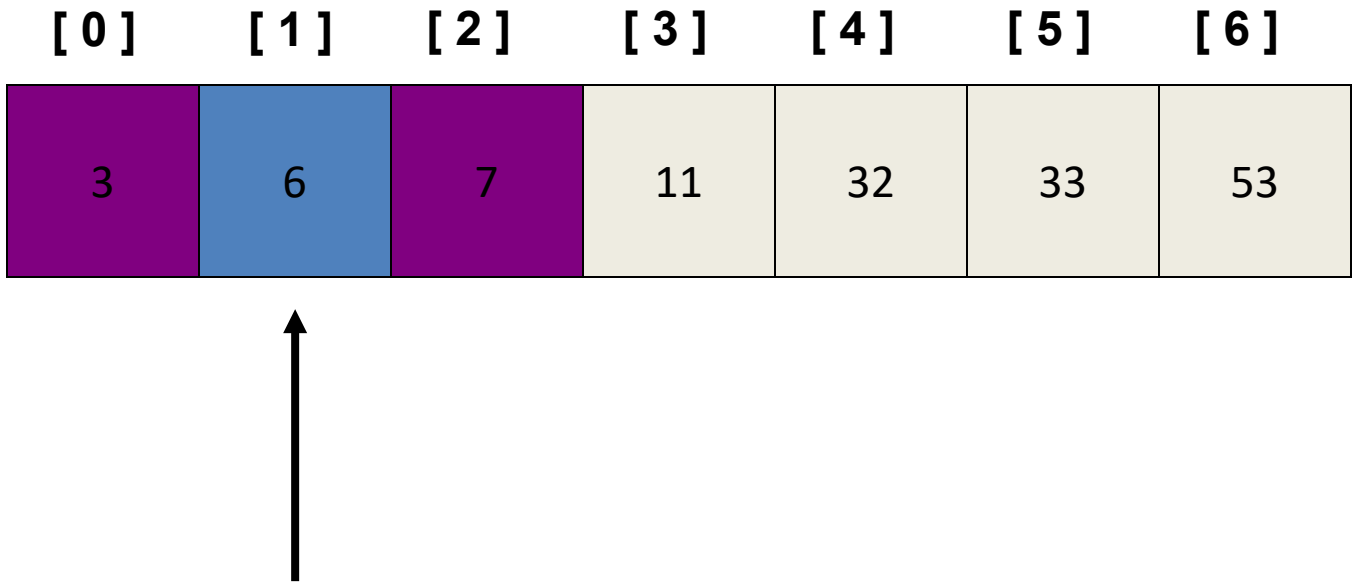
Hedef = orta noktanın anahtarı? HAYIR.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

IKİLİ ARAMA

Örnek: tam sayı anahtarlarının sıralanmış dizisi. Hedef=7.



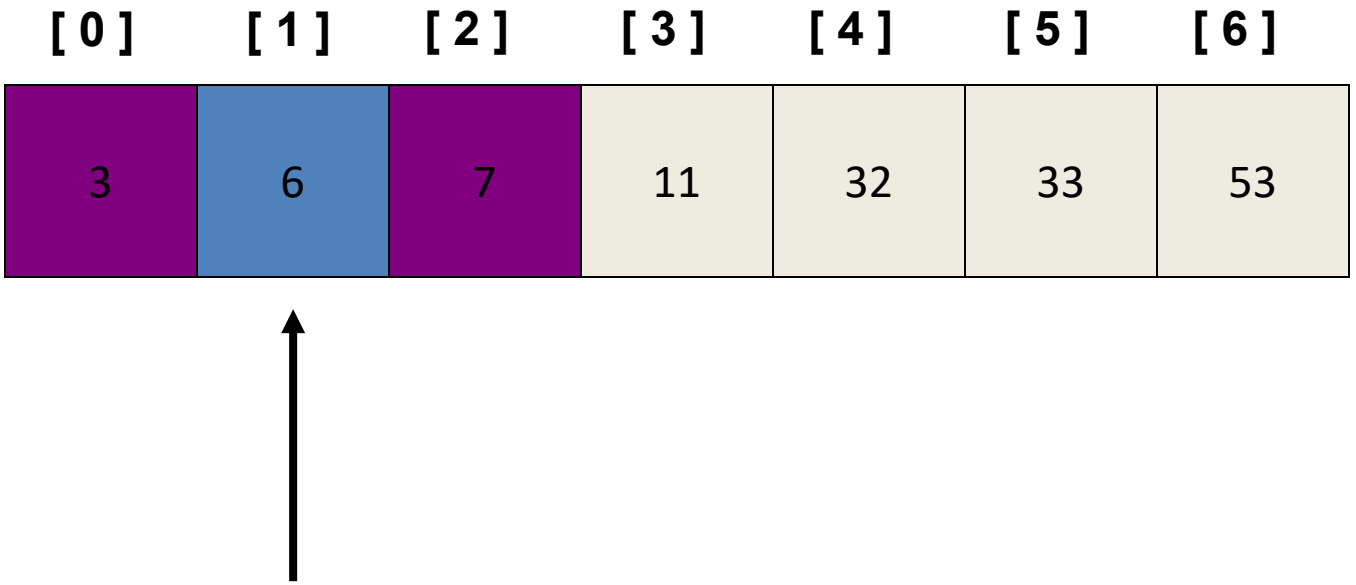
Hedef < orta noktanın anahtarı? HAYIR.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

IKİLİ ARAMA

Örnek: tam sayı anahtarlarının sıralanmış dizisi. Hedef=7.



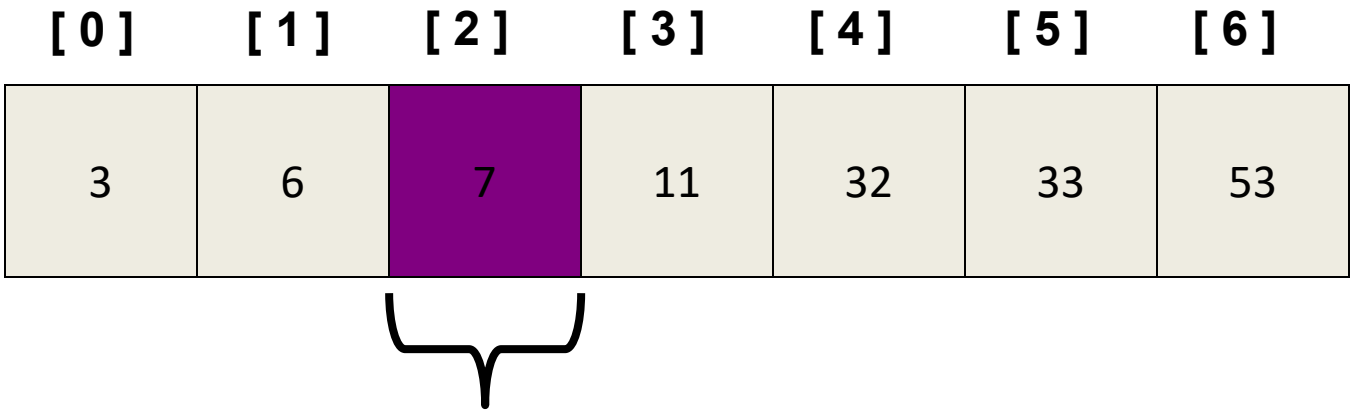
Hedef > orta noktanın anahtarı? EVET.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

IKİLİ ARAMA

Örnek: tam sayı anahtarlarının sıralanmış dizisi. Hedef=7.



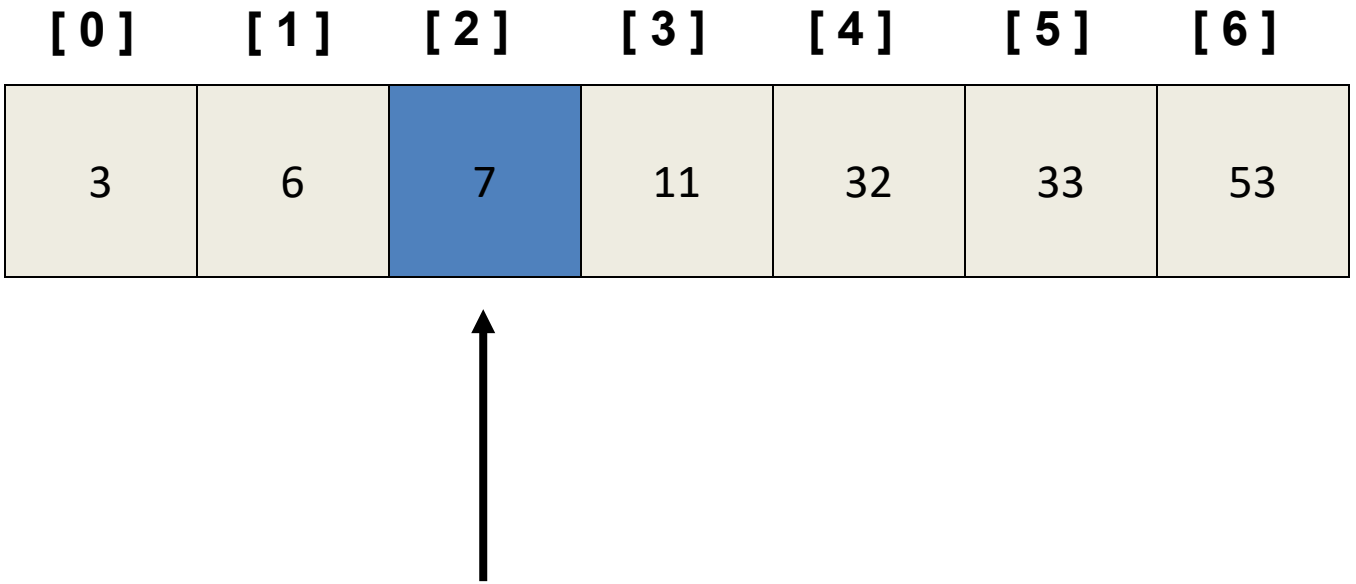
Hedefi orta noktadan sonraki alanda arayın.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

IKİLİ ARAMA

Örnek: tam sayı anahtarlarının sıralanmış dizisi. Hedef=7.



Yaklaşık orta noktayı bulun.
Hedef = orta nokta anahtarı mı? EVET.

İkili Arama Uygulaması

```
geçersiz arama(const int a[ ], önce size_t, size_t size, int hedef, bool& bulunan, size_t& konum)
{
size_t orta;
if(size == 0) bulunan = false;
başka {
orta = birinci + boyut/2;
if(hedef == a[orta]){
konum = orta;
bulunan = doğru;
}
else if (hedef < a[orta])
// hedef ortadan küçük olduğundan alt diziyi ortadan önce arayın
arama(a, ilk, boyut/2, hedef, bulunan, konum);
başka
// hedef ortadakinden büyük, bu nedenle alt diziyi ortadan sonra arayın
arama(a, orta+1, (boyut-1)/2, hedef, bulunan, konum);
}
}
```

TEMEL ARAMA YÖNTEMLERİ

BASIS FOR COMPARISON	LINEAR SEARCH	BINARY SEARCH
Time Complexity	The formula can be written as **O(N)**	**O(log 2 N)** is the formula that can be followed for this search
Sequential	Linear search is led by sequence; it starts from the first point and ends at the last point.	The binary search begins from the middle point.
The most compelling case time	The first Element serves to the most appropriate case time	Centre Element is the most relevant case time in this search.
Prerequisites needed for an array	Not needed	The array must be formed in sorted order
The worst-case scenario for n elements	N number of comparisons will be needed	The conclusion can be derived only after log2N comparisons
Capable of being implemented on	Linked lists and arrays	Incapable of being implemented directly on linked lists
Insert operation	Can be inserted with ease at the end of lists	Processing is needed to make insertions at their proper place and for the sake of maintaining sorted lists.
Algorithm type	Iterative characteristics depicted	Characteristics of **"divide and conquer"** features are described.
Utility	Easy to decipher and apply. There is no need for ordered elements.	The algorithms are tricky to understand and apply. The elements have to be organized in the proper manner and order.
Lines of Coding	Less	More

Sorular ve alıştırmalar:

1. Temel arama yöntemleri hangisidir?
2. Doğrusal arama nedir?
3. İkili arama nedir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

❑ Modül 1. Veritabanlarının Teorik Temelleri

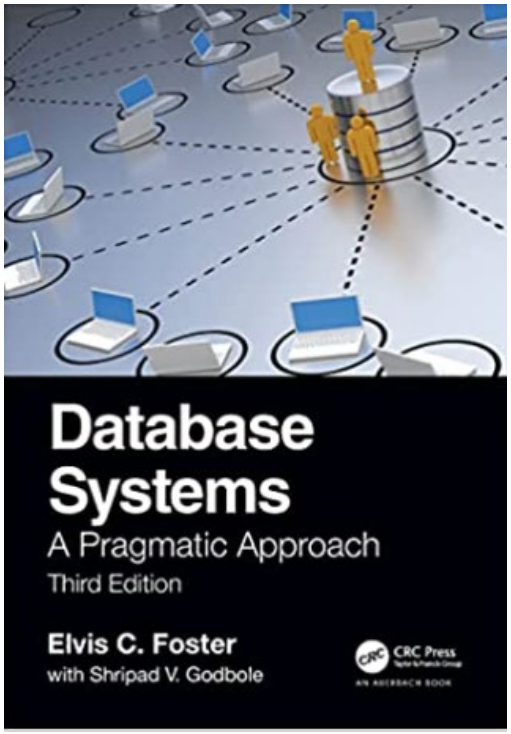
❑ Konu 4. Temel arama yöntemleri. Dosya türleri.

- ❑ Ders 2. Veritabanı dosyalarının türleri.

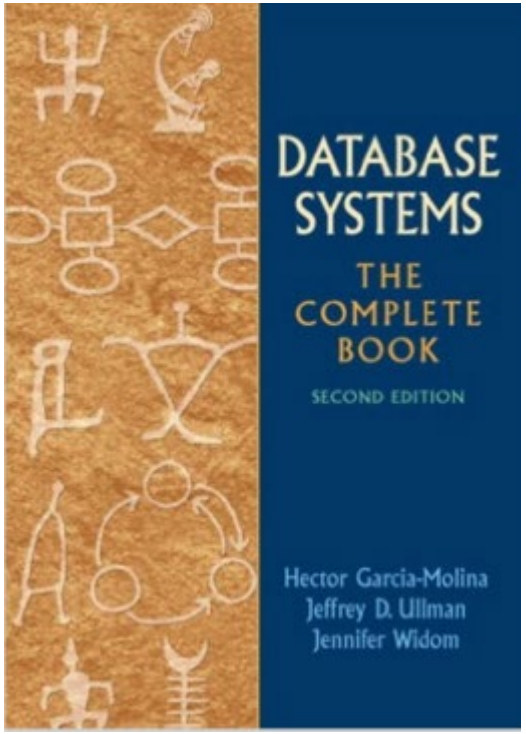
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANI DOSYA TÜRLERİ



Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.



Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

Azalov, P. (1991) Database, Sofia.

ENDEKS DOSYALARI

➢ *TEMEL KONSEPTLER*

Bir kayıt dosyasına bakıyoruz :

$$F_1, F_2, \ldots, F_m, F_{m+1}, F_{m+2}, \ldots, F_{2m}, \ldots$$

anahtar değerlerinin artan sırasına göre düzenlenmiş olanlar,

Kayıt tipinde iki alan bulunan yeni bir I(F) dosyası oluşturuyoruz:

$$k(F_1) < k(F_2) < \ldots < k(F_m) < \ldots$$

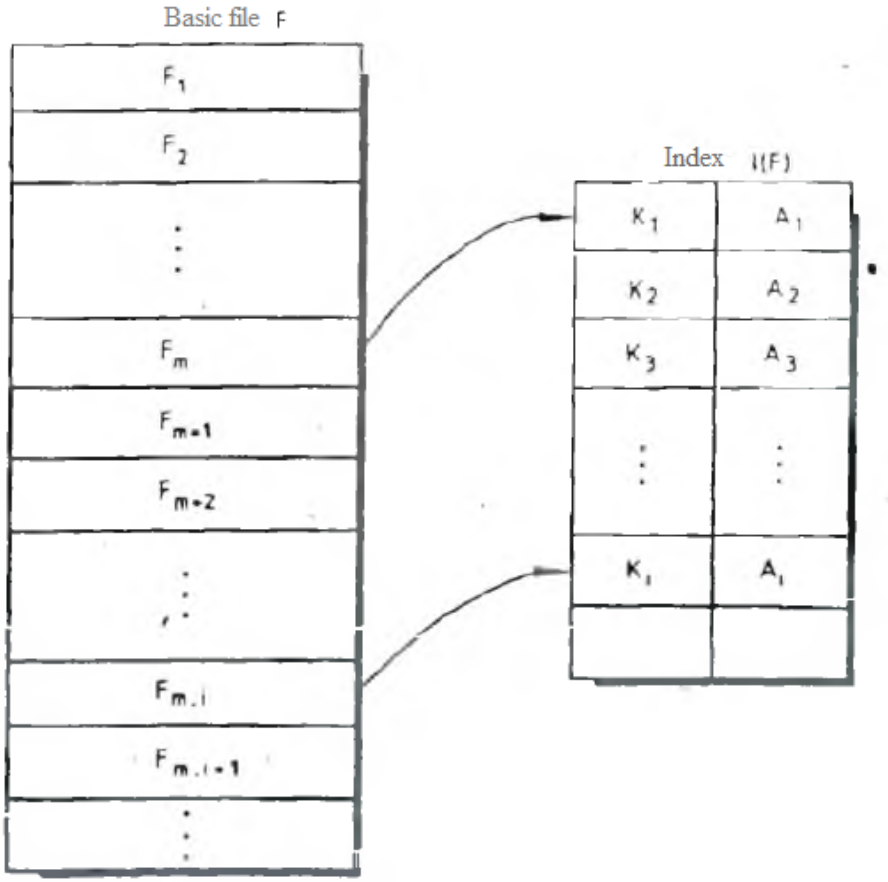
$$(K_i, A_i), \qquad i = 1, 2, 3, \ldots$$

, burada A, F temel dosyasının ı grubunun kayıtlarının saklandığı fiziksel bloğun adresidir, i=1,2,....

ENDEKS DOSYALARI

- ***TEMEL KONSEPTLER***

Temel indeks dosyası türü



Co-funded by the Erasmus+ Programme of the European Union

ENDEKS DOSYALARI

➢ *TEMEL KONSEPTLER*

- Yeni alınan I(F) dosyasına F dosyasının indeksi adı verilir ve böyle bir indekse sahip olan F dosyasına indeks dosyası denir.
- Aslında indeks, ana dosyadaki kayıtlara erişim için prosedür (yöntem) oluşturmak amacıyla kullanılabilecek bir tablodur.
- Bu erişim yöntemi indeks erişim yöntemi olarak bilinir.
- Ana dosya sıralandığı için indeksin de sıralanacağı sonucu çıkar.
- Bu, aramanın önemli ölçüde hızlandırılabileceği bir dizin organizasyonu için fırsatların olduğu anlamına gelir.
- Bu gerçek çok önemlidir çünkü ana dosyadaki arama hızı büyük ölçüde dizinindeki arama hızına bağlıdır.

ENDEKS DOSYALARI

➢ *DİZİN-SIRALI DOSYA ORGANİZASYONU*

- Ana dosyanın her m kaydı (m>1) için indekste bir kayıt yaratılıyorsa, bu indeksin bu durumda ana dosyadan çok daha küçük boyutta bir dosya olduğu anlamına gelir.
- Sonuç olarak, endeksin tamamen veya kısmen RAM'de mevcut olması mümkündür ve bu, kayıtlar için toplam arama süresini gerçekten büyük ölçüde azaltacaktır.
- Bununla birlikte, ana dosyanın çok büyük bir hacme sahip olduğu ve dolayısıyla karşılık gelen dizinin de çok büyük olduğu durumlar vardır.

ENDEKS DOSYALARI

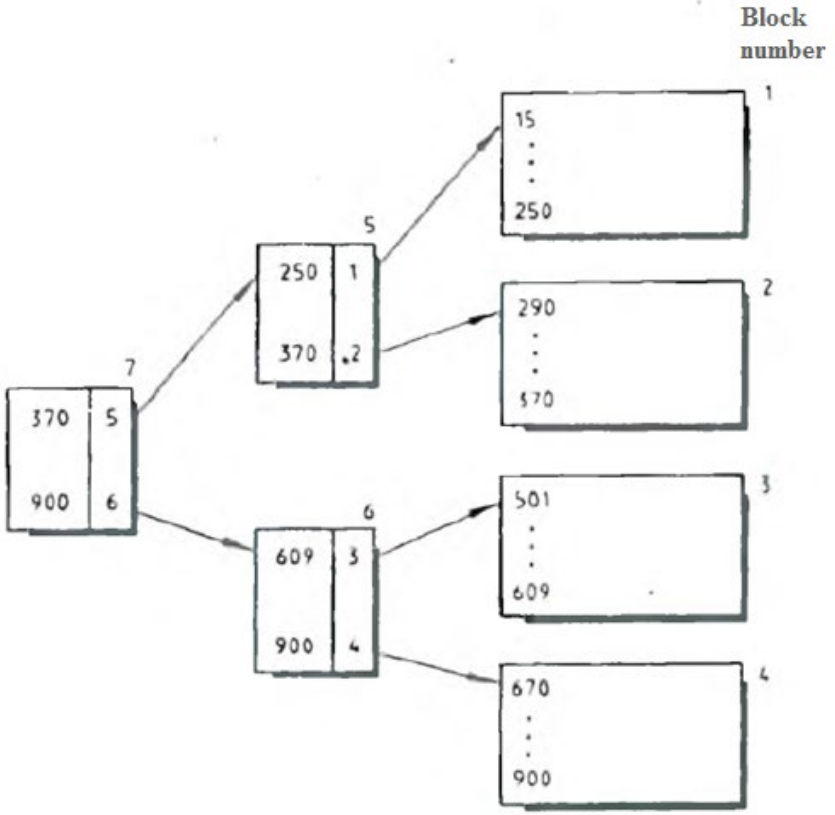
➢ *DİZİN-SIRALI DOSYA ORGANİZASYONU*

- İndeks aynı zamanda bir dosya olduğundan, onun da indekslenebileceği ve yeni bir indeks oluşturulabileceği sonucu çıkar.
- Bu indeksleme süreci devam ederek çok seviyeli bir indeks elde edilebilir.
- Böyle bir organizasyonla dosyadaki kaydın aranması somut bir örnekle ele alınacaktır.

Co-funded by the Erasmus+ Programme of the European Union

ENDEKS DOSYALARI

- ***DİZİN-SIRALI DOSYA ORGANİZASYONU***



Index-sequential organization with two levels index

ENDEKS DOSYALARI

➢ *DİZİN-SIRALI DOSYA ORGANİZASYONU*

- K=576 anahtarlı bir kayıt arayalım. İlk önce üst düzey dizin kontrol edilir.
- Bizim durumumuzda bu ikinci düzey endekstir. Aranan kaydın K anahtarı için $370 < K < 900$ koşulu karşılanır.
- Bu nedenle aramaya birinci düzey indeksin 6 numaralı bloğunda devam edilmelidir.
- $K < 609$ olduğundan, istenen kayıt en sonunda anahtarları [501,609] aralığında olan kayıtların saklandığı blok 3'te olacaktır.

ENDEKS DOSYALARI

➢ *DİZİN-SIRALI DOSYA ORGANİZASYONU*

- Dizin sıralı dosyalardaki kayıtlar, aranan dizinin anahtarı ayarlanarak ilkinden sonuncuya doğru veya rastgele sırayla okunabilir.
- Dizin sıralı organizasyonla ilgili sorunlar iki durumda ortaya çıkar:

1. Ana dosyanın organizasyonu nedeniyle basit bir şekilde tanımlanan belirli bir bloğa bir kayıt girmek gerekir ve bu blok zaten tamamen doludur.

2. Belirli bir bloğun son kaydının bile ortadan kaldırılması gerekir.

ENDEKS DOSYALARI

➢ *DİZİN-SIRALI DOSYA ORGANİZASYONU*

- İlk sorun iki şekilde çözülebilir:

a) taşma alanı adı verilen ek bir dosya kullanılır;

b) Taşan bloğun kayıtlarının bir kısmı (örneğin yarısı) yeni bir blokta taşınır ve bu, dizinde yeni bir dosya oluşturulmasına yol açar.

Blokların ana dosyadan tekrar tekrar taşması durumunda, indeks sıralı erişim yönteminin verimliliği ciddi şekilde düşebilir, bu nedenle yeniden düzenlenmesi zorunlu hale gelir.

ENDEKS DOSYALARI

- ***DİZİN-SIRALI DOSYA ORGANİZASYONU***

- İkinci sorunu çözmek daha basittir ve yalnızca endeks değişikliklerine indirgenir. Dosyaların indeks sıralı organizasyonunun bir çeşidi, keyfi indeks organizasyonu olarak adlandırılır.
- Bununla birlikte, bir gruptaki girişlerin sayısı birdir (yani m= 1), bunun anlamı şudur:

(a) ana dosyanın her öğesi bir dizin kaydına karşılık gelir; böyle bir indekse yoğun indeks adı verilir (endeks tutarlı organizasyondaki indeks yoğun değildir);

b) ana dosyanın sipariş edilmesine gerek yoktur; bu, kayıt ekleme ve kaldırma işlemlerini kolaylaştıracaktır.

ENDEKS DOSYALARI

- Veritabanındaki depolama yapıları, işletim sistemlerinin kullandığı dosyalara benzer dosyalardan oluşur.
- Örneğin bir ilişkiyi depolamak için bir veri dosyası kullanılabilir.
- Veri dosyasında bir veya daha fazla indeks dosyası bulunabilir.
- Her dizin dosyası, arama anahtarının değerlerini, arama anahtarının öznitelik(ler)i için bu değere sahip olan veri dosyası kayıtlarına yönelik işaretçilerle ilişkilendirir.

ENDEKS DOSYALARI

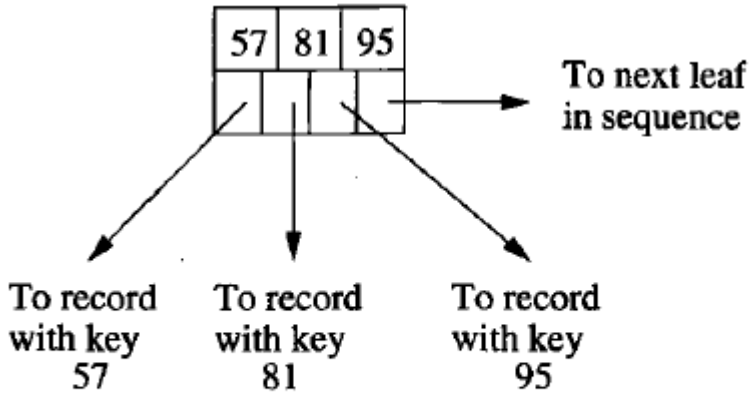
- İndeksler "yoğun" olabilir, bu da veri dosyasının her kaydı için indeks dosyasında bir giriş olduğu anlamına gelir.
- "Seyrek" olabilirler, yani dizinde yalnızca bazı veri kayıtlarının temsil edildiği, genellikle veri dosyası bloğu başına bir dizin girişinin temsil edildiği anlamına gelir. Dizinler ayrıca "birincil" veya "ikincil" olabilir.
- Birincil dizin, veri dosyasındaki kayıtların konumunu belirlerken ikincil dizin belirlemez. Örneğin, bir ilişkinin birincil anahtarında birincil dizin oluşturmak ve diğer bazı niteliklerde ikincil dizinler oluşturmak yaygındır.

B-Ağaçlar

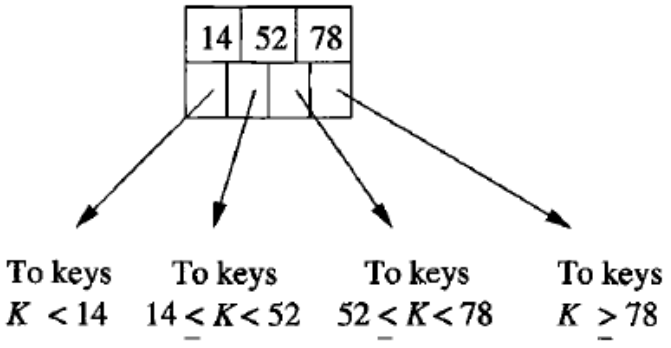
- Bir veya iki dizin düzeyi genellikle sorguları hızlandırmada çok yardımcı olsa da, ticari sistemlerde yaygın olarak kullanılan daha genel bir yapı da vardır. Bu veri yapıları ailesine B ağaçları denir ve en sık kullanılan özel değişken B+ ağacı olarak bilinir. Özünde:
  - B-ağaçları, indekslenen dosyanın boyutuna uygun olan sayıda indeks düzeyini otomatik olarak korur.
  - B-ağaçları, kullandıkları bloklardaki alanı, her bloğun yarı kullanılmış ile tamamen dolu olacak şekilde yönetir.

B-Ağaçlar

- B-ağacı, bloklarını dengeli bir ağaç halinde düzenler; bu, kökten yaprağa giden tüm yolların aynı uzunluğa sahip olduğu anlamına gelir. Tipik olarak bir B ağacında üç katman vardır: kök, bir ara katman ve yapraklar, ancak herhangi bir sayıda katman.

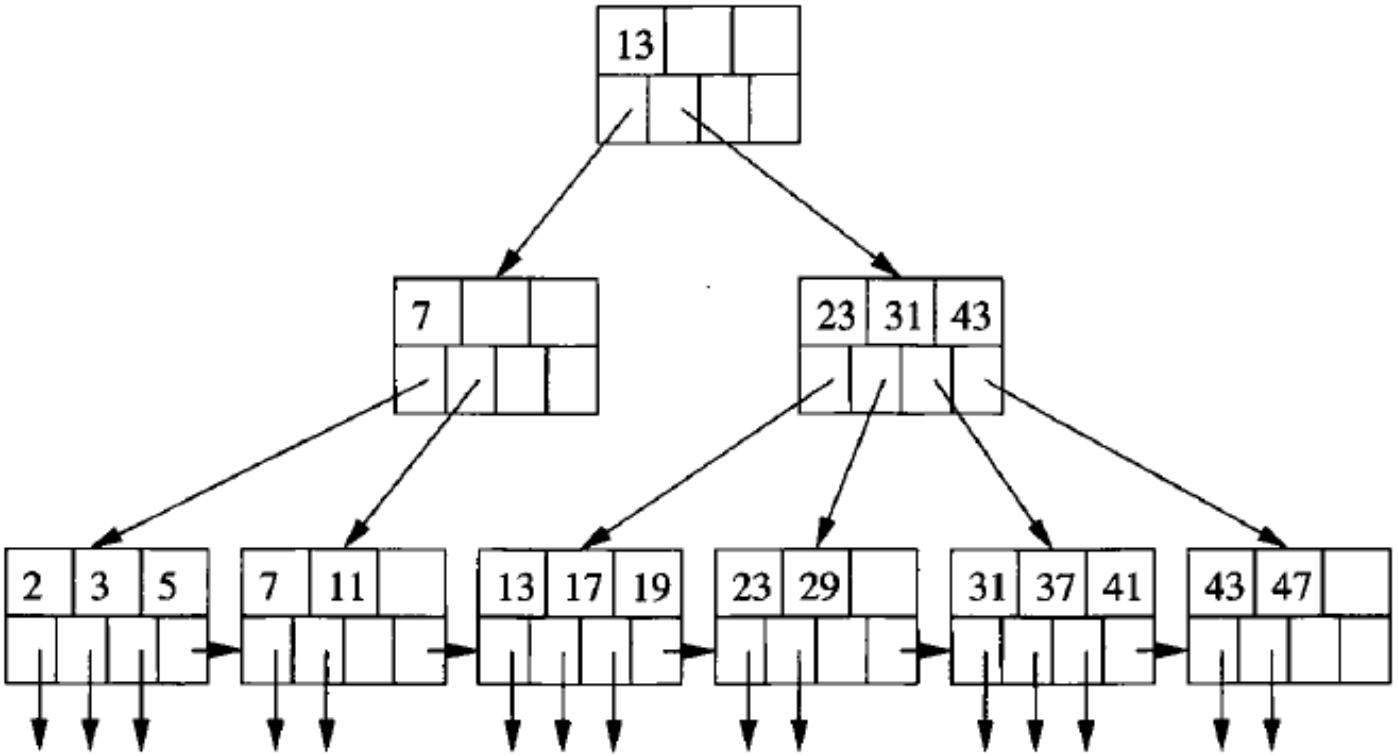


A typical leaf of a B-tree



A typical interior node of a B-tree

B-Ağaçlar



A B-tree

HASH-DOSYALARI

➢ *TEMEL KONSEPTLER*

- Hashing, her kaydın anahtarının belirli bir işlev kullanılarak bir adrese dönüştürüldüğü doğrudan erişim yöntemidir.
- Her biri belirli bir F dosyası için olası bir anahtar olan öğeler kümesine anahtar uzayı denir ve $S_K(F)$ ile gösterilir.
- Bir F dosyasındaki kayıtların başlangıç adresleri olan adresler kümesine dosyanın adres alanı adı verilir ve $S_A(F)$ ile gösterilir.
- $S_K(F)$ ve $S_A(F)$ 'nin elemanları arasında bir uygunluk kuran herhangi bir h_F fonksiyonu, i. e.
- geleneksel olarak karma işlevi olarak adlandırılır.

$$h_F : S_K(F) \longrightarrow S_A(F).$$

HASH-DOSYALARI

➢ *TEMEL KONSEPTLER*

- F dosyasının kayıtları için anahtar olarak en fazla 30 karakter uzunluğundaki karakter dizilerinin kullanılmasına izin verin ve izin verilen karakterler Latin alfabesinin büyük harfleridir.
- O halde $S_K(F)$ uzayının tüm elemanlarının sayısı 26^{30} olacaktır.
- Her karma işlevi $S_K(F)$ alanını aşağıdaki özelliklerle K1, K2, ..., Kn parçalarına "böler":

а) $S_K(F) = K_1 \cup K_2 \,..\, K_n$

б) $K_i \cap K_j = \varnothing$

в) If $h_F(k_\alpha) = h_F(k_\beta)$, Than such K_i, exists so that $k_\alpha \in K_i$ и $k_\beta \in K_i$.

г) If $k_\alpha \in K_p$ и $k_\beta \in K_q$, $p \neq q$, Than $h_F(k_\alpha) \neq h_F(k_\beta)$.

HASH-DOSYALARI

➢ *TEMEL KONSEPTLER*

- Başka bir deyişle, her karma işlevi, anahtar alanını her biri kayıtları - eşanlamlıları içeren parçalara böler. İki veya daha fazla kayıt anahtarının aynı $S_A(F)$ adresine eşlenmesine geleneksel olarak çarpışma adı verilir.
- Organizasyonu karma işlemine dayanan dosyalara genellikle karma dosyaları denir.
- Hash dosyalarıyla çözülmesi gereken en ciddi sorun çarpışma çözümüdür.
- Hash fonksiyonunu tanımlamak zor ve sorumlu bir iştir. Rastgele bir S_K alanı için minimum sayıda çarpışma üreten evrensel bir karma işlevi tanımlamak zordur.

HASH-DOSYALARI

➢ *TEMEL KONSEPTLER*

- Çeşitli karma fonksiyonlarının belirli anahtar türleri için uygun olduğu bilinmektedir. Her karma işlevi için karşılanması her zaman kolay olmayan belirli gereksinimler vardır:

a) girişleri adres alanına eşit ve rastgele dağıtmak;

b) anahtarın tüm parçaları karma işlevi hesaplamasına katılmalı ve anahtarlarda küçük farklılıklar olması durumunda karma işlevi S_A 'dan farklı adresler üretmelidir;

c) Hashing sırasında oluşacak çarpışma sayısı mümkün olduğu kadar az olmalı ve S_A 'ya ait n eleman için iki kaydın eş anlamlı olma olasılığı 1/n olmalıdır.

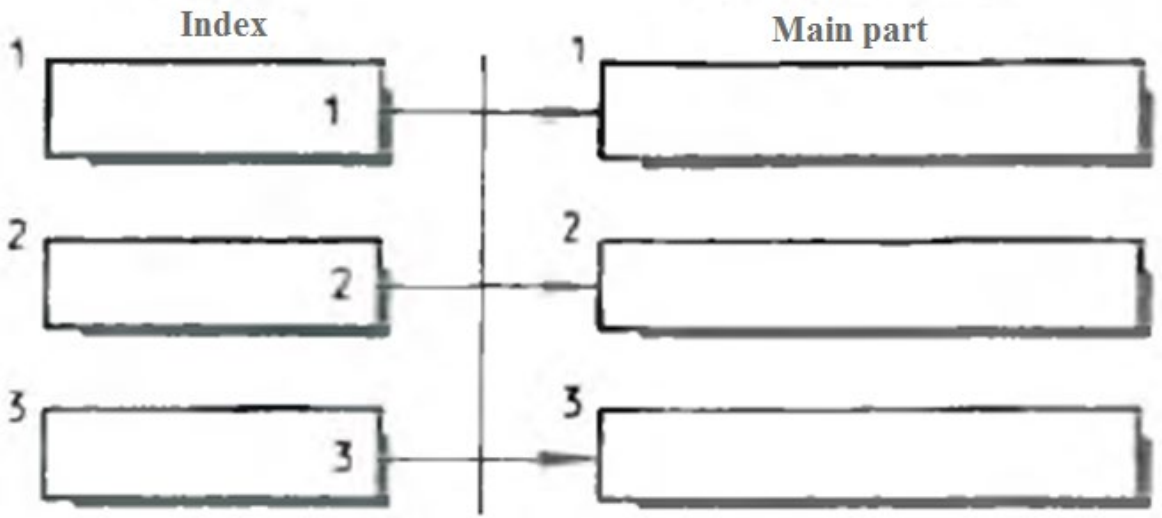
Co-funded by the Erasmus+ Programme of the European Union

HASH-DOSYALARI

- Boyutu önceden sabitlenen ve çalışma sırasında değişmeyen karma dosyasına **statik karma dosyası** denir.
- Statik hash dosyaları, kullanılan ilk hash dosyalarıydı. Boyutları yaratıldıklarında belirlenecektir.
- Statik karma dosyası bloklardan veya sözde bölgelerden oluşur. Her bölgede yalnızca eşanlamlı mantıksal kayıtlar depolanır
- Yeni bir kayıt eklendiğinde, bir bölgedeki eş anlamlılar listesine son giriş olarak kaydedilir. Hiçbir bölge dolana kadar tek giriş-çıkış işlemi ile ekleme ve arama işlemleri gerçekleştirilir.

HASH-DOSYALARI

- **Dinamik karma dosyası (kısaca DH dosyası)** iki bölümden oluşur; veri kayıtlarını içeren bir ana bölüm ve indeks adı verilen ek bir bölüm:



DH-file - main part and index

HASH-DOSYALARI

- Başlangıçta indeks, her biri aracılığıyla DH dosyasının bir bloğuna erişilen m öğeyi içerir.
- R kayıtları bir blokta (DH dosyasının) saklanabilir.
- Hash fonksiyonu ile (H ile gösterelim) eklenen kayıtların her biri indeksin m elemanlarından birine, yani i'ye tahsis edilir. e. karma işlevi şu forma sahiptir:

$$H : K \longrightarrow (1, 2, \ldots, m) ,$$

burada K, DH dosyası kayıtlarının anahtar uzayıdır.

R kayıtları bir bloğa girilirse, i. e. blok tamamen doldurulur, ardından aşağıda özetlenen bir dizi eylem gerçekleştirilir:

Co-funded by the Erasmus+ Programme of the European Union

HASH-DOSYALARI

1. Ana kısma yeni bir boş blok eklenir.

2. Eski bloktaki kayıtların yeniden düzenlenmesi gerçekleştirilir ve bunların bir kısmı yeni eklenen bloğa gider.

3. Karşılık gelen indeks öğesi, ikili ağaç olarak büyüdükçe ve geliştikçe değiştirilir.

Tartışılan karma dosyası organizasyonuna dayalı olarak dinamik karma dosyalarının birçok farklı çeşidi oluşturulmuştur.

HASH-DOSYALARI

Hash Fonksiyonu Seçimi :

- Ortaya çıkan tam sayının anahtarın görünüşte rastgele bir işlevi olması için karma işlevi anahtarı "karmalamalı" olmalıdır.

- Bu, bir kayda erişim için ortalama süreyi artırır.

- Ayrıca karma fonksiyonunu birçok kez hesaplayacağımız için hesaplaması kolay olmalıdır.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar :

1. Dosya türleri nedir?
2. İndeks dosyaları nedir?
3. Hash dosyaları nedir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

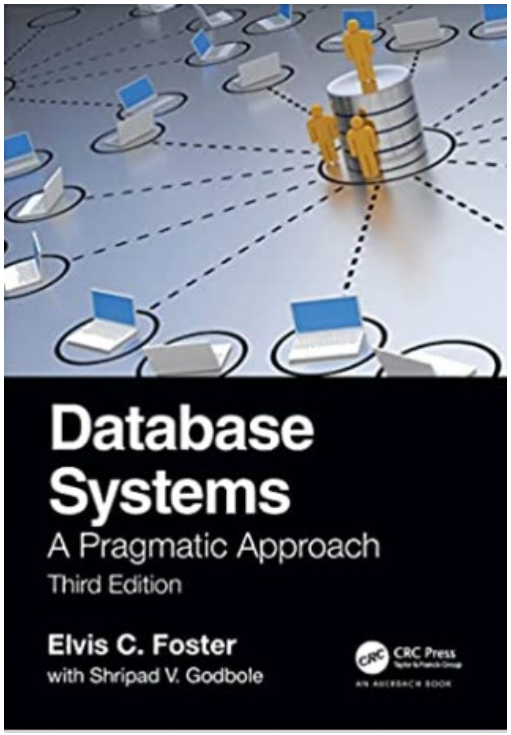
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

- **Modül 2. Veritabanlarında ilişkisel yaklaşım**
  - **Konu 1. İlişkisel model - temel kavramlar, ilişkisel şemalar.**
    - Ders 1. İlişkisel yaklaşım. İlişkisel model.

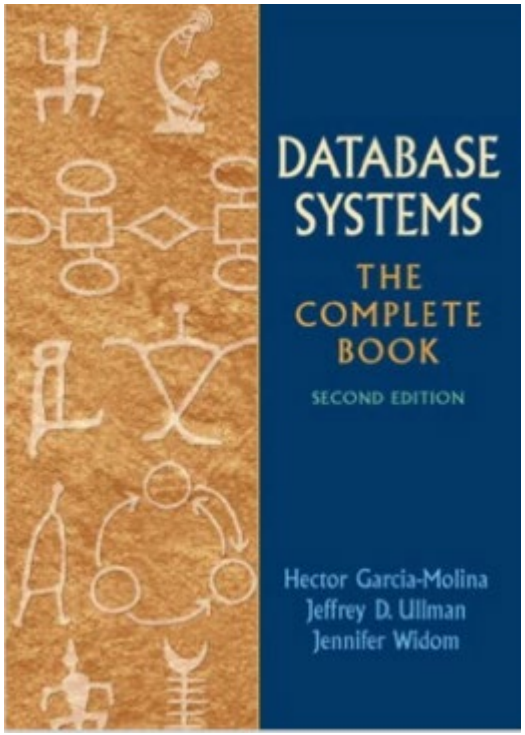
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

İLİŞKİSEL YAKLAŞIM. İLİŞKİSEL MODEL.



Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.



Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

Azalov, P. (1991) Database, Sofia.

İLİŞKİSEL MODEL.

- İlişkisel model 1970 yılında E. F. Codd'un "A Relational Model of Data for Large Shared Data Banks" adlı makalesiyle başladı.
- İlişkisel modelin temeli, n-üye ilişkisinin matematiksel kavramıdır.
- Her ilişki, n bileşenden oluşan ve n-demetler olarak adlandırılan bir öğeler kümesidir.
- Aslında bir ilişki, bir nesne sınıfını modeller ve ilişkinin her bir kümesi, o sınıfın belirli bir nesnesini temsil eder.

İLİŞKİSEL MODEL.

- İlişkisel model bize verileri temsil etmenin bir yolunu sunar: ilişki adı verilen iki boyutlu bir tablo olarak.
- Aşağıdaki şekilde "Filmler" adı verilen bir ilişkinin örneği verilmektedir.

title	*year*	*length*	*genre*
Gone With the Wind	1939	231	drama
Star Wars	1977	124	sciFi
Wayne's World	1992	95	comedy

Relation "Movies

- Her satır bir filmi, her sütun ise filmlerin bir özelliğini temsil eder.

İLİŞKİSEL MODEL.

- İlişkisel modelde bir R nesneleri sınıfı ile onun karakterize edici nitelikleri kümesi A_1, A_2,...,A_n arasında var olan yazışma, $R(A_1, A_2,...,A_n)$ kaydıyla ifade edilir.
- İlişkisel model, DB'yi, ilişkisel cebir adı verilen işlemleri kullanarak en basit matematiksel nesnelerle (sayılar) çalıştırılabilen bir dizi basit (normalleştirilmiş) ilişkiler olarak görme fırsatı verir.
- İlişkisel model aynı zamanda verinin dahili temsilini ve ona nasıl erişileceğini bilmeye ihtiyaç duymadığından yüksek derecede veri bağımsızlığı da sağlar.
- İlişkisel modelin diğer DM'lere göre önemli bir avantajı, nesne sınıflarının ve aralarındaki ilişkilerin tek tip temsilidir.

İLİŞKİSEL YAPILAR

- İlişkisel model üç kavramla oluşturulmuştur: etki alanı, nitelik ve ilişki. D_1, D_2, D_3,..., D_n
- D_1, D_2,..., D_n, n>=1 küme olsun, mutlaka farklı olsun. Bu kümelerin her biri, bir niceliğin kabul edilebilir değerleri kümesi olarak kabul edilir ve aynı zamanda bölge veya etki alanı olarak da adlandırılır.
- Bölgeler sonsuz olabilir veya sonlu sayıda öğe içerebilir.
- D_1, D_2, D_3,..., D_n n kümesinin Kartezyen çarpımını ele alıyoruz:

$$D_1 \times D_2 \times \ldots \times D_n = \{ \langle d_1, d_2, \ldots, d_n \rangle \mid d_i \in D_i, \quad i = 1, 2, \ldots, n \}$$

Co-funded by the Erasmus+ Programme of the European Union

İLİŞKİSEL YAPILAR

- **Örneğin** D_1 bir kütüphanedeki kitapların imza numaralarını gösteriyorsa,

$$D_1 = \{895CIX,\ 1028CX,\ 964AIX\},$$

- D_2 ise bir kurumdaki çalışan sayısını,

$$D_2 = \{312, 205, 128\},$$

- o zaman $D_1 x D_2$ tabloda gösterilen eleman çiftlerini içerecektir:

Co-funded by the Erasmus+ Programme of the European Union

İLİŞKİSEL YAPILAR

Signature №	Employee №
859CIX	312
859CIX	205
859CIX	128
1028CX	312
1028CX	205
1028CX	128
964AIX	312
964AIX	205
964AIX	128

Co-funded by the Erasmus+ Programme of the European Union

İLİŞKİSEL YAPILAR

- <İmza No. , Çalışan No. > öğe çiftinin anlamı, **örneğin,** "859CIX imza numaralı bir defter, 312 numaralı çalışan tarafından geçici olarak alınmıştır" olabilir.
- Sonuç olarak, $D_1 x D_2$ Kartezyen çarpımının her çiftinin anlamlı olmadığı sonucuna varılabilir.
- **Örneğin** imza numarası 859CIX olan defter tek nüsha ise ve 312 numaralı çalışan tarafından alınmışsa <859C1X,205> ve <859C1X,128> eleman çiftleri anlamsızdır.
- Diğer eleman çiftleri için de benzer tartışmalar yapılabilir.

İLİŞKİSEL YAPILAR

- Bu bizi Kartezyen çarpımı D_1xD_2'nin alt kümelerini dikkate alma fikrine getiriyor. $\boldsymbol{D_1xD_2x...xD_n}$
- **N bölgenin Kartezyen çarpımı $\boldsymbol{D_1xD_2x...xD_n}$ 'nin her bir alt kümesine ilişki denir.**
- **Örnek olarak** belirli bir zamanda kütüphaneden alınan kitapların ilişkisini düşünün.

$R \subseteq D_1 x D_2$

R	Signature №	Employee №
	859CIX	312
	964AIX	205
	1028CX	205

Co-funded by the Erasmus+ Programme of the European Union

İLİŞKİSEL YAPILAR

- Kabaca söylemek gerekirse, zamanın başka bir noktasında bu ilişkinin içeriği farklı olacaktır.
- İlişkileri temsil etmenin esas olarak iki yolu vardır.
- Bunlardan biri ilişki kavramının tanımına dayanmaktadır; bu kavramın her bir öğesi tam olarak sıralı bir d_1, d_2, kümesidir. . . . , $d_n>$, burada d_i , D_i alanına aittir, $i=1,2,...,n$.
- Dolayısıyla bir ilişkinin, satırların n'li olduğu ve sütunların yalnızca aynı etki alanından öğeler içerdiği bir öğe tablosu olarak değerlendirilebileceği açıktır.

İLİŞKİSEL YAPILAR

- Bir ilişkinin tablo halinde gösteriminin her adlandırılmış sütununa geleneksel olarak ilişkinin bir niteliği denir.
- İlişkinin bilgi içeriği niteliklerinin sıralamasına bağlı olmadığından ikinci bir temsil yolu aramak doğaldır.
- Tablo şeklinde gösterimin aksine, ikinci yöntem, tablodaki sütunların adlandırıldığını ve dolayısıyla bunların düzenine uygunluğun gerekli olmadığını varsayar.
- Dolayısıyla herhangi bir ilişki, her biri öznitelik kümesinde tanımlanan ve ilişkinin bazı n-demetini (öğesini) tanımlayan bir işlevler kümesi olarak da temsil edilebilir.

İLİŞKİSEL YAPILAR

- **Örneğin,** R, A = {İmza No, Çalışan No} niteliklerine sahip bir ilişki ise, formun f_1, f_2 *ve* f_3 üç fonksiyonuyla tanımlanabilir:

$$f : A \longrightarrow D_i, \qquad i = 1, 2$$

where:

f_1(**Signature №**) = 856CIX
f_1(**Employee №**) = 312
f_2(**Signature №**) = 964AIX
f_2(**Employee №**) = 205
f_3(**Signature №**) = 1028CIX
f_3(**Employee №**) = 205.

- Aslında iki gösterimin eşdeğer olduğu ve bu durumun her ikisini de kullanmamıza izin verdiği sonucuna varmak zor değil.

İLİŞKİSEL YAPILAR

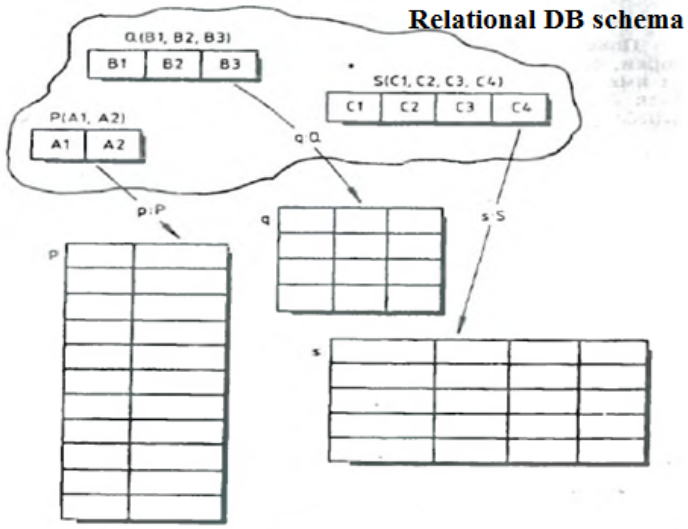
- İlişkiler, içeriği değişken olan ve herhangi bir zamanda bir nesne sınıfının mevcut durumunu temsil eden veri yapılarıdır.
- Bu durumda önemli olan, varlıkları boyunca yapılarını korumalarıdır.
- Her ilişkinin yapısı ya da diğer adıyla ilişkinin türü, ilişki şemasıyla belirlenir. $R(A_1,A_2,A_n)$.
- Her ilişkinin ilişki şeması, ilişkinin adı ve ona karşılık gelen öznitelikler ile tanımlanır. A_1,A_2,A_n,
- **Örneğin,** A_1,A_2,A_n, R isimli n üyeli bir ilişkinin nitelikleri ise, ilişki şeması $R(A_1,A_2,A_n)$ ifadesiyle yazılacaktır.

İLİŞKİSEL YAPILAR

- Niteliklerin her biri yalnızca belirli bir alandan değer alabildiğinden, gerektiğinde bazen daha eksiksiz olan $R(A_1{:}D_1, A_2{:}D_2, \ldots, A_n{:}D_n)$ kaydı kullanılır.
- $R(A_1, A_2, \ldots, A_n)$ bir ilişkisel şema olsun ve r de bu ilişkisel şema ile bir ilişki olsun. Bu gerçek kısaca şu şekilde ifade edilir: r:R.
- Nesne sınıflarını ve aralarındaki ilişkileri tanımlayan tüm ilişkisel şemaların kümesi DB şemasıdır. $R1, R2, \ldots, Rk$ $\{R_1, R_2, \ldots, R_k\}$.
- Eğer $R1, R2, \ldots, Rk$ DB şemasını oluşturan ilişkisel şemalar ise bu kısaca şu şekilde yazılır: $R_1, R_2, \ldots, R_k\}$.

İLİŞKİSEL YAPILAR

- Bu ilişkisel şemaların her birinin güncel bir değeri vardır ve bu, karşılık gelen ilişkinin içeriğidir.
- *i=1,2* 'de *r:R* olan herhangi bir . r_1, r_2,...,r_κ somut ilişkiler kümesi. ,. . . ,*k*, ilişkisel DB'nin (RDB) mevcut durumu olarak adlandırılır.



Relational database with three relations

İLİŞKİSEL YAPILAR

- Bireysel ilişkisel şemaların tanımı çok serbesttir, ancak çoğu zaman bunları oluştururken aşağıdaki iki **kurala** uyulur:

1. *Her nesne sınıfı, şeması aynı zamanda ilişkinin nitelikleri haline gelen tüm niteliklerini içeren bir ilişki tarafından temsil edilir. İlişkinin her n-tuple'ı sınıfın belirli bir nesnesini (örneği) temsil eder. Nesne sınıfının anahtar niteliği veya anahtar niteliklerinin listesi, i. e. Sınıftaki bireysel örnekleri benzersiz şekilde tanımlayanlar ilişkinin anahtarı olarak alınır.*
2. *İki veya daha fazla nesne sınıfı arasındaki ilişkiler, ilişkisel şeması bu nesne sınıflarının her birinin temel niteliklerini içeren bir ilişkiyle temsil edilir.*

İLİŞKİ ANAHTARI

- Her ilişki tekrarlanmayan n-demetlerden oluşan bir küme olduğundan, her ilişkinin bir anahtarı vardır.
- Bir ilişkinin birden fazla anahtarının olması doğaldır.
- Eğer bir *A* öznitelik seti, bir *R* ilişkisi için anahtar ise, o zaman $A \subset X$'in *R*'nin elemanlarını benzersiz bir şekilde tanımlayacağı herhangi bir *X* öznitelik kümesi R'dir.
- Bu nedenle her anahtar aynı zamanda aşağıdaki şekilde ifade edilebilecek bir asgari gereksinime de tabidir:

1. Herhangi iki eleman için $r_1 \in R$ ve $r_2 \in R$, $r_1[K] \neq r_2[K]$ geçerlidir;
2. Özellik 1'in geçerli kaldığı *K* niteliğinin hiçbir alt kümesi yoktur.

Co-funded by the Erasmus+ Programme of the European Union

İLİŞKİ ANAHTARI

- Her ilişkinin olası anahtarlarından biri ilişkinin anahtarı olarak seçilir ve buna birincil anahtar denir.
- Geri kalan anahtarlar için diğer terimler kullanılır: olası, olası veya alternatif.
- Bileşenlerden birinin değerinin olduğu durumlar vardır, yani. e. Bir ilişkiye belirli bir öğe girilirken öznitelik bilinmiyor veya tanımlanmıyor. Böyle bir değere genellikle sıfır denir.
- Bir R ilişkisi için yabancı anahtar *K′*, *R* için anahtar olmayan bir nitelik veya nitelikler listesidir, ancak *K*'nin birincil anahtar olduğu bir *Q* ilişkisi vardır.
- Birincil ve yabancı anahtarlar temel olarak iki ilişkinin n-demetleri arasında bağlantı kurmanın bir yoludur.

İLİŞKİ ANAHTARI

- İlişkisel modelde iki ana bütünlük kısıtlaması uygulanır :

1. *Birincil anahtarı oluşturan niteliklerin değerleri sıfır olamaz.*
2. *Yabancı anahtar olan bir özniteliğin sıfırdan farklı her değeri için, o değeri içeren başka bir ilişkiden bir anahtar niteliğinin bulunması gerekir.*

- Bu iki kısıta ek olarak iki kısıt daha kullanılmaktadır. - *bir özelliğin değerleri belirli bir aralıkta olmalı ve bir özelliğin değeri (ana özellik olması şart değil) sıfır olmamalıdır.*
- Bütünlük kısıtlamalarının bir başka geniş sınıfı, işlevsel bağımlılıklar kullanılarak tanıtılmıştır.

Co-funded by the Erasmus+ Programme of the European Union

İLİŞKİ ANAHTARI

R1

Employee №	Book	
	Signature №	Date
128	723AX 674CIX 1021BIX	15.09.89 18.10.89 09.08.89
305	624AU	28.05.89
64	503AU 372BUI 1009BXI 506AUI	30.06.89 15.07.89 18.07.89 09.08.89

R1 **(Employee №, book (Signature №, date of taking))**

Normalleştirilmemiş ilişki

Co-funded by the Erasmus+ Programme of the European Union

İLİŞKİ ANAHTARI

R2

Employee №	Signature №	Date
128	723AX	15 09 89
128	674CIX	18 10 89
128	1021BIX	09 08 89
305	624AU	28 05 89
64	503AU	30 06 89
64	372BUI	15 07 89
64	1009BXI	18 07 89
64	506AUI	09 08 89

R2 Employee №, Signature №, Date of taking)

Normalleştirilmiş ilişki

İLİŞKİ ANAHTARI

- İlişkisel model, ilişkilerin tanımlanmasını ve bunlar üzerinde işlem yapılmasını gerektirir, ancak herhangi bir ilişkiyle değil.
- İlişkisel model için izin verilen ilişkiler, normalleştirilmiş ilişkiler olarak adlandırılan ilişkilerdir, yani. e. , ilişkisel şemaları yalnızca atomik (basit, bölünemez) nitelikler içerenler.
- Yukarıdaki iki şekilde iki ilişki gösterilmiştir. Bunlardan biri R_1 normalleştirilmemiş, diğeri ise aynı anlam ve içeriği ifade eden R_2 normalleştirilmiş bir ilişkidir.
- Matematiksel nesneler olarak da düşünüldüğünde farklıdırlar. Bunlardan ilki binom bağıntısı, ikincisi ise üç terimli bağıntıdır.
- İlişkisel normalleştirme, ilişkisel veri tabanlarının tasarımıyla doğrudan ilgilidir ve bu konular ayrı bir derste tartışılacaktır.

İLİŞKİ ÖZELLİKLERİ

- Veritabanındaki her ilişkinin (tablonun) benzersiz bir adı vardır.
- Her özelliğin belirli bir ilişki içinde benzersiz bir adı vardır.
- Her ilişki benzersiz kayıtlar içerir, yinelenen aynı kayıtları içeremez.
- Kayıtların belirli bir ilişkiye veya belirli bir kayıttaki niteliklere yerleştirildiği belirli bir sıra yoktur.
- Kayıtlardaki değerler atomiktir; farklı veri türlerinden (farklı alanlardaki veriler) oluşamazlar veya bir hesaplamanın veya birleştirmenin sonucu olamazlar.

Verilerin bellekteki fiziksel organizasyonunun, yalnızca mantıksal veri organizasyonunun geçerli olduğu ilişkisel modelle alakası yoktur.

İLİŞKİ TÜRLERİ

- 2 tablo arasında ilişki oluşturmak için bunların ortak (eşit) bir alana sahip olması gerekir.

a) 1:1 ilişki – bire bir

b) 1:n ilişkisi – birden çoğa

в) m:n ilişkisi – çoktan çoğa

1:1 ilişkiler - bir tablodaki bir kayıt diğerindeki bir kayda karşılık gelir

-1 departman - 1 yönetici

-1 üniversite - 1 rektör

-1 ders - 1 ders sorumlusu

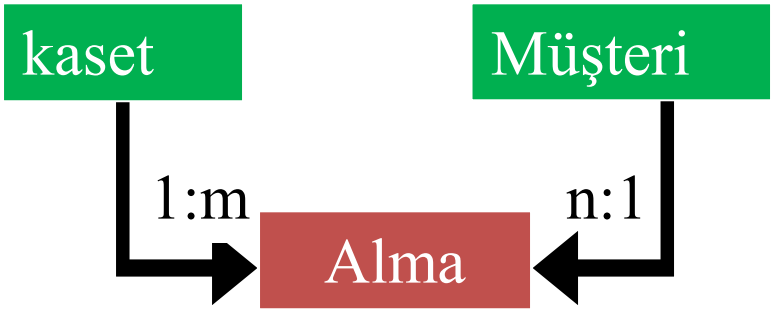
İLİŞKİ TÜRLERİ

- 1:n ilişki türü – birden çoğa – bir tablodaki bir kayıt diğerindeki birçok kayda karşılık gelir
  - 1 tür – birçok film
  - 1 departman – birçok çalışan
  - 1 okul – birçok öğrenci

  m:n ilişki türü - çoktan çoğa - bir tablodaki bir kayıt diğerindeki birçok kayda karşılık gelir ve bunun tersi de geçerlidir.

Co-funded by the Erasmus+ Programme of the European Union

İLİŞKİ TÜRLERİ



Çoka çoğa ilişkiler DBMS tarafından desteklenmez. Yeni bir tablo eklenerek 2 1:n ilişkisine bölünmelidirler

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. İlişkisel veritabanları öncelikle aşağıdakilerden hangisini kullanarak birden fazla tabloyu birbirine bağlar?
   - ✓ DDL ve DML dili
   - ✓ Birincil Anahtar ve Yabancı Anahtar
   - ✓ Süper anahtar ve benzersiz anahtar
   - ✓ Kontrol et ve BOŞ

2. Bir RDBMS'deki tablolar arasındaki farklı ilişki türlerini açıklayın.

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

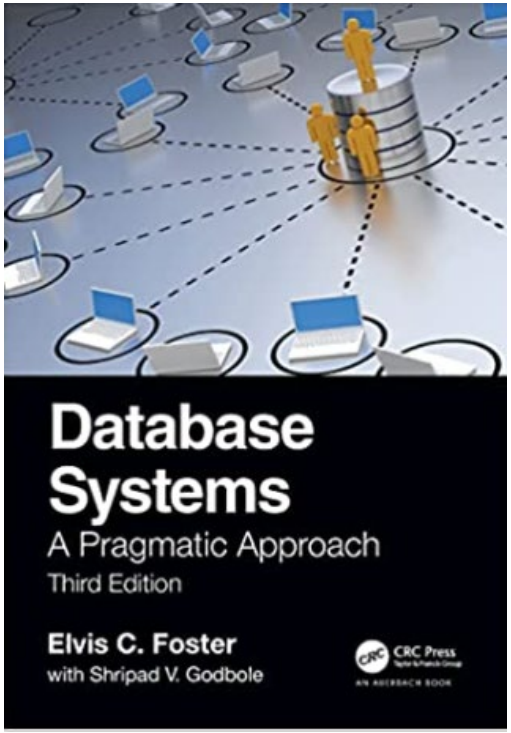
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

- **Modül 2. Veritabanlarında ilişkisel yaklaşım**
  - **Konu 1. İlişkisel model - temel kavramlar, ilişkisel şemalar.**
    - Ders 2. İlişkisel cebir.

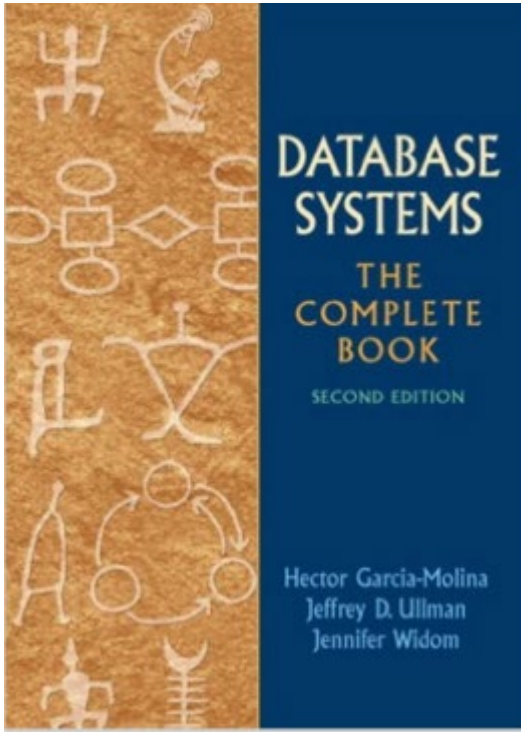
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

İLİŞKİSEL CEBİR



Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.



Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

Azalov, P. (1991) Database, Sofia.

Co-funded by the Erasmus+ Programme of the European Union

Giriş

- İlişkisel cebir ilişkisel veritabanını işlemek için temel işlemleri gösteren resmi bir dildir.
- Bir döngü düzenlemeden tek bir operatörle bir veya daha fazla ilişki halinde düzenlenmiş kortejleri (kayıtlar veya satırlar) işler.
- Yeni ilişkiler kurmanın yolunu gösterdiği için çekincelerle birlikte prosedürel dil olarak sınıflandırılabilir. Diğer ilişkisel dil türlerinin temelini oluşturur.
- İlişkisel analiz E. Codd tarafından geliştirilen, kullanıcının sonuç ilişkisinde nelerin yer alacağını belirlediği prosedürel olmayan resmi bir dildir.

Co-funded by the Erasmus+ Programme of the European Union

Giriş

- Bu dil, diğer ilişkisel dil türlerinin yeteneklerini belirlemek için kullanılır.
- CODD tarafından önerilen ilişkisel cebir işlemleri iki gruba ayrılır:
- **Çoklu işlemler–** kullanılırlar çünkü her ilişki genel olarak bir grup kortejden oluşur:
- Birlik;
- Kavşak
- Fark
- Kartezyen ürün.

Giriş

- CODD tarafından önerilen ilişkisel cebir işlemleri iki gruba ayrılır:

İlişkisel işlemler - özellikle veritabanı için geliştirildi:

- Seçme;
- Proje;
- Katılmak;
- Bölüm.

Yukarıda bahsedilen dillerin yüksek karmaşıklığı nedeniyle, bunlara dayalı olarak pratik veri işleme dilleri geliştirilmiştir.

Giriş

- $D_1, D_2, \ldots, D_n$ n, $n \geq 1$ tanım kümesi olsun. A_i niteliğinin D_i, $i=1,2,\ldots,n$. alanından değerleri aldığı $A_1, A_2, \ldots A_n$, niteliklerinin listesini ele alıyoruz. Bu listeyi kısaca sadece A ile gösteriyoruz.
- $B_1, B_2, \ldots B_m$ m niteliklerinin başka bir listesi olsun.
- A ve B niteliklerinin listeleri aşağıdaki durumlarda karşılaştırılabilir:
- $m=n$
- A_k ve B_k her $\kappa=1,2,\ldots,n$ için aynı türdendir.
- P, $R(A)$ ilişkisel şemasına sahip n-üyeli bir ilişki olsun ve q e, $R(B)$ ilişkisel şemasına sahip m-üyeli bir ilişki olsun; burada a ve b sırasıyla p ve q'nun elemanlarıdır ve: $a = <a_1, a_2, \ldots, a_n>$ u $b= <b_1, b_2, \ldots, b_n>$

Co-funded by the Erasmus+ Programme of the European Union

Giriş

- *a*'nın *b* ile birleşimi *a* *(n+m)*-torustur ve aşağıdaki gibi tanımlanır:

$$ab = \langle a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m \rangle .$$

- Yapılan varsayımlara göre, $a.A_i$ ile *a*'nın *p* öğesinin *i*'inci bileşenini belirtiyoruz. Benzer şekilde $[A_{i1},A_{i2},\ldots,A_{ik}]$ ile değerleri sırasıyla $a[D_{i1}, D_{i2},\ldots, D_{ik}]$, *i* olan *a* bileşenlerinin listesini belirtiriz.

$$a[A_{i1}, A_{i2}, \ldots, A_{ik}] = \langle a.A_{i1}, a.A_{i2}, \ldots, a.A_{ik} \rangle .$$

Co-funded by the Erasmus+ Programme of the European Union

Giriş

We consider 3-membered relation r

r	Employee №	Name	Age
x1	235	Hristo	38
x2	109	Kiril	27
x3	203	Vasil	42
x4	174	Boris	40

According to the given denotions:

$x1 = \langle 235, \text{Hristo}, 38 \rangle$

$x1[\text{Name}] = \text{Hristo}$

$x1[\text{Age, Employee №}] = \langle 38, 235 \rangle$

Giriş

- İlişkinin satırlarının (öğelerinin) bir kısmı hariç, tüm ilişkinin bazı kısımları (sütunları) alıntılanabilir. İlişkinin adı ve karşılık gelen nitelikler bu amaç için kullanılır.

r [Name] =

Hristo
Kiril
Vasil
Boris

r [Age, № of employee] =

38	235
27	109
42	203
40	174

İlişkisel operatörler - UNION

- *p* ve *q*'nun aynı *R(A1,A2,...,An)* ilişkisel şemasına sahip iki n-üyeli ilişki olmasına izin verin.
- *P* ve *q* ilişkilerinin birliği, *p* ve *q*'nun öğelerini içeren aynı ilişkisel şemaya sahip üçüncü bir r ilişkisidir,

$$r = \{t \mid t \in p \text{ VEYA } t \in q\}.$$

- Birleşim, genellikle "U" kümelerinin birleşimi için yaygın olarak kullanılan matematiksel işaretle gösterilen, ancak bazen aşağıdaki tanımla iki bağımsız değişkenli bir fonksiyon olarak da temsil edilen iki bi

```
function union(p:R,q:R):R
```

İlişkisel operatörler - UNION

Example

B1	B#	A	T	E	Y
	237AV	Ivanov	Geometry	Narodna prosveta Publishing house	1975
	508CX	Petrov	Informatics	Tehnika Publishing house	1985
	367C1X	Hristov	Pascal	Tehnika Publishing house	1987

B2	B#	A	T	E	Y
	508CX	Petrov	Informatics	Tehnika Publishing house	1985
	674AVI	Angelov	Basic physics	Narodna prosveta Publishing house	1973
	483AVI	Vasilev	Nuclear physics	Nauka i izkustvo Publishing house	1978
	367C1X	Hristov	Paskal	Tehnika Publishing house	1987

B=B1_B2	B#	A	T	E	Y
	237AV	Ivanov	Geometry	Narodna prosveta Publishing house	1975
	508CX	Petrov	Informatics	Tehnika Publishing house	1985
	367C1X	Hristov	pascal	Tehnika Publishing house	1987
	674AVI	Angelov	Basic physics	Narodna prosveta Publishing house	1973
	483AVI	Vasilev	Nuclear physics	Nauka i izkustvo Publishing house	1978

```
B := union(B1,B2)
```

İlişkisel operatörler - FARK

- Yine *p* ve *q*'nun aynı $R(A_1, A_2, \ldots, A_n)$ ilişkisel şemasına sahip iki ilişki olduğu varsayılmaktadır.
- *P* ve *q* ilişkilerinin farkı, *p*'nin *q*'ya ait olmayan elemanlarını içeren aynı ilişkisel şemaya sahip üçüncü bir r ilişkisidir,

$$\text{function difference}(p:R, q:R):R.$$

- İki ilişkinin farkı matematikte yaygın olarak kullanılan "-" işaretiyle gösterilir ve ayrıca aşağıdaki tanımla iki bağımsız değişkenli bir fonksiyon olarak temsil edilir:

$$r = \{t \mid t \in p \ \text{u} \ t \notin q\}.$$

İlişkisel operatörler - FARK

Example

B1	B#	A	T	E	Y
	237AV	Ivanov	Geometry	Narodna prosveta Publishing house	1975
	508CX	Petrov	Informatics	Tehnika Publishing house	1985
	367C1X	Hristov	Pascal	Tehnika Publishing house	1987

B2	B#	A	T	E	Y
	508CX	Petrov	Informatics	Tehnika Publishing house	1985
	674AV1	Angelov	Basic physics	Narodna prosveta Publishing house	1973
	483AV1	Vasilev	Nuclear physics	Nauka i izkustvo Publishing house	1978
	367C1X	Hristov	Pascal	Tehnika Publishing house	1987

B=B1_B2	B#	A	T	E	Y
	237AV	Ivanov	Geometry	Narodna prosveta PH	1975

İlişkisel operatörler - KARTEZYEN ÜRÜN

- *p*'nin $P(A_1, A_2, \ldots, A_m)$, ilişkisel şemasına sahip olduğu ve *q*'nun $Q(B_1, B_2, \ldots, B_n)$ ilişkisel şemasına sahip olduğu *p* ve *q* iki ilişkisi verilsin.
- *P* ve *q*'nun Kartezyen çarpımı, *p* ve elemanlarının tüm olası birleşimlerini içeren $R(A_1, A_2, \ldots, A_m, B_1, B_2, \ldots, B_n)$ ilişkisel şemasına sahip üçüncü $(m+n)$ üye ilişkisi r'dir. *Q*

$$r = \{ab \mid a \in p \cup b \in q\}.$$

- İki ilişkinin Kartezyen çarpımı, matematikte yaygın olarak kullanılan "x" işaretiyle gösterilir, ancak aynı zamanda aşağıdaki tanımla iki bağımsız değişkenli bir fonksiyon olarak da temsil edilir:

```
function product(p:P,q:Q):R
```

İlişkisel operatörler - KARTEZYEN ÜRÜN

Example

B	B#	A	T
	237AV	Ivanov	Geometry
	508CX	Petrov	Informatics
	367CIX	Hristov	Pascal

B_E	B#	S#	Date
	508CX	92	15.06.89
	367CIX	17	08.10.89

BBS=BxB_E	B#	A	T	B#	S#	Date
	237AV	Ivanov	Geometry	508CX	92	15.06.89
	237AV	Ivanov	Geometry	367CIX	17	08.10.89
	508CX	Petrov	Informatics	508CX	92	15.06.89
	508CX	Petrov	Informatics	367CIX	17	08.10.89
	367CIX	Hristov	Pascal	508CX	92	15.06.89
	367CIX	Hristov	Pascal	367CIX	17	08.10.89

BBS := product(B,B_E)

İlişkisel operatörler - Projeksiyon

- $P(A_1, A_2, \ldots, A_n)$ ilişkisel şemasına sahip n üyeli bir ilişki olsun.
- P ilişkisinin $A_{i1}, A_{i2}, \ldots, A_{ik}$, $i<=k<=n$ niteliklerine göre izdüşümü, $R(A_{i1}, A_{i2}, \ldots, A_{ik})$ ilişkisel şemasına sahip k üyeli bir r ilişkisidir ve elde edilir R şemasında yapılandırılmamış ve tekrarlanan öğelerin (satırların) yalnızca bir kez temsil edildiği tüm nitelikleri kaldırarak p'den,

$$r = \{\langle b_1, b_2, \ldots, b_k\rangle \mid (\exists a),\ a=\langle a_1, a_2, \ldots, a_n\rangle,\ a\in p \ \text{ve}\ a_{i1}=b_1,\ a_{i2}=b_2, \ldots, a_{ik}=b_k\}.$$

İlişkisel operatörler - Projeksiyon

- Kabul edilen gösterimlere uygun olarak p ilişkisinin r izdüşümü şu şekilde de yazılabilir:

$$r = p[A_{i1}, A_{i2}, \ldots, A_{ik}].$$

- Bir fonksiyonla ifadesi aşağıdaki tanıma sahiptir:

$$\text{function project}(p:P,\ A_{i1}, A_{i2}, \ldots, A_{ik}:\text{Attr}):R.$$

Co-funded by the Erasmus+ Programme of the European Union

İlişkisel operatörler - Projeksiyon

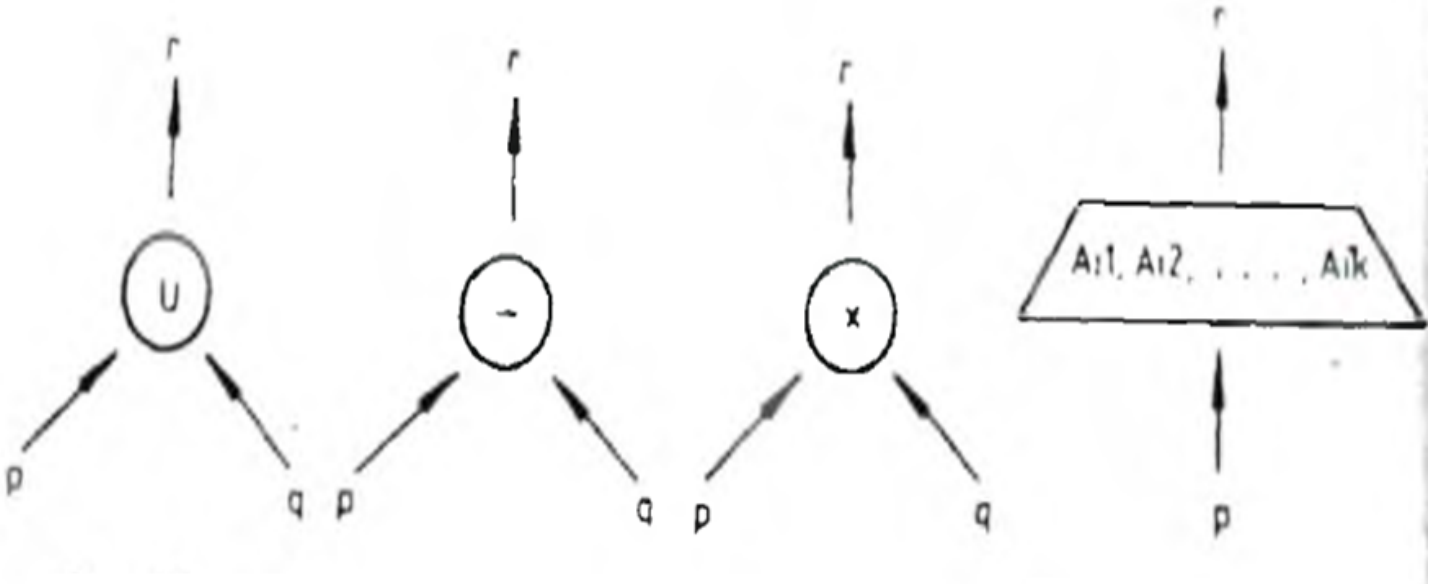
Example

B	B#	A	T	E	Y
	237AV	Ivanov	Geometry	Narodna prosveta	1975
	508CX	Petrov	Informatics	Tehnika	1985
	367CIX	Hristov	Pascal	Tehnika	1987
	792BVI	Petrov	Trygonometry	Narodna prosveta	1985

B1 = $\pi_{A,Y}(B)$	A	Y
	Ivanov	1975
	Petrov	1985
	Hristov	1987

B1 := project(B,A,Y)

GRAFİK GÖSTERGELER



graphic indication of the operation "union"

graphic indication of the operation "difference"

graphical notation of the operation "Cartesian product"

graphical notation of the operation "Projection"

İlişkisel operatörler - KISITLAMA

P, R(A1, A2,..., An) ilişkisel şemasına sahip n üyeli bir ilişki olsun. Belirli bir koşulla ilgili *P* ilişkisi kısıtlaması *F*, aynı ilişkisel şemaya sahip başka bir *r* ilişkisidir ve her bir öğesi *F* koşulunu yerine getirir;

$$r = \{ t \mid t \in p \ \text{u} \ F(t) = \textbf{true} \ \}$$

Kısıtlamanın belirtildiği açıktır $r \subseteq p$.

ile $r = \sigma_F(p)$

Ve bir tonksıyon olarak aşağıdaki tanıma sahiptir:

```
function restriction (p:R, F:Expression) : R.
```

İlişkisel operatörler - BÖLÜM

Karşılıklı ilişkisel şemaya sahip iki *p* ve *q* ilişkisinin bölümü, aynı ilişkisel şemaya sahip üçüncü ilişki *r*'dir, burada:

$$r = \{t \mid t \in p \ u \ t \in q\}$$

Matematikte genel kabul görmüş "∩" işaretiyle gösterilir. Bu bölüm aynı zamanda aşağıdaki tanıma sahip bir fonksiyon olarak da yazılmıştır:

$$\text{function intersect } (p,q:R):R$$

Bölüm ve fark tanımlarına göre aşağıdaki gibidir:

$$r = p \cap q = p - (p-q) = q - (q-p)$$

İlişkisel operatörler - QUOTIENT

İki *p* ve *q* ilişkisi verilsin, *q* - *P(A1,A2,...AK,B1,B2,...,BM)* ilişkisel şemalarıyla boş olmayan ilişki. P ve q'nun bölümü, *R(A1,A2,...,AK)* ilişkisel şemasına sahip üçüncü ilişki *r*'dir ve her *l*ϵ*r* ve her *s*ϵ*q* için; *ts* birleştirmesi bir *p* elemanıdır. İki *p* ve *q* ilişkisinin bölümü *p* : *q* olarak yazılır ve ilişkisel cebirin üç ana işlemiyle aşağıdaki şekilde ifade edilebilir:

$$p : q = \pi_A(p) - \pi_A((\pi_A(p) \times q) - p)$$

$A_1, A_2, \ldots, A_K$ niteliklerinin listesini gösterir.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

İlişkisel operatörler - QUOTIENT

The graphical notation of the quotient of two relations is similar to the other two-argument operations by placing the ":" sign in the circle

Example

B_E	B#	S#
	546AX	72
	328CX	47
	328CX	43
	607BV1	63
	459AV	37

S	S#
	43
	47

B_E : S	B#
	328CX

Çözüm

- *Kullanıcılar, ilişkilere ilişkin işlemleri ifade etmek için resmi notasyonlardan birinde yazılmış sorgular yaparak ilişkisel veritabanındaki bilgileri ararlar. Bu gösterimler iki türdür :*

  1. Sorguların ilişkilere özel işlemler uygulanarak ifade edildiği, ilişkisel cebir adı verilen cebirsel bir gösterim.

  2 Sorguların, sonuçtaki değişkenleri karşılaması gereken mantıksal formüller yazılarak ifade edildiği, ilişkisel analiz adı verilen mantıksal bir gösterim. İki notasyon (bazı kısıtlamalara tabi olarak) eşdeğer ifade gücüne sahiptir, yani. e. her biri diğerinin yapabileceği herhangi bir keyfi sorguyu ifade edebilir.
- *İlişkisel cebir, sorguları ilişkilere açıklamak için kullanılan bir gösterimdir.*
- *İlişkisel cebir işlemleri resmi olarak demet kümeleri olarak ifade edilen ilişkiler üzerinde gerçekleştirilir.*

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. İlişkisel Cebir Nedir?

2. İlişkisel Cebir ile İlişkisel Analizi nasıl ayırt edebilirsiniz?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

❑ Modül 2. Veritabanlarında ilişkisel yaklaşım

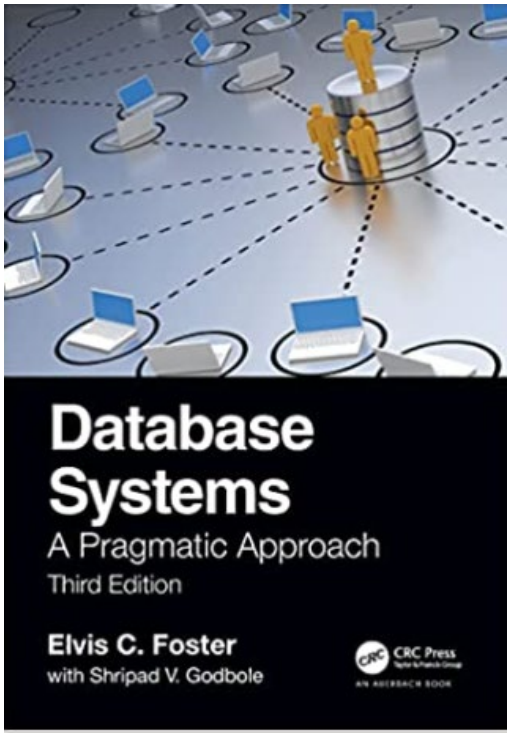
❑ Konu 2. İlişkisel diller.

- ❑ Ders 1. İlişkisel diller. İlişkisel Dil Türleri.

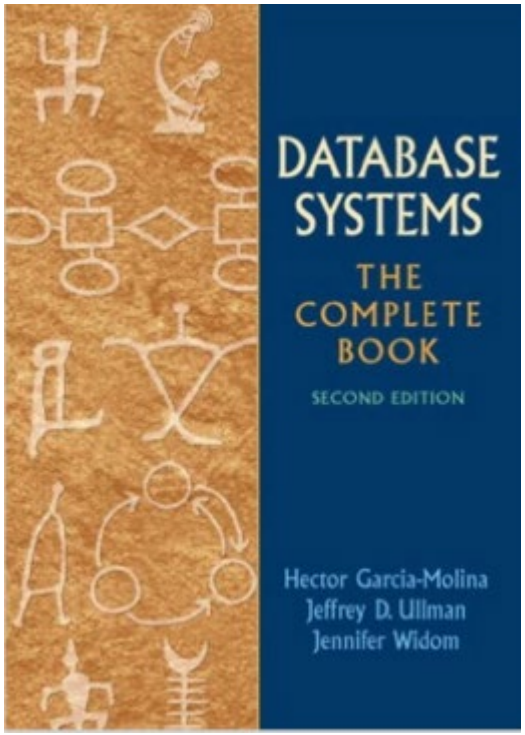
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

İLİŞKİSEL DİLLER. İLİŞKİSEL DİL TÜRLERİ.



Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.



Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

Azalov, P. (1991) Database, Sofia.

Giriş

- İlişkisel diller, ilişkisel DBMS'lerdeki verileri tanımlamaya ve işlemeye yönelik dillerdir.
- Bunlar aynı zamanda veri dilleri olarak da adlandırılır ve çoğunlukla eksik oldukları ve diğer dillerin içine gömülü oldukları için veri alt dilleri olarak da adlandırılırlar.
- İlişkisel diller iki ana kategoriye ayrılır: ilişkisel cebir dilleri ve ilişkisel analiz dilleri.
- İlişkisel cebir dilleri, önceki bölümde tartışılan ilişkisel işlemler aracılığıyla, ilişkisel cevaba giden hesaplama sürecinin adım adım tanımlanmasına izin veren prosedürel dillerdir.

Giriş

- *Yüklem dilleri* olarak da adlandırılan *ilişkisel hesap dilleri*, tek tek işlemleri ve bunların sırasını belirtmeden ilişki-cevabın genel formunu yazmaya olanak sağlar.
- İlişkisel analiz dilleri, dil yapılarında izin verilen değişkenlerin türüne bağlı olarak iki gruba ayrılabilir.
- Birinci tür dillerde, izin verilen değerleri bir ilişkinin öğeleri olan değişkenler kullanılır; e. n-demetler.
- Bu nedenle bunlara değişken n-tuple'larla ilişkisel hesaplama dilleri denir.

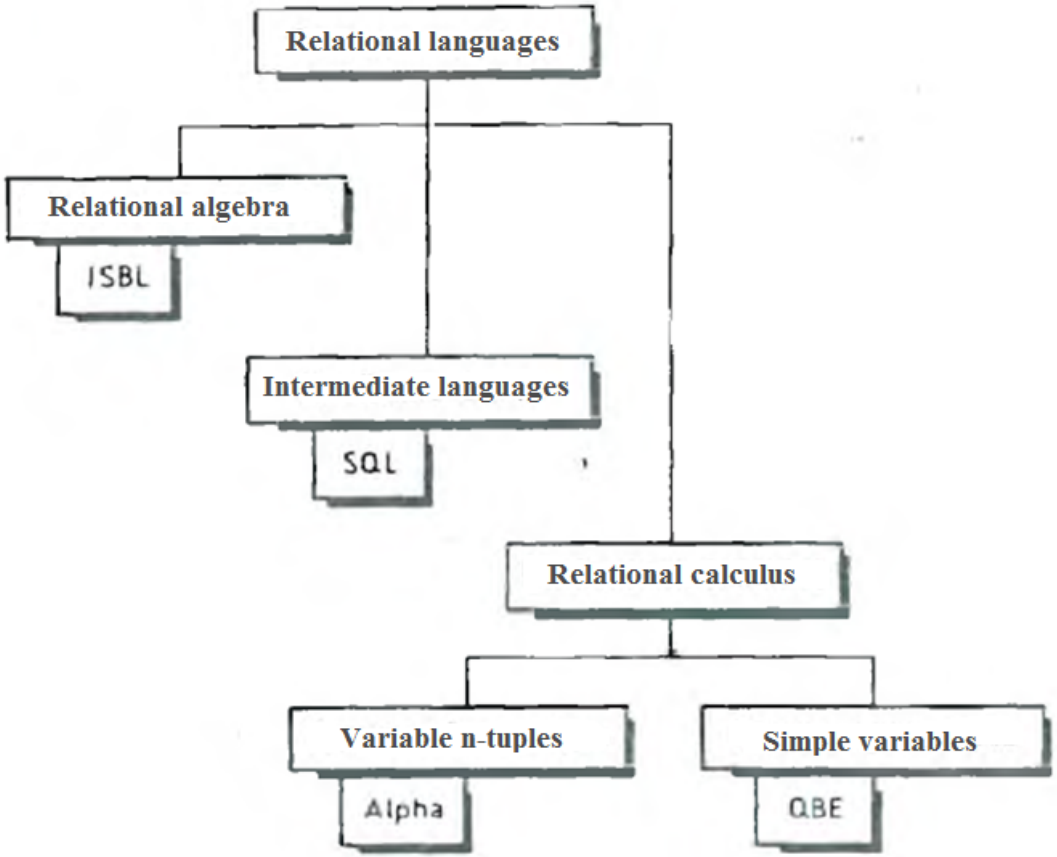
Co-funded by the Erasmus+ Programme of the European Union

Giriş

- Bu türün en yaygın dillerinden biri Alfa dilidir.
- Diğer dil türleri, izin verilen değerleri ayrı alanların öğeleri olan değişkenleri kullanır; basit değişkenler ve bu nedenle basit değişkenlerle ilişkisel hesaplama dilleri olarak adlandırılır.
- Bu tür dillerin temsilcisi QBE dilidir.
- İki ana dil kategorisinin dışında, ilişkisel cebir ve ilişkisel hesap dillerinin yeteneklerine sahip olan sözde ara ilişkisel diller de vardır. Bu türün en çok kullanılan dili SQL dilidir.

Co-funded by the Erasmus+ Programme of the European Union

Giriş



Relational languages categories and their representatives

ALFA DİLİ

- Alfa dili yüklemler hesabına dayanmaktadır ve yazarı E. Codd'dur.
- Dil yapıları azdır ancak güçlü ve anlamlıdır.
- Dilin yazarı, Alpha'nın, araçları veritabanını oluşturan nesneleri tanımlamak için kullanılan PL/1, COBOL veya FORTRAN gibi bazı programlama dillerinde bir veri alt dili olarak gömülü olduğunu varsayar.
- Kullanıcı, dil araçlarını kullanarak veritabanından veri çıkarır ve otomatik olarak çalışma alanına yerleştirilir.
- Veri tabanına veri girildiğinde, öncelikle bazı çalışma alanlarında konumlandırılması gerekir.

ALFA DİLİ

- Başka bir deyişle çalışma alanı, kullanıcı ile veritabanı arasında bir ara bağlantıdır.
- Kullanıcı aynı anda birden fazla çalışma alanını, adlarıyla birlikte alıntı yaparak kullanabilir.
- Her çalışma alanı, bazı ilişkisel operatörlerin yürütülmesi sonucunda elde edilen ilişkinin tipini (şemasını) otomatik olarak alır.
- Veritabanından veri aramak ve almak için kullanılan operatör şu forma sahiptir:

Co-funded by the Erasmus+ Programme of the European Union

ALFA DİLİ

get *W*[(*N*)](*L*)[*KS*]

And the meaning of the symbols used is as follows:

get -	**reserved word of the language**
W -	**working area name**
[...] -	**the expression in these brackets is optional and may be absent**
N -,	**the maximum number of elements in the working area W; by default, it is assumed that W will contain the whole relation - reponse**
L -	**the target list specifying the relation-response scheme; the list can contain attribute names or a relation name**
KS -	**the criterion for selecting data and/or setting the order of the elements of the relation-response**

ALFA DİLİ

- Veritabanı şu durumlarda **<u>güncellenir</u>**:
- - yeni ilişkiler yaratmak veya mevcut ilişkilere öğeler eklemek;
- - DB'den ilişkilerin veya verilen ilişkilerden öğelerin kaldırılması;
- - belirli ilişkilerdeki bazı niteliklerin değerlerinin değiştirilmesi. Bu işlemler bir örnekle gösterilmiştir.

Example **A new element to be added to relation BOOK**

<881 CIX, Ivanov, Database, Tehnika, 1989>

Co-funded by the Erasmus+ Programme of the European Union

ALFA DİLİ

Solution:

First, by means of the used programming language the components of this element are entered in W workspace

```
W.B# := '881CIX';
W.A := ' Ivanov ';
W.T := 'Database';
W.E := ' Tehnika ';
W.Y := 1989 ;
```

The transfer of the element from the workspace W to the DB is done with the operator: put W (BOOK)

ALFA DİLİ

- Alfa dili çeşitli yerleşik özellikler sunar. Get operatörü ile birlikte kullanılırlar ve özellikle hem listenin tamamına hem de seçim kriterine katılabilirler. Daha önemli işlevler arasında şunlar yer almaktadır:

a) count (öznitelik adı) **-** belirtilen öznitelikteki farklı değerlerin sayısını bulur;

b) toplam (öznitelik adı) **-** belirtilen öznitelikteki öğelerin toplamını bulur;

c) top (n, nitelik adı) – bu bir Boolean işlevidir. Değeri, yalnızca "öznitelik adı" parametresinin değeri, o niteliğin n'inci en büyük değerine eşitse "true" olur. Özniteliğin değer olarak {5,3,8,4,6,7,10} sayıları varsa, bu durumda sırasıyla üçüncü en büyük değer 7 sayısı ve üçüncü en küçük değer ise 5 sayısıdır.

d) alt (n, öznitelik adı) – işlev, üst işlevle aynıdır ancak n'inci en küçük değeri ifade eder.

Co-funded by the Erasmus+ Programme of the European Union

QBE DİLİ

- QBE dilinin tam adı Örnekle Sorgulamadır.
- Bu dilin en karakteristik özelliği iki boyutlu sözdizimidir, bu yüzden sadece terminal tarafından kullanılır.
- Kullanıcı, özel bir ekran düzenleyici kullanarak, aslında ilişkisel şemaları tanımlayan boş tablolar (ilişkiler) oluşturur ve ardından kullanır.
- Bazı yazarlar bu kadar boş bir tabloyu "iskelet masa", "masa yapısı" veya "masa şekli" olarak adlandırırlar.

Co-funded by the Erasmus+ Programme of the European Union

QBE DİLİ

- Her form, tablonun adını ve sütunlarının adlarını (özniteliklerini) içerir.

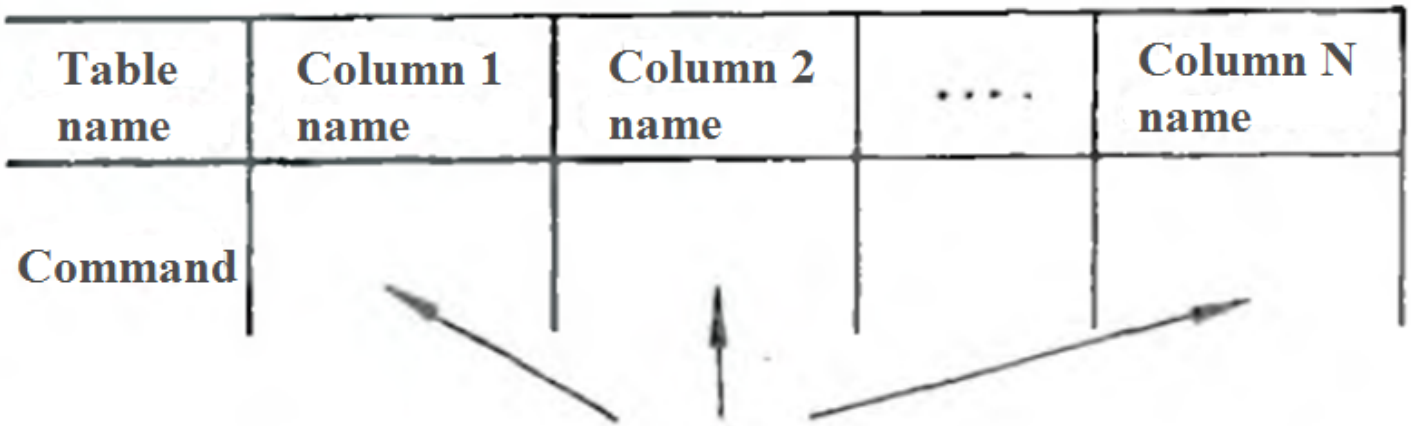


Table form with N-columns

Co-funded by the Erasmus+ Programme of the European Union

QBE DİLİ

- QBE ilişkisel bir veritabanı yönetim dilidir ve veritabanındaki ilişkisel (form) verilerin tanımlanmasını, güncellenmesini ve aranmasını sağlar.
- Bu durumda önemli olan tüm bu tür işlemlerin formlar kullanılarak birleşik bir yöntemle gerçekleştirilmesidir.
- QBE dilindeki ana prensipler ve fikirler şu şekilde özetlenebilir [4, 6, 16, 106]:

1. Kullanıcıya dili kullanırken mümkün olduğu kadar kolaylık sağlanmalıdır.

2. Dilin söz dizimi basit fakat ifade gücü yüksek olmalıdır.

3. DB ile iletişim, formların istenen tüm satırların türünü tanımlayan uygun "örnek" satırla "doldurulması" ile gerçekleştirilir.

QBE DİLİ

- Veritabanından veri arama ve alma, uygun formda örnek bir cevap belirtilerek yapılır. Örnek, formun uygun sütunlarına yerleştirilen değişkenlerin ve sabitlerin adlarını içerir.
- Örneğin bilgisayar bilimlerindeki tüm projelerin numaralarını yazdıracak olsaydınız, bu şu şekilde yapılabilir. PROJ tablosunun adını girin ve form ekranda görünecektir:

QBE DİLİ

- Şimdi sorgunun kendisi hazırlanmalıdır. İki temel unsur içerir:
- - sabit 'bilgi';
- - proje numarasının yazdırılması.
- Bu iki öğe *D* ve *P#* sütunlarına yazılmıştır.

PROJ	P#	S#	D	C
	P.p1		Informatics	

Co-funded by the Erasmus+ Programme of the European Union

QBE DİLİ

- Bu kayıt şu anlama gelir:
- Bilişim alanına ait *p1* gibi proje numaralarını yazdırın.
- *p1* örnek öğesinin altı çizilmiştir ve 'bilgi' metin sabitinin altı çizilmemiştir.
- Proje numaralarının yazdırılması, *p1* örnek elemanından önce gelen *P*. (Yazdır) komutu ile yapılır.
- Örnek eleman "*p1*" oldukça keyfi olarak seçilmiştir ve böyle bir elemanın hiç mevcut olmaması mümkündür.
- İşte bir örnek:

Co-funded by the Erasmus+ Programme of the European Union

QBE DİLİ

Example **Extract publushers which books are used by more than one employee**

Solution:

BOOK	B#	A	T	E	Y
	Z			Tehnika P.	

B_E	B#	S#	D1	D2
	Z	. X		
	Z	≠ X		

Co-funded by the Erasmus+ Programme of the European Union

QBE DİLİ

The following operations are used to update a databse: U - update, I - insert and D - delete.
Here is an example for their implementation

Example

Add a new row in table PROJ:
<p8,s9, matematics ,18x>

Solution:

PROJ	P#	S#	D	C
I.'	p8	s9	matematics	18x

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

QBE DİLİ

Example **Combine tables BOOK and B (suppose that B has the same relation scheme as BOOK)**

Solution:

B	B#	A	T	E	Y
	B1				

BOOK	B#	A	T	E	Y
1.	B1				

Co-funded by the Erasmus+ Programme of the European Union

QBE DİLİ

Example **Reduce the sum for project number p1 by 5 thousand**

Solution:

PROJ	P#	S#	D	C
	p1			X
U.	p1			X-5

Co-funded by the Erasmus+ Programme of the European Union

QBE DİLİ

Veri sözlüğü, bazı sistem tablolarından oluşan açıklamalar kümesinden oluşur. Bunlardan üçü burada açıklanmaktadır:

TABLO – DB tabloları hakkında bilgi içeren bir tablo;

SÜTUNLAR – DB öznitelikleri hakkında bilgi içeren bir tablo;

YETKİLİ – kullanıcılar ve onların farklı ilişkilere erişim hakları hakkında bilgi içeren bir tablo. Erişim türü ayrıntılı olarak açıklanmaktadır: DB'den veri çıkarmak, güncellemek, eklemek veya silmek için.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

QBE DİLİ

A new table is defined by opertaion I (Add)

Example **Create table PROJ.**

Solution: **Begins with an empty table of the type:**

Co-funded by the Erasmus+ Programme of the European Union

QBE DİLİ

If neccessary new columns are added automatically and some of them can be expanded:

I.PROJ I.	P#	S#	D	.C

Operation P is used to enter characteristics in each column:

I.PROJ I.	P#	S#	D	C
P.X				

Co-funded by the Erasmus+ Programme of the European Union

QBE DİLİ

The result of the operation is

PROJ		P#	S#	D	C
TYPE	I.	string	string	string	integer
LENGTH	I.	3	4	16	4
KEY	I.	Y	N	N	N
DOMAIN	I.	NUMBER	NUMBER	SCIENCE	MONEY
SYSNULL	I.		-	-	0

The meaning of each characteristic is:

TYPE - **defines the type of data in the column:**
char, string, real, integer, boolean;

LENGTH - **defines the table field lenght (in number of positions)**

KEY - **specifies the key (Y) and non-key (N) attributes in the table**

DOMAIN - **contains the fields names which give values to the attributes**

SYSNULL - **defines the zero value to each attribute**

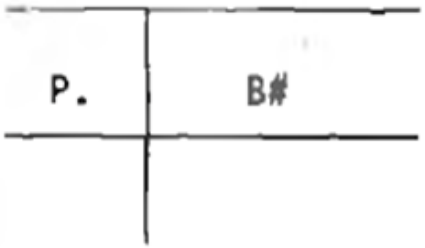
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

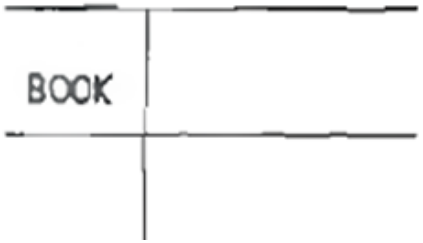
QBE DİLİ

Example **Get the names of the tables containing columns named B#**

Solution:



The answer is:



QBE DİLİ

- Microsoft Access ve Microsoft SQL Server Enterprise Manager gibi ürünler QBE'nin özelliklerini kullanır.
- İlişkisel veritabanlarının popüler hale gelmesiyle veri işlemleri için standart bir dile ihtiyaç duyuldu. Cevap SQL'di (Yapılandırılmış Sorgu Dili).
- Yavaş yavaş SQL, aşağıdakiler için kontrol ifadeleri içeren çok amaçlı bir veritabanı diline dönüştü: veri oluşturma, değiştirme ve silme; veri tanımı (tablolar, sütunlar); gruplarla ve bireysel kullanıcılarla çalışarak veritabanı öğelerine erişimi korumak; yedekleme, blok kopyalama ve güncelleme gibi veri yönetimi işlemleri; ve en önemlisi işlem işleme.

SQL dili bir sonraki başlıkta detaylı olarak ele alınacaktır.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. İlişkisel diller nelerdir?

2. İlişkisel Model hangi dili kullanıyor?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

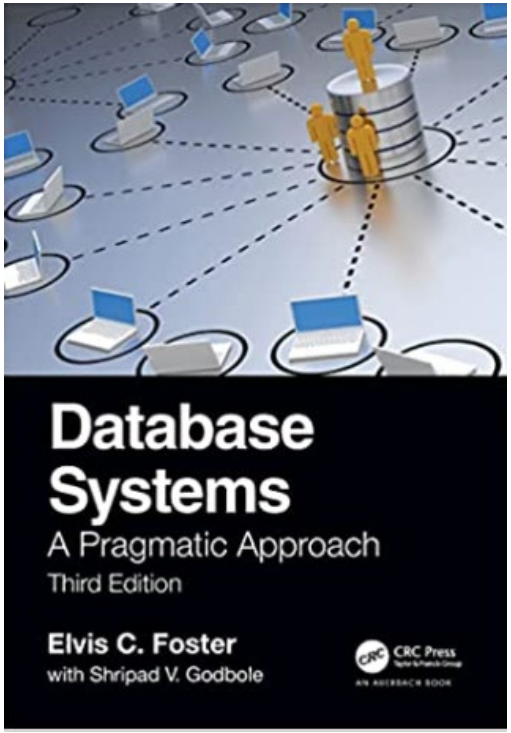
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

- **Modül 2. Veritabanlarında ilişkisel yaklaşım**
  - **Konu 2. İlişkisel diller.**
    - Ders 2. SQL İlişkisel dil. Veri seçimi. Yerleşik işlevler. Veri güncelleme.

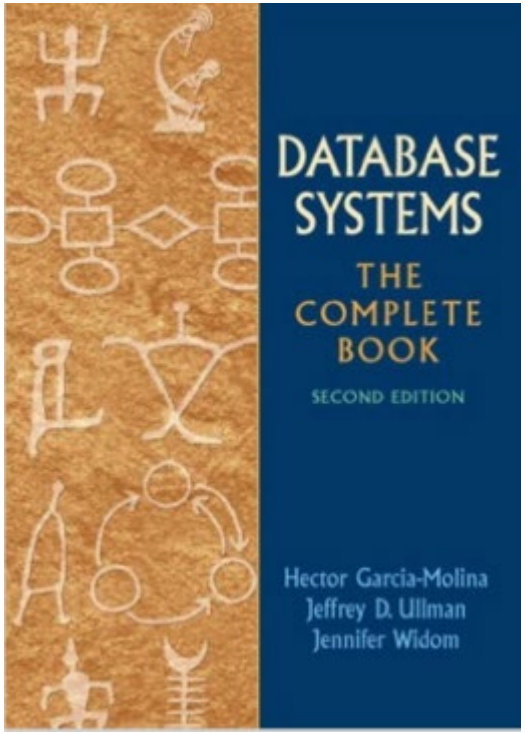
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

SQL İLİŞKİSEL DİL. VERİ SEÇİMİ. YERLEŞİK FONKSİYONLAR. VERİ GÜNCELLEME



Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.



Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

Azalov, P. (1991) Database, Sofia.

Co-funded by the Erasmus+ Programme of the European Union

Giriş

- SQL Dili (Yapılandırılmış Sorgu Dili) ve QBE, ilişkisel veritabanlarında düzenlenen verileri tanımlamak ve işlemek için birleşik bir yaklaşım sağlar.
- 1974 yılında, aralarında D. Chamberlin'in de bulunduğu IBM ortakları tarafından geliştirildi.
- SEQUEL, daha sonra SEQUEL2 olarak bilinen ilk biçiminde SQL dili, System R, SQL/DS, DB2 ve Oracle gibi çeşitli ilişkisel sistemlerde kullanıldı.
- SQL operatörleri doğrudan modda gerçekleştirilebilir veya PL/1, COBOL, FORTRAN, PASCAL, C vb. gibi üst düzey dil programlarına yerleştirilebilir.

Co-funded by the Erasmus+ Programme of the European Union

Giriş

- **SQL** dili, operatörlerinin istenen sonucu elde etme yolunu belirtmeden neyin elde edileceğini belirtmesi nedeniyle prosedürel olmayan bir dildir.
- Dilin anahtar kelimeleri saklıdır ve başka nesneleri (tablolar, sütunlar vb.) belirtmek için kullanılamaz.
- Açıklık ve kısalık sağlamak için, daha karmaşık SQL operatörlerinin sözdizimi, sözdizimi diyagramlarıyla grafiksel olarak temsil edilir.
- Bir sözdizimsel diyagramın temel yaratıcı unsurları her zamanki gibi oval, daire, dikdörtgen ve oktur:

Giriş

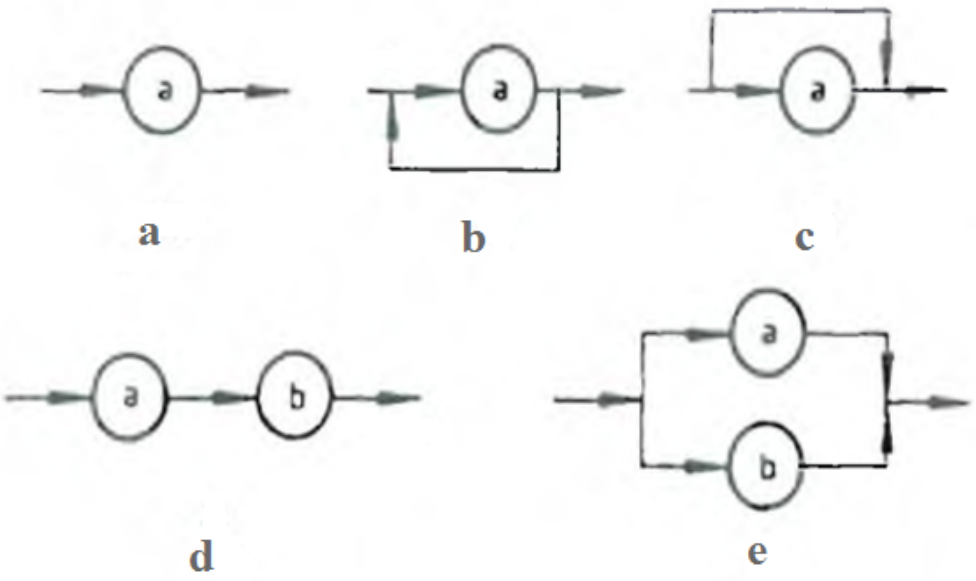
Basic geometric objects used in syntactic diagrams

- SQL'e ayrılmış sözcükleri belirtmek için bir oval kullanılır. Daire, oval ile aynı amaca sahiptir ancak genellikle daha az karakter içeren sözcükleri içerir.
- Dikdörtgen, muhtemelen başka bir sözdizimsel diyagramda tanımlanmış olan terminal olmayan karakterleri belirtmek için kullanılır.

Co-funded by the Erasmus+ Programme of the European Union

Giriş

- Ok, sözdizimsel diyagramın "hareket" (okuma) yönünü gösterir.

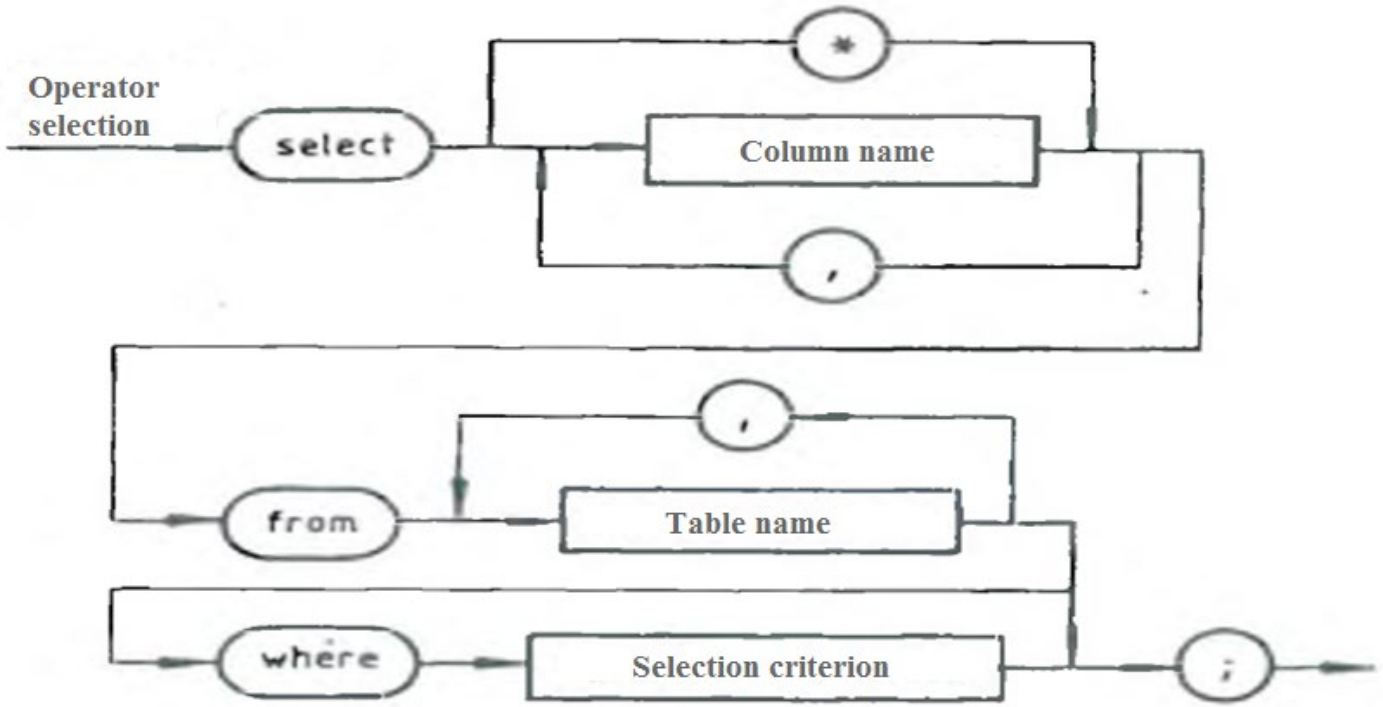


a – a character string consisting of only the letter a is defined;
b – a character string consisting of one or more characters: a, aa, aaa, aaaa, ...;
c – a character string consisting of a single character a or is the blank string;
d – a character string ab is defined;
e – a character string which is either a or b is defined

Her sözdizimsel diyagramın, tanımlanan kavramın yazıldığı bir çıkışı ve bir girişi vardır. Sözdizimsel diyagram boyunca "hareket" okların yönüne göredir.

VERİ SEÇİMİ

- Veritabanından veri seçimi select operatörü ile yapılır. Çoğunlukla eksik haliyle kullanılır:

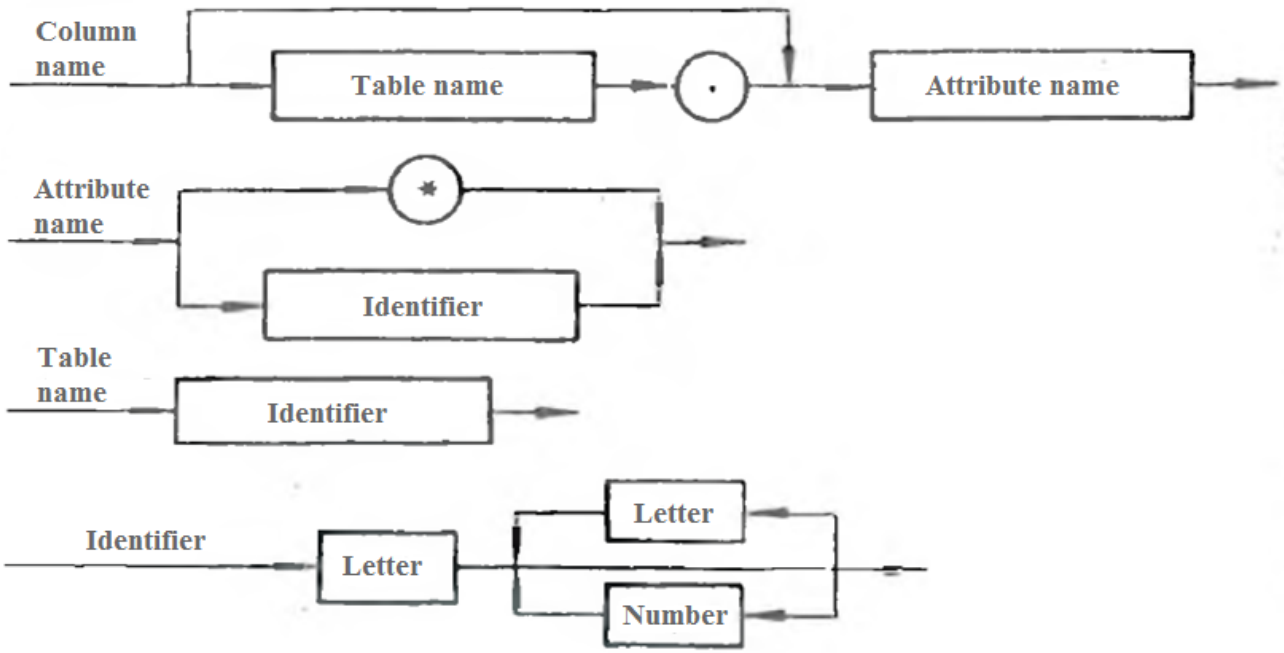


Incomplete format of selection operator

VERİ SEÇİMİ

- İçindeki anahtar kelimeler :
  - **Seçme;**
  - **İtibaren;**
  - **Nerede**.
- Sütun ve tablo adları için tanımlayıcılar (bir harfle başlayan harf ve rakamlardan oluşan bir karakter dizisi) kullanılır.
- Diğer dillerde olduğu gibi, gerektiğinde sütun adlarının başına karşılık gelen tablonun adı eklenir.

VERİ SEÇİMİ



Definition of the main objects (names) used in SQL queries

VERİ SEÇİMİ

Örnek Kitapların imza numaralarını ve konularını alın

Çözüm: B#, T'yi seçin
KİTAP'tan;

Örnek Kitapların imza numaralarını ve konularını alın,
1989'dan sonra “Tehnika” Yayınevi tarafından basılmıştır.

Çözüm: B#,T'yi seçin
KİTAP'tan
burada (E='Tehnika') ve (Y>1989)

VERİ SEÇİMİ

- Seçim kriteri, daha basit durumda ilişkilere yönelik işlemleri içerebilen bir yüklemdir:

  =, <>, >, <, >=, <=

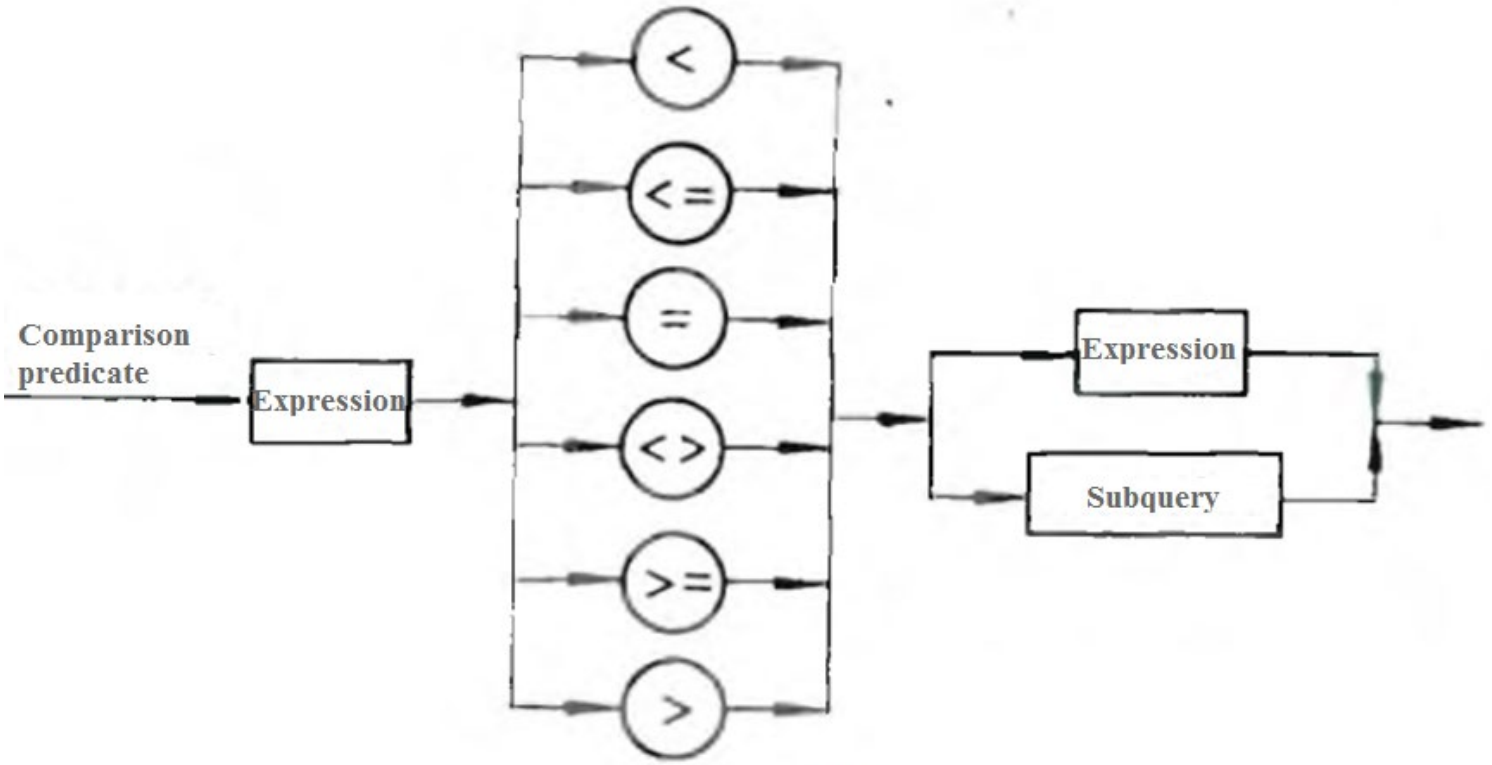
  boole işlemleri:

  ve, veya, değil

  ve küçük braket.

- Seçim kriterini ifade eden koşul, ifadelerin değerlerini karşılaştırmak için bir yüklemdir:

VERİ SEÇİMİ



Comparison predicate definition

VERİ SEÇİMİ

- İfade olarak sütun adları, sabitler (sayısal ve metin), yerleşik işlevlere yönelik sorgulamalar (count, avr, max, min, sum) ve aritmetik işlemler olarak dört temel aritmetik işlem (+, -, *, /).

Örnek “Tehnika” Yayınevi'nde 1980'den 1990'a kadar yayınlanan kitaplar hakkında detaylı bilgi edinin.

Çözüm:

```
seçme *
KİTAP'tan
burada (E='Tehnika) ve (Y>1980)
 ve (Y<=1990)
```

Co-funded by the Erasmus+ Programme of the European Union

VERİ SEÇİMİ

Example “Tehnika” tarafından yayınlanan kitapların tüm çalışanlarının ve yazarlarının isim listesini edinin.

Çözüm: Bu örneğin çözümünde iki değerin birliği

BOOK.A ve EMPLOYEE.N'nin karşılaştırılabilir nitelikleri gereklidir.
Bu, sendika operatörü tarafından gerçekleştirilir.

A'yı seç
KİTAP'tan
burada E='Teknik'
birlik
N'yi seç
ÇALIŞAN'dan

VERİ SEÇİMİ

Örnek 1985 yılına kadar "Teknika"da yayınlanmayan çalışanların listesini edinin.

Çözüm: Cevap, 1985 yılına kadar "Tehnika"da yayın yapan tüm çalışanların kalabalıktan çıkarılmasını, yani eksi işlemini gerçekleştirmesini gerektirir.

N'yi seç
ÇALIŞAN'dan
eksi
A'yı seç
KİTAP'tan
burada (E='Tehnika') ve (Y<= 1985)

VERİ SEÇİMİ

- Aşağıdaki örnekler, seçim kriterinin şu ana kadar dikkate alınanlardan önemli ölçüde daha karmaşık ve farklı olduğu bazı olasılıkları vurgulamaktadır.

Örnek Kütüphaneden kitap alan çalışanların isimlerini alın.

Solution: Bu sorunun cevabını alabilmek için EMPLOYEE ve B_E tablolarına ihtiyaç vardır. Kütüphaneden kitap kullanan çalışan sayısının (s1, s2, s15) olduğunu varsayalım. Sorusunun cevabı şu şekilde olacaktır:

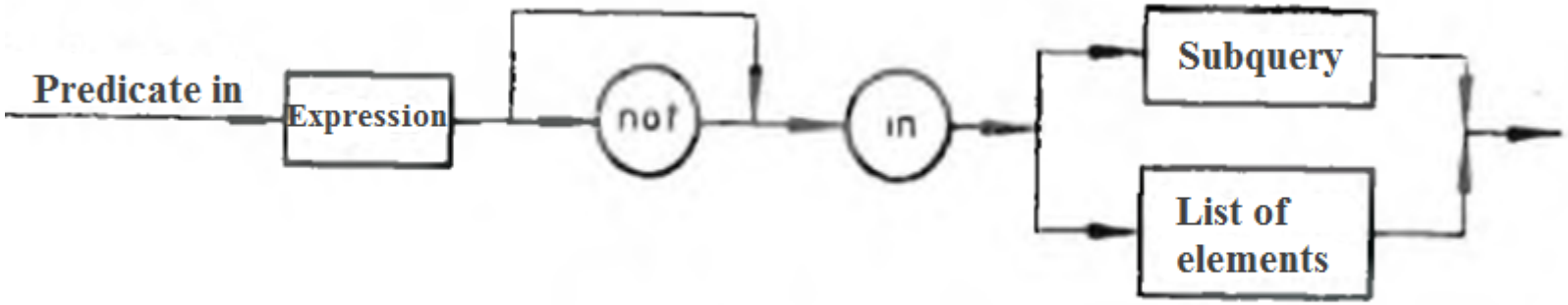
ÇALIŞAN.N'yi seçin

ÇALIŞAN'dan

burada ÇALIŞANLAR.S# (s1, s2, s15)

VERİ SEÇİMİ

- Üyelik yüklemi, S# çalışan numarasının verilen kümeye üyeliğini kontrol eder.
- Yüklemin genel biçimi sözdizimsel diyagramla verilir:



Definition of membership predicate to the set

VERİ SEÇİMİ

- Kitap kullanan çalışan sayıları genel olarak bilinmediğinden çalışan sayıları (sl,s2,s15) listesinin kaydedilmesi aslında imkansızdır.
- Bu nedenle bu listenin yukarıdaki sorgunun alt sorgusu olan başka bir sorguyla elde edilmesi gerekir:

ÇALIŞAN.N'yi seçin

ÇALIŞAN'dan

burada ÇALIŞAN.S#

(B_E.S#'yi seçin

B_E'den);

Co-funded by the Erasmus+ Programme of the European Union

VERİ SEÇİMİ

Örnek p1 projesinin yöneticisi olmayan çalışanların isimlerini alın

Çözüm:

ÇALIŞAN seç
ÇALIŞAN'dan
p1'in olmadığı yerde
(PROJ.P#'yi seçin
PROJ'dan
burada ÇALIŞANLAR#=PROJ.S#)

Yerleşik işlevler

- Diğer ilişkisel diller gibi SQL de çeşitli yerleşik işlevler kullanır.
- Bu işlevler, seçim kriterinin yanı sıra kelime seçiminin ardından gelen öğeler listesinde de ele alınabilir.
- Aşağıda yerleşik işlevlerin temel kullanım durumlarını gösteren bazı örnekler verilmiştir:
  - **Saymak;**
  - **Toplam;**
  - **Min.**

Yerleşik işlevler

Örnek Tüm projelerin sayısını alın.
Çözüm: sayımı seçin (P#)
PROJ'dan;

Örnek Proje yöneticilerinin sayısını öğrenin.
Çözüm: Bir çalışan birden fazla projenin yöneticisi olabildiği için PROJ tablosundaki yalnızca farklı çalışan sayıları sayılmalıdır.

sayımı seçin (farklı PROJ.P#)
PROJ'dan

Co-funded by the Erasmus+ Programme of the European Union

Yerleşik işlevler

Örnek Farklı konulardan (bilişim, matematik vb.) projelerin toplam tutarını alın.

Çözüm: Bu, örnek 18'dekinden daha genel bir görevdir. Bu durumda projeler araştırma konularına göre gruplandırılmalı ve her bir proje grubu için toplam tutar sayılmalıdır. Bu gruplandırma şu ifade grubuyla gösterilir:

PROJ.P#'yi seçin, toplam(PROJ.C)
PROJ'dan
PROJ.D grubu

VERİ GÜNCELLEME

- Veritabanını güncellemek için **güncelleme** (değerleri değiştirme), **ekleme** (satır ekleme) ve **silme** (satırları ve tabloları kaldırma) operatörleri kullanılır.
- Somut örnekler aracılığıyla her birinin ana işlevleri ve yetenekleri özetlenmiştir.

Example 24 **Double the sum for project number 'p1'**

Solution:

```
update PROJ
set C=2*C
where PROJ.P#='p1'
```

- **Güncelleme** operatörünün **set** kısmında birden fazla sütunun değerlerinde değişiklik yapılabilir. Bu amaçla birbirinden virgülle ayrılan atama operatörleri kullanılır.

VERİ GÜNCELLEME

- Seçim kriteri elbette çok daha karmaşık olabilir ve bu nedenle seçilen tablo öğeleri alt kümesinin değerlerini güncelleyebilir.

Example 25 **Add a new row <p7,126,informatics,19 thousand> in table PROJ**

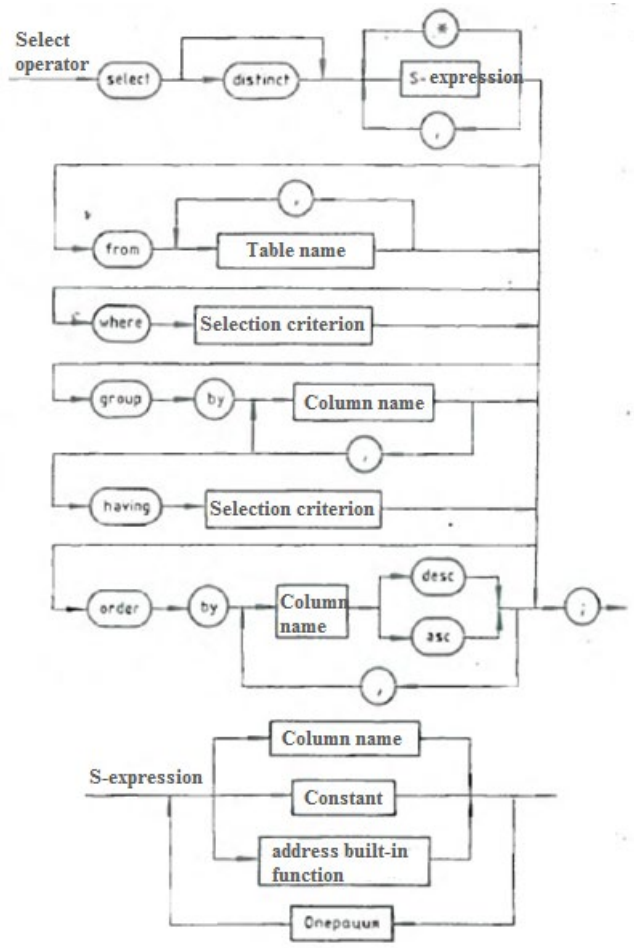
Solution:

```
insert
into PROJ
values ('p7',126,' informatics ',19)
```

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

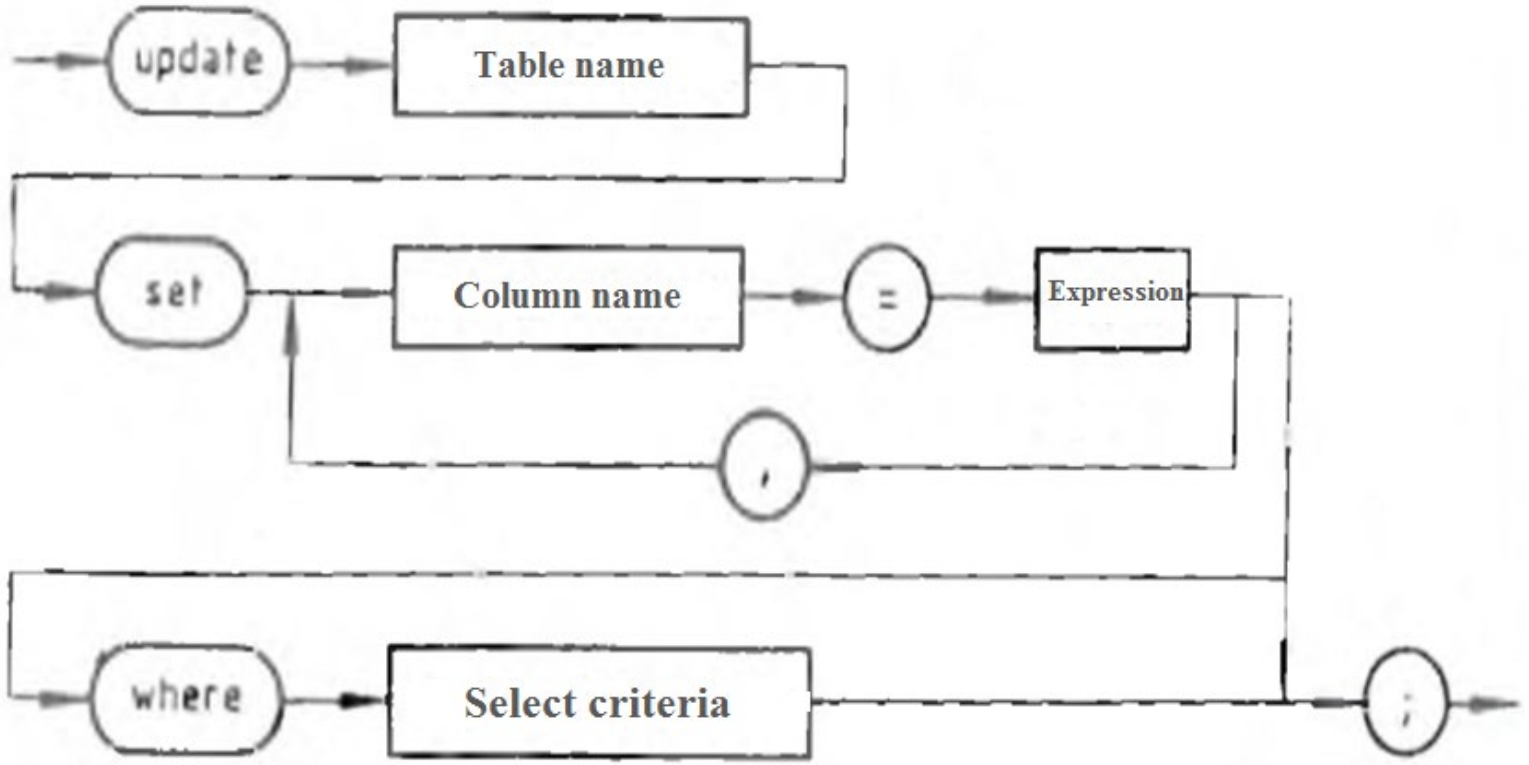
VERİ GÜNCELLEME



Full format of the select operator

Co-funded by the Erasmus+ Programme of the European Union

VERİ GÜNCELLEME



General type of change operator

Co-funded by the Erasmus+ Programme of the European Union

VERİ GÜNCELLEME

- Tabloya yeni bir satır eklerken anahtar niteliklerin değerlerinin belirtilmesi zorunludur. Bu, diğer özellikler için zorunlu değildir, örneğin:

sokmak

PROJ'e(P#,D,S#)

değerler ('p7','информатика',126)

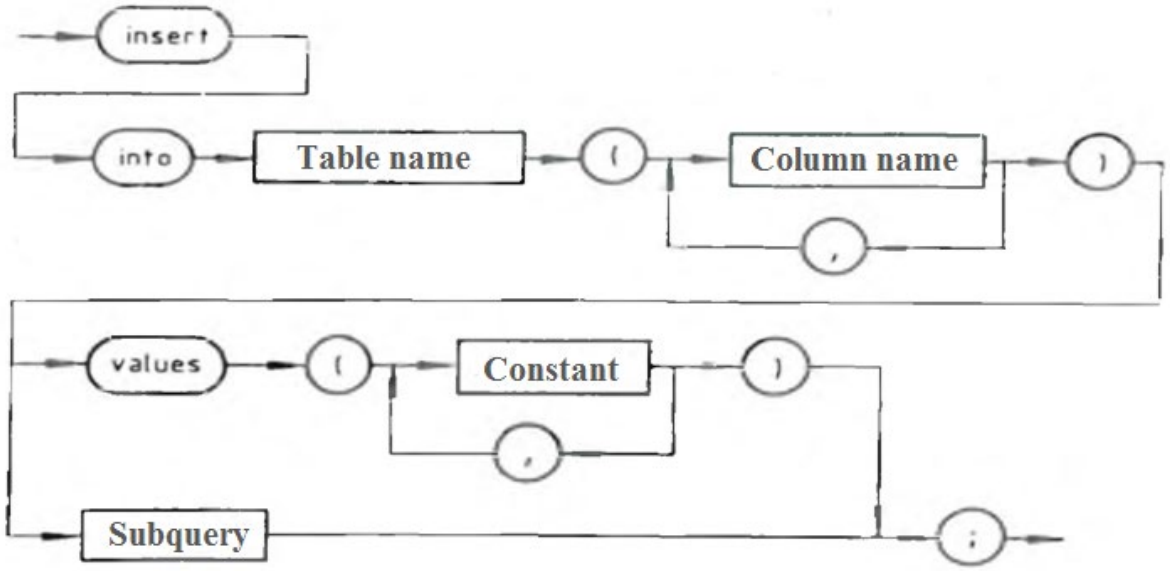
Örnek PROJ tablosundaki bilişim projelerinin yöneticisi olan tüm çalışan sayıları EMPLOYEE tablosuna eklenecektir.

Çözüm :

sokmak
ÇALIŞAN'a (S#)
PROJ.S#'yi seçin
PROJ'dan
burada D='bilgi'

VERİ GÜNCELLEME

- Bu operatör uygulandığında PROJ tablosundan giriş otomatik olarak yapılır ve bu **seçim** operatörü ile ayarlanır.
- Veri eklemeye ilişkin iki örnek, **ekleme** operatörünün iki çeşidini temsil eder.



General type of the operator for adding rows to a table

VERİ GÜNCELLEME

Örnek 1970 yılında iade edilen kitaplarla ilgili tüm satırları B_E tablosundan Silin

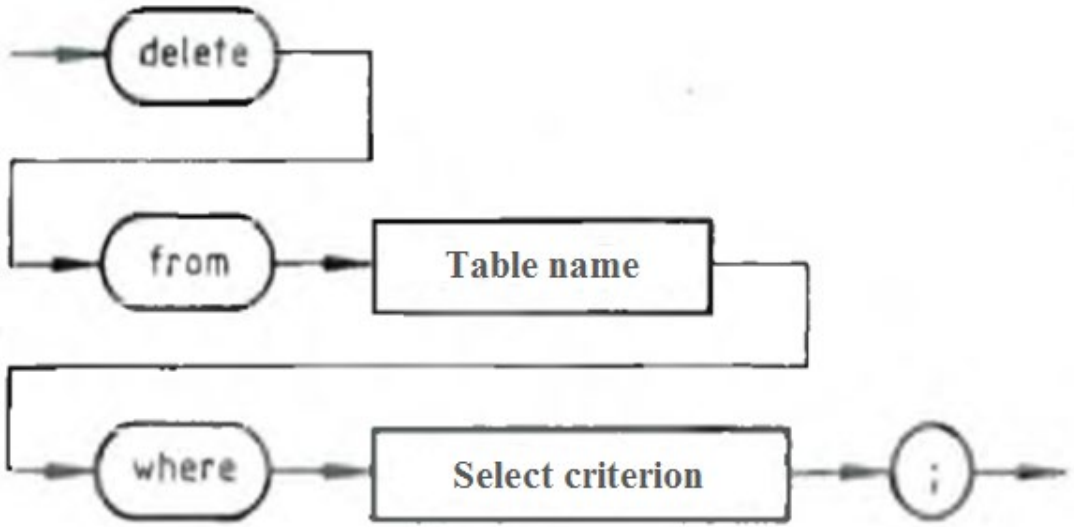
Çözüm: silmek
B_E'den
burada B_.D2 %1970 gibi;

Örnek B_E tablosundaki tüm satırları sil

Çözüm : silmek
B_E'den

VERİ GÜNCELLEME

- Silme operatörünün genel görünümü aşağıdaki şekilde gösterilmiştir:



Operator for deleteing new rows of a table

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. SQL nedir?
2. SQL'deki Kısıtlamalar Nelerdir?
3. "Kütüphane" adında bir veritabanı oluşturmak için doğru SQL komutu nedir?
   - ✓ veritabanı kütüphanesi oluştur
   - ✓ yeni veritabanı kitaplığı oluştur
   - ✓ veritabanı kütüphanesi oluştur (okuyucular, dosyalar, kitaplar)
   - ✓ yeni veritabanı kitaplığı oluştur (okuyucular, dosyalar, kitaplar)

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkürler!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

❑ Modül 2. Veritabanlarında ilişkisel yaklaşım

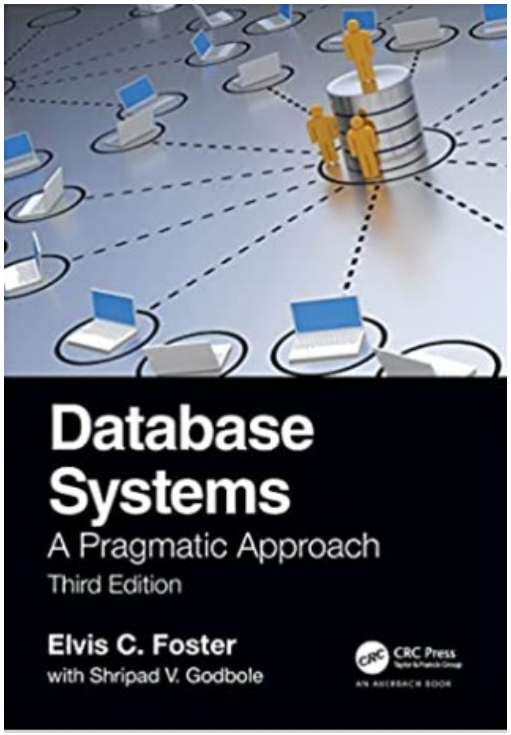
❑ Konu 3. İlişkisel sistemler.

- ❑ Ders 1. İlişkisel sistemlerin temel özellikleri ve sınıflandırılması

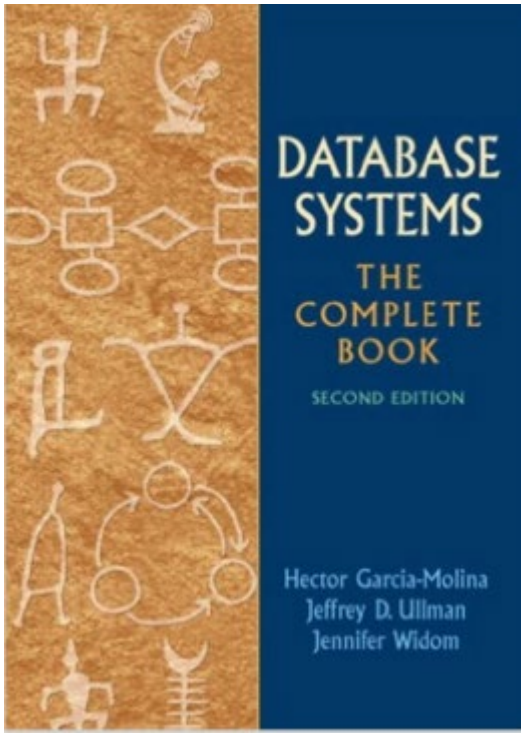
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

İLİŞKİSEL SİSTEMLERİN TEMEL ÖZELLİKLERİ VE SINIFLANDIRILMASI.



Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.



Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

Azalov, P. (1991) Database, Sofia.

Co-funded by the Erasmus+ Programme of the European Union

Giriş

- Modern Veritabanı Yönetim Sistemi (DBMS)'ler, programcıların bilgi işlem sistemlerini tasarlama ve geliştirmedeki üretkenliğini artırmak ve son kullanıcının belirli programlama sistemlerini oluşturma, değiştirme, eşlik etme ve çalıştırma rolünü artırmak için gerekli bir programlama aracı olarak ortaya çıktı.

- Birçok uzmanın görüşüne göre, herhangi bir Veritabanı Yönetim Sistemi (DBMS)'ler 'nin gereksinimleri, ilişkisel veri modelini kullanan Veritabanı Yönetim Sistemi (DBMS)'ler 'ler tarafından en fazla karşılanır.

İlişkisel sistemlerin temel özellikleri.

- Bu sistemlere ***ilişkisel veritabanı yönetim sistemleri*** veya kısaca ***RDBMS*** adı verilmektedir.
- İlişkisel veri modeli üzerine kurulu RDBMS'nin temel avantajları şunlardır:

- Basitlik. Verilerin mantıksal temsilinin ve aralarındaki ilişkilerin homojenliği ile ifade edilir.

- Veri bağımsızlığı. İlişkisel şemalar ve ilişkisel diller, veri temsil yöntemlerinden ve veri erişim yöntemlerinden bağımsızdır.

- Sorguları optimize etme yeteneği.

İlişkisel sistemlerin temel özellikleri.

- ***Üst düzey arayüz.*** Prosedürel olmayan ilişkisel diller aracılığıyla son kullanıcı, sorgularını veritabanına açıklama yeteneğine sahiptir.

- ***Etkili fiziksel sunum ve arama yeteneği***

- ***Sanal nesnelerle çalışın.*** Kullanıcı, tanımlanmış ilişkilere dayanarak sanal, fiziksel olarak var olmayan ilişkileri tanımlama yeteneğine sahiptir..

- ***Dağıtılmış işleme yeteneği.***

- ***Güçlü teorik temel.*** N-üyeli ilişkisellik, küme teorisi ve birinci dereceden yüklem hesabının matematiksel konseptine dayanmaktadır.

İlişkisel sistemler sınıflandırması

- İlişkisel sistemleri sınıflandırırken en alt düzey ***minimal ilişkisel sistemler*** olarak adlandırılanlardır..
- Bunlar yalnızca ilişkisel yapıları destekleyen, bütünlük kısıtlamalarını desteklemeyen sistemlerdir ve bunlara karşılık gelen DML (Veri işleme dili) ilişkisel olarak tamamlanmamıştır.
- Bir sonraki seviyede, ilişkisel yapılar dışında, ilişkisel olarak tam DML'yi destekleyen, ancak boş değer işlemesi olmayan, ***ilişkisel olarak eksiksiz sistemler*** vardır..
- Hiyerarşinin en üstünde ***tamamen ilişkisel sistemler*** bulunur. İlişkisel yapıları desteklerler, karşılık gelen DML'leri ilişkisel olarak tamamlanır ve bütünlük kısıtlamalarını desteklerler

İlişkisel sistemler sınıflandırması

- Tamamen ilişkisel sistemler E. Codd tarafından ayrıntılı olarak açıklanmaktadır. Bir sistemin tamamen ilişkisel olmasını tanımlayan gereksinimler aşağıda kısaca listelenmiştir:
- 1. RDBMS'deki her bir bilgi, tablo formatında sunulacaktır. (DB'deki verilere ve aralarındaki ilişkilere ek olarak, verilerin tablo halinde açıklamaları, bütünlük kısıtlamaları vb.)
- 2. Veritabanındaki her veri öğesine bir tablo adı, bir birincil anahtar değeri ve bir sütun adı ile erişilebilir.
- 3. Boş (anahtar olmayan) değerleri işleyebilme.

İlişkisel sistemler sınıflandırması

4. Sistem kataloğu, sistemde mevcut olan ilişkisel dil ile işlenen bir dizi tablodur.

5. İlişkisel dil şunları içermelidir:

- sanal tabloların tanımı da dahil olmak üzere veri tanımlama dili;
- veri işleme dili;
- bütünlük kısıtlamalarını tanımlamak için kullanılan dil.

6. Sanal tabloları güncelleyebilme.

7. Tüm tabloları güncelleyebilme.

İlişkisel sistemler sınıflandırması

8. Fiziksel veri bağımsızlığı.

9. Mantıksal bağımsızlık.

10. Dürüstlüğün "Bağımsızlığı":

- bütünlük kısıtlamaları kullanıcı programı içinde değil ilişkisel dil aracılığıyla tanımlanır;

- Bütünlük kısıtlamaları kümesi en az iki ana yapısal bütünlük kısıtlamasını içermelidir.

11. Merkezi ve dağıtılmış işleme yeteneği.

Co-funded by the Erasmus+ Programme of the European Union

İlişkisel sistemler sınıflandırması

12. Sistemde düşük seviyeli bir dil de mevcutsa, yüksek seviyeli ilişkisel dilin getirdiği bütünlük kısıtlamalarının ihlal edilmesine izin vermemelidir.

Gördüğünüz gibi ilişkisel sistemlere yönelik gereksinimler son derece ciddi. Bu, yazılım pazarında on iki gereksinimin tamamını karşılayan sistemlerin bulunmasının nadir olduğu gerçeğini tam olarak açıklamaktadır, ancak geliştiricilerin sistemlerini tamamen ilişkisel sistemlere mümkün olduğunca yakın hale getirmeye çalışmasının nedeni budur.

Bazı temel ilişkisel sistemlere genel bakış

- Burada, çeşitli dilsel araçlarla ilişkisel yaklaşımı farklı şekillerde yansıtan çeşitli ilişkisel sistemler seçilmiştir.

INGRES SİSTEMİ

- INGRES (Etkileşimli Grafik ve Erişim Sistemi), Berkeley'deki Kaliforniya Üniversitesi tarafından geliştirilen ilişkisel bir sistemdir.
- INGRES'teki ana başarılar, yüksek veri bağımsızlığı ve açıklama, arama, veri güncelleme ve bütünlük kısıtlamalarını korumak için prosedürel olmayan bir dilin kullanılabilirliğidir. Bu anlamda tamamen ilişkisel sistemlere çok yaklaşmaktadır.

Bazı temel ilişkisel sistemlere genel bakış

- INGRES'te dört arayüz desteklenir:
- - QUEL - n-tuples değişkenli ilişkisel hesaplamaya dayanan ilişkisel olarak eksiksiz bir sorgulama dili;
- - EQUEL - QUEL operatörlerinin C dili programlarına dahil edilebildiği bir arayüz;
- - CUPID - arama ve güncelleme yeteneklerine sahip resimler aracılığıyla prosedürel olmayan bir arayüz;
- - GEO-QUEL - veritabanındaki ilgili verileri coğrafi haritalar şeklinde görüntülemek için kullanılan bir arayüz.

Bazı temel ilişkisel sistemlere genel bakış

- QUEL dilinin seçici olasılıkları, Alfa dilinin get operatörüne benzer bir operatör tarafından ancak niceleyiciler olmadan ifade edilir:

```
range of t1 is R1
range of t2 is R2
...
retrieve (t1.A, t2.B, ...) where C
```

- Burada tl ve t2, modifikasyon alanları sırasıyla R1 ve R2 olan değişken n-tuple'ların adlarıdır; A ve B, R1 ve R2'deki öznitelik adlarıdır ve C, DB'den veri seçiminin gerçekleştirildiği bir durumdur.

QUEL, sanal tabloların ve dizinlerin tanımlanmasının yanı sıra karma yoluyla erişime de olanak tanır.

Bazı temel ilişkisel sistemlere genel bakış

ORACLE SİSTEMİ

ORACLE tamamen ilişkisel bir sistem olup Relational Software Inc. şirketi tarafından geliştirilmiştir. İlk sürümü 1978 yılına aittir. Sistemde kullanılan dil SQL olup, doğrudan veya PASCAL, C, PL/1 programları aracılığıyla çalışılabilmektedir. , COBOL, FORTRAN, SQL çağrılarını içerir.

ORACLE, mikro bilgisayarlar ve çeşitli işletim sistemlerini çalıştıran büyük bilgisayarlar için kullanılır.

Sistemde aşağıdaki seçeneklere sahip bir rapor oluşturucu oluşturulur:

Bazı temel ilişkisel sistemlere genel bakış

- etkileşimli uygulamaların geliştirilmesi;
- raporların çıktısı;
- metinleri düzenlemek

ORACLE, sanal tabloların tanımlanması ve işlenmesinin yanı sıra veritabanında veri bütünlüğünü sağlamak için gelişmiş olanaklara sahiptir.

PASCAL/R SİSTEMİ

PASCAL/R, PASCAL algoritmik dilinin ilişkisel bir uzantısı olan ilişkisel bir sistemdir. Hamburg Üniversitesi'nde geliştirildi ve ilk sürümü 1978'e kadar uzanıyor. PASCAL/R sistemi üç arayüzü destekler:

Bazı temel ilişkisel sistemlere genel bakış

- PASCAL/R programları aracılığıyla;
- etkileşimli arayüz;
- DB yöneticisi için ayrıcalıklı arayüz.

- Veritabanı açıklaması, PASCAL/R programının tür tanımlama bölümünde yapılır.
- Veri seçimi, formun ilişkisel ifadeleri kullanılarak yapılır:

```
some t in R : p(t);
each t in R : p(t).
```

- burada *t* bir n-demet adıdır, *R* bir ilişki adıdır ve *p(i)* bir yüklemdir.

Bazı temel ilişkisel sistemlere genel bakış

Example

```
var  ORDERLIST : file of Item;
     ...
  with  MYDATABASE  do
    begin
      ...
      for each P in Parts : (P.Colour = blue) and
          some SP in Shipments : (SP.PNR = P.PNR) do
        begin
          ORDERLIST^ := P;
          put(ORDERLIST)
        end
    end;
```

- Boolean ilişkisel ifadeler döngü operatörlerinde ve koşullu operatörde kullanılabilir. Sistem sanal tabloları desteklemiyor.

Bazı temel ilişkisel sistemlere genel bakış

ÖRNEK SORGULAMA SİSTEMİ

- ÖRNEKLE SORGULAMA (QBE), 1978 yılında ABD'deki IBM laboratuvarında geliştirilen ilişkisel bir sistemdir. Bu sistemin genişletilmiş versiyonuna OPE (Örnek Olarak Ofis Prosedürleri) adı verilir.
- Sistemde kullanılan dil QBE'dir. Doğrudan ek olarak kullanıcı, QBE adreslemeye izin verilen PL/1 veya APL'de derlenen programlar aracılığıyla sistemle etkileşime girebilir. Sistem ayrıca aşağıdaki olasılıklara da izin verir:
- - bütünlük kısıtlamalarının tanımı;

Bazı temel ilişkisel sistemlere genel bakış

- yeni bir DB'nin oluşturulması ve mevcut bir DB'nin yeniden düzenlenmesi;

- çıkış jeneratörü;

- dizinler oluşturma, sütunları ters çevirme.

Sistem, katalogda TABLE, DOMAIN, PROGRAM ve AUTHORITY adlı dört tür sistem tablosunu destekler. Son iki tablo, bireysel kullanıcılar için program veya Veritabanı sorgu adlarını ve erişim haklarını içerir.

Co-funded by the Erasmus+ Programme of the European Union

Bazı temel ilişkisel sistemlere genel bakış

SİSTEM R SİSTEM

- SYSTEM R, 1975 yılında ABD'deki IBM Laboratuvarı tarafından geliştirilen tamamen ilişkisel bir sistemdir. Sistemde kullanılan dil SQL'dir.

- Kullanıcı, SQL komutlarını kullanarak veya SQL komutlarını eklediği PL/1 veya COBOL programları aracılığıyla sistemle diyalog modunda çalışır.

Bazı temel ilişkisel sistemlere genel bakış

MySQL SİSTEMİ

- MySQL, bir veritabanına girmek, erişmek ve diğer veri işlemlerini gerçekleştirmek için en popüler dil olan SQL'i kullanan açık kaynaklı bir ilişkisel veritabanı yönetim sistemidir (RDBMS).
- Açık kaynak olduğundan, genel kurallara bağlı olarak herkes MySQL'i indirebilir ve ona ekleme yapabilir.
- MySQL esas olarak hızı, güvenliği ve esnekliği ile bilinir. Ancak içeriği yönettiğinde ve işlem yapmadığında en iyi şekilde çalıştığı biliniyor.

Bazı temel ilişkisel sistemlere genel bakış

Microsoft Access SİSTEMİ

Microsoft Access tamamen işlevsel bir ilişkisel veritabanı yönetim sistemidir. Veri işleme ve kontrol için tüm olanakları sağlar:

1. Veri tanımı ve depolanması. Microsoft Access, farklı veri türlerini (metin, sayılar, tarihler, para birimi, resimler, sesler, belgeler, elektronik tablolar, İnternet bağlantıları) tanımlamak için esnek seçeneklere sahiptir.

Kullanıcı, verilerin ekranda veya yazıcıda saklanma ve görüntülenme şeklini tanımlayabilir. Veritabanına doğru bilgilerin girilmesini sağlayan basit ve karmaşık veri geçerliliği kurallarını tanımlama yeteneği özellikle önemlidir.

Bazı temel ilişkisel sistemlere genel bakış

2. Veri işleme. Microsoft Access, güçlü SQL ilişkisel veri modeli dilini kullanır. Gerekli sonuç bilgisini elde etmek için kullanıcının SQL bilgisine ihtiyaç duymadan oluşturulan sorguları tanımlama konusunda güçlü bir araca sahiptir. Birçok tablodaki verileri birleştirerek sorgular oluşturulabilir.

3. Veri kontrolü. Microsoft Access, tek bir iş istasyonunda veya ağ üzerinde istemci/sunucu modunda kullanılabilir. Diğer kullanıcılarla eksiksiz ve verimli veri paylaşımı için Access mükemmel güvenlik ve veri entegrasyonu yeteneklerine sahiptir. Hangi kullanıcıların veya kullanıcı gruplarının nesnelere erişebileceğini ve erişim düzeyini (verileri okuyabilmek veya düzenleyebilmek ve güncelleyebilmek için) tanımlayabilirsiniz.

Bazı temel ilişkisel sistemlere genel bakış

Microsoft SQL Sunucu SİSTEMİ

- Microsoft SQL Server, çeşitli e-ticaret çözümleri, iş uygulamaları ve büyük veri depolama oluşturmak için kullanılan genel bir ilişkisel veritabanı yönetimi ve veri analitiği sistemidir.
- Tüm Microsoft ürünleri gibi, Microsoft SQL Server'ın da birçok farklı sürümü vardır. Küçük tek makineli uygulamalardan, birçok kullanıcının aynı anda üzerinde çalıştığı devasa İnternet tabanlı uygulamalara kadar çok çeşitli hedef kitleler ve iş yükleri için tasarlanmıştır.
- Microsoft SQL Server, masaüstü tabanlı olan ve büyük kurumsal veri tabanlarını yönetmek için tasarlanmamış olan MS Access'in aksine, büyük sunucu tabanlı veri tabanlarını yönetmek üzere tasarlanmıştır.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. RDBMS nedir? DBMS'den farkı nedir?

2. RDBMS'nin temel özellikleri nelerdir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkürler!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

❑ Modül 2. Veritabanlarında ilişkisel yaklaşım

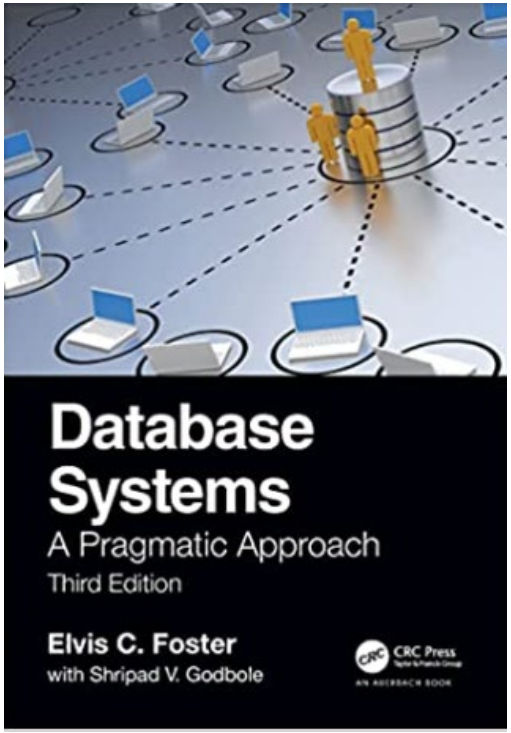
❑ Konu 3. İlişkisel sistemler.

- ❑ Ders 2. İlişkisel şema analizi. İşlevsel bağımlılıklar

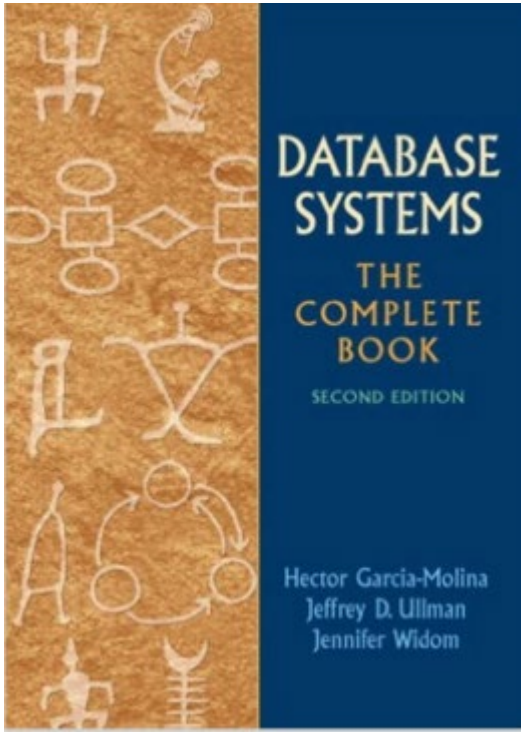
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

İLİŞKİSEL ŞEMA ANALİZİ. FONKSİYONEL BAĞIMLILIKLAR.



Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.



Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

Azalov, P. (1991) Database, Sofia.

Giriş

- Bir konu alanına ilişkin modellerin oluşturulmasındaki temel sorun, konu alanındaki nesne sınıfları için seçilen varlıkların (ilişkiler, tablolar) ve bunları karakterize eden özelliklerin (nitelikler, dallar) tanımıdır.
- Sorunun karmaşıklığı, çözümünün belirsizliğinden kaynaklanmaktadır.
- Böyle bir durumda şu sorular ortaya çıkabilir: ***Bir nesne sınıfını temsil eden ilişkisel şema iyi olduğunda; iki şema eşdeğer olduğunda veya iki şemadan hangisinin daha iyi olduğu***.
- Bu soruların her biri aslında ilişkisel şemaları analiz etmek için resmi bir araca ihtiyaç duyulmasına yol açmaktadır.

Giriş

- İlişkisel şemaların analizi ve değerlendirilmesine yönelik yöntemleri sunmadan önce, belirli bir ilişkisel şemanın çeşitli eksikliklerini gösteren bir örnek sunulmaktadır.
- Literatürde bu eksiklikler ***fazlalık, güncelleme, ekleme ve silme anormallikleri*** olarak bilinmektedir.
- Bir ilişki şeması sunulmuştur:

LIBRARY (Lib#, Address, Book, Number)

burada her kütüphane numarası ***(Lib#)*** ve adresi ***(Adres)*** ile sunulur ve her kitap kendi adı ***(Kitap)*** ve kütüphanede mevcut olan kopya sayısı ***(Numara)*** ile sunulur.

Co-funded by the Erasmus+ Programme of the European Union

Giriş

- Şunu varsaymak doğaldır:

- her kütüphane, numarasıyla benzersiz bir şekilde tanımlanır;

- bir kitap birden fazla kütüphanede saklanabilir.

- Sonuç olarak aşağıdaki sonuçlara varılabilir:

1. Aynı kütüphanedeki her kitap için kütüphane adresi tekrarlanacaktır.

2. (1) numaralı noktadan, kütüphane adresi değiştiğinde, içinde depolanan kitap sayısı kadar ilişkinin satırının güncellenmesinin gerekli olacağı sonucu çıkmaktadır.

Co-funded by the Erasmus+ Programme of the European Union

Giriş

Bu işlem, veritabanı verilerinde tutarsızlıkların meydana geleceği bazı satırların güncellenmeden kalması olasılığını dışlamaz.

Yeni oluşturulan bir kütüphanenin adresi, o kütüphanede saklanacak en az bir kitap belirtilmeden DB'ye girilemez. Bunun nedeni, bu ilişkisel şemadaki anahtar niteliklerin Lib# ve Book nitelikleri olması ve dolayısıyla ilişkinin her öğesi için iyi tanımlanmış değerlere sahip olmaları gerektiğidir.

Bir kütüphanede tadilat yapılması nedeniyle o kütüphanede bulunan kitapların başka bir kitap deposuna veya kütüphaneye taşınması durumunda, o kütüphaneye ait ilişkideki tüm satırların ilişkiden çıkarılması gerekir.

Giriş

Sonuç olarak, kütüphane mevcut olsa ve adresi aynı kalsa bile kütüphanenin adresi de "kaybedilecektir".

Artıklık anomalisi (1), güncelleme anomalisi (2), anomali ekleme (3) ve anomaliyi kaldırma (4) olarak adlandırılan yukarıdaki eksiklikler, KÜTÜPHANE ilişkisel şemasının "iyi" olmadığını göstermektedir.

Onun "gelişmesine" yol açan yöntemler, öncelikle ilişkisel bir şemanın nitelikleri arasında açıkça veya örtülü olarak var olan bağımlılıkları analiz etmeye yönelik yöntemlerin sunulmasını gerektirir.

Co-funded by the Erasmus+ Programme of the European Union

Giriş

- Onun "gelişmesine" yol açan yöntemler, öncelikle ilişkisel bir şemanın nitelikleri arasında açıkça veya örtülü olarak var olan bağımlılıkları analiz etmeye yönelik yöntemlerin sunulmasını gerektirir.

Çoğu zaman iki tür bağımlılık dikkate alınır: işlevsel ve çok değerli.

Fonksiyonel Bağımlılık, İlişki Veritabanı Yönetim Sisteminde bir özelliğin başka bir özellik ile ilişkisini belirleyen bir kısıtlamadır.

FONKSİYONEL BAĞIMLILIKLAR

- $A_1, A_2,...,A_n$ n niteliğin bir listesi olsun, $n>=1$, bunu $A_1, A_2,...,A_n$ (öznitelik adları arasında virgül olmadan) veya yalnızca tek bir adla A, $A=$ ile göstereceğiz. $A_1, ..., A_n$
- A listesinin sırasız olduğu ve tekrarlanan elemanların olmadığı, A'nın bir dizi nitelik adı olduğu varsayılmaktadır. $P(A)$ ile A'nın tüm alt kümelerinin kümesini belirtiyoruz.
- Eğer A'nın n elemanı varsa $|A| = n$ ise, A'nın alt kümelerinin sayısı 2^n, $|P(A)|= 2^n$.
- $R(A)$ bir ilişkisel şema olsun ve $X, Y \in P(A)$ olsun.

FONKSİYONEL BAĞIMLILIKLAR

- Bir *X* niteliğinin, bir *Y* niteliğini işlevsel olarak belirlediği söylenir; eğer ilişkisel şema *R(A)* ile herhangi iki satır verilen i_1, $i_2 \in r$ ve r $i_1.X = i_2.X$ olan herhangi bir *r* ilişkisi için, bundan $_1.Y = i_2.Y$ çıkar. *.Y, X* özelliğinde satırlar eşit olduğunda *Y* özelliğinde de eşittir.
- *X* ve *Y*'nin hem *basit nitelikler* hem de *bileşik nitelikler* olabileceği, birkaç basit nitelik içerebileceği unutulmamalıdır.
- *X* işlevsel olarak *Y*'yi belirlediğinde, *R. X - - - > R. Y* gösterimi kullanılarak *Y*'nin işlevsel olarak *X*'e bağımlı olduğu veya *X*'in *Y*'de işlevsel bir bağımlılığı olduğu da söylenir.

FONKSİYONEL BAĞIMLILIKLAR

- Belirsizlik yoksa ilişkisel şemanın belirtilmesi isteğe bağlıdır.
- Aslında iki nitelik arasındaki işlevsel bağımlılığın, $X, Y \in P(A)$ olduğu $X - - - > Y$ biçimindeki ilişkisel şema $R(A)$ üzerinde bir tamlık kısıtlaması olarak da değerlendirilebileceği fark edilebilir.
- Bu kısıtlama, herhangi bir r; $r:R(A)$ ilişkisi için X özelliğinin herhangi bir değerinin, Y özelliğinin değerini benzersiz bir şekilde belirlediğini garanti eder.
- Fonksiyonel bağımlılığın biçimsel ifadesi için ilişkisel cebirdeki işlemler de kullanılabilir.

FONKSİYONEL BAĞIMLILIKLAR

- Çoğunlukla işlevsel bağımlılıklar kümesi F ile gösterilir:

$$F = \{(X,Y) \mid X \dashrightarrow Y,\ X \in \wp(A),\ Y \in \wp(A)\},$$

F ikili bir ilişkidir.

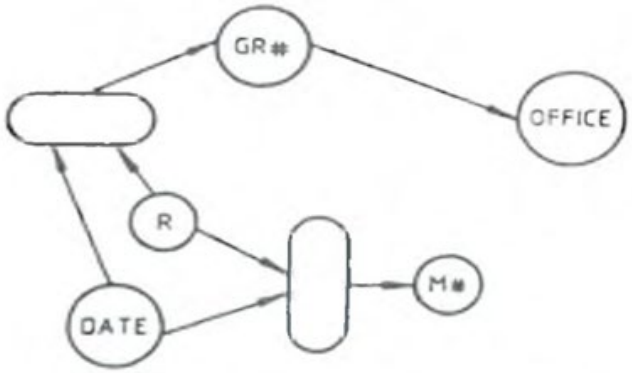
- Ayrıca *X* -- -- -> *Y*'nin *F*'ye ait olması durumunda *X* -- -- -> *Y*'nin bir *F*-bağımlılığı olduğunu söylemek de gelenekseldir.
- Eğer *X* - - - >*Y, r* ilişkisi tarafından karşılanan bir *F* bağımlılığı ise, kısaca bunu *r(X* - - - >*Y) = doğru* olarak yazacağız ve *X* - - - >*Y* olmaması durumunda *r* için bir *F* bağımlılığı varsa, *r(X* - - - >*Y) = yanlış* yazacağız.

FONKSİYONEL BAĞIMLILIKLAR

- F bağımlılıkları kümesi ikili bir ilişki olduğundan, bunu temsil etmenin uygun ve görsel bir yolu $G = (A, F)$ grafiğidir; burada A = $A = \{A_1, A_2, \ldots, A_n\}$ bir dizidir. niteliklerdir ve $F, X - - - > A, X \in P(A)$ formundaki bir F bağımlılıkları kümesidir.
- Her bir $A_i \in A$ özelliğinde grafiğin bir köşesini eşleştiririz.
- X basit bir nitelik olduğunda, F-bağımlılığı $X - - - > A_i$'ye bir yay eşleriz.
- Ancak X'in bileşik bir nitelik olduğu durumlar da vardır. Bu durumda kurucu niteliklere karşılık gelen ikinci türden köşeler girilir.

FONKSİYONEL BAĞIMLILIKLAR

- Eğer $X = A_{i1}, A_{i2}, \ldots,$ ise ilk önce $(A_{ij}, X), j = 1,2, \ldots, k\ A_{ik}$ yayları ve ardından (X, A_i) yayı oluşturulur.
- İlişkisel şemanın F bağımlılıkları GEZİ



EXCURSION (OFFICE#, R, GR#, M#, DATE)

Attributes

OFFICE# - Travel agency (number);
R - Exrursion group guide
GR# - Excursion group number
M# - Route number
DATE - Departure date

F-dependencies

(R, DATE) → GR#
(R, DATE) → M#
GR# → OFFICE

F-dependencies graph sample

FONKSİYONEL BAĞIMLILIKLARIN AKSİYOMATİĞİ

- 3 üyeli *r* ilişkisini göz önünde bulunduruyoruz:

r	A	B	C
	a	b	c
	a	b1	c
	a1	b	c1

- *r*'nin tablo halinde gösteriminin birinci ve ikinci satırlarından A --- > C olduğu düşünülebilir.
- Bununla birlikte, *F*-bağımlılığı belirli bir ilişkinin değil ilişkisel şemanın bir özelliği olduğundan, *r* ilişkisinin tüm durumları bilinmeden bu sonuç doğru olmayacaktır.

FONKSİYONEL BAĞIMLILIKLARIN AKSİYOMATİĞİ

- Aslında her F bağımlılığı, veritabanının mevcut bilgi içeriğine göre değişmez olduğundan, modellenen nesne sınıfının anlambilimini tamamlar.
- Bir ilişki şeması içerisinde tanımlanabilecek bağımlılıklar, sayıları sınırlı da olsa bazen çok fazla olabiliyor.
- Bu nedenle, belirli bir dizi fonksiyonel bağımlılık F üzerinden tüm fonksiyonel bağımlılıkların nasıl elde edilebileceğini sormak doğaldır.

FONKSİYONEL BAĞIMLILIKLARIN AKSİYOMATİĞİ

- Bir F-bağımlılığı *X --- > Y,* F-bağımlılıklarını karşılayan herhangi bir *r* ilişkisi için, *r(F)=doğru* ise, *F* fonksiyonel bağımlılıklar kümesinin mantıksal bir sonucu olarak adlandırılır, *r(X - - - > Y)=doğru.*
- Kısaca bunu *F:= X --- >Y* şeklinde göstereceğiz.
- Armstrong, belirli bir F-bağımlılıkları dizisi için çıkarım kuralları veya aksiyomları kullanarak yeni F-bağımlılıklarının elde edildiğini ve tüm F-bağımlılıklarının elde edilebileceğini gösteriyor.

FONKSİYONEL BAĞIMLILIKLARIN AKSİYOMATİĞİ

- Aşağıda verilen çıkarım aksiyomlarının formülasyonunda şu varsayılmaktadır: **R(A)** ilişkisel bir şemadır; **F**, **A**'daki F bağımlılıklarının kümesidir; **X, Y, Z** ve **W, A**'nın niteliklerinin listesidir; **r, R(A)** şemasıyla keyfi bir ilişkidir.

F1. ***Reflectiveness***

If $Y \subseteq X$, than $F \models X \dashrightarrow Y$.

F2. ***Augumentation***

If $X \dashrightarrow Y$, than $F \models XZ \dashrightarrow YZ$.

F3. ***Transitiveness***

If $X \dashrightarrow Y$ and $Y \dashrightarrow Z$, than $F \models X \dashrightarrow Z$.

FONKSİYONEL BAĞIMLILIKLARIN AKSİYOMATİĞİ

- Bu kuralların kanıtı doğrudan F-bağımlılığının tanımından gelir.
- Örneğin birincinin kanıtı şudur: Eğer *r*'nin $i_1.X=i_2.X$ olacak şekilde $\dot{I}_1$ ve i_2 elemanları varsa, $Y \subseteq X$ olduğundan $i_1.Y=i_2.Y$, sonucu çıkar. *F1* geçerlidir.
- Benzer şekilde *F2* ve *F3* de kanıtlanabilir.
- *F1, F2* ve *F3*'e dayanarak diğer çıkarım kuralları kanıtlanabilir.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

FONKSİYONEL BAĞIMLILIKLARIN AKSİYOMATİĞİ

F4. Unification

If $X \dashrightarrow Y$ and $X \dashrightarrow Z$, than $X \dashrightarrow YZ$.

Really from $X \dashrightarrow Y$ and from F2 follows that $X \dashrightarrow XY$, and from $X \dashrightarrow Z$ and from F2 follows that $XY \dashrightarrow ZY$. Than according to F3 $X \dashrightarrow YZ$.

F5. Pseudo transitiveness

If $X \dashrightarrow Y$, $WY \dashrightarrow Z$, than $XW \dashrightarrow Z$.

Really, from $X \dashrightarrow Y$ and from F2 follows that $XW \dashrightarrow YW$. Than from F3 follows that $X \dashrightarrow Z$.

F6. Decomposition

If $X \dashrightarrow Y$ and $Z \subseteq Y$, than $X \dashrightarrow Z$.

Really, from $Z \subseteq Y$ and F1 follows that $Y \dashrightarrow Z$ and from F3 finally follows F6

FONKSİYONEL BAĞIMLILIKLARIN AKSİYOMATİĞİ

- F4 ve F6 aksiyomlarından birleştirme ve ayrıştırmanın sonucu formüle edilebilir:

$X \dashrightarrow Y_1, Y_2, \ldots, Y_k$, ancak ve ancak

$X \dashrightarrow Y_i$, за $i = 1,2,\ldots,n$

KAPANIŞLAR VE KAPAKLAR

- F'nin mantıksal sonuçları olan F bağımlılıkları kümesine F kapanması denir ve F^+ ile gösterilir.
- Genel durumda F fonksiyonel bağımlılıkları kümesinin F+ kapanışını bulmak kolay değildir.
- Bunun temel nedeni genellikle kapatma özelliklerinin sayısının çok fazla olmasıdır.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

KAPANIŞLAR VE KAPAKLAR

Example: A relational scheme *R(A,B,C)* with functional dependencies *F={A-->B,B-->C}* is given. Define the F^+ closure as F-dependencies if *ϕ-->X or X -->ϕ* are not considered.

1. A--->A
2. A--->B
3. A--->C
4. A--->AB
5. A--->AC
6. A--->BC
7. A--->ABC
8. B--->B
9. B--->C
10. B--->BC
11. C--->C
12. AB--->A
13. AB--->B
14. AB--->C
15. AB--->AB
16. AB--->AC
17. AB--->BC
18. AB--->ABC
19. AC--->A
20. AC--->B
21. AC--->C
22. AC--->AB
23. AC--->AC
24. AC--->BC
25. AC--->ABC
26. BC--->B
27. BC--->C
28. BC--->BC
29. ABC--->A
30. ABC--->B
31. ABC--->C
32. ABC--->AB
33. ABC--->AC
34. ABC--->BC
35. ABC--->ABC.

KAPANIŞLAR VE KAPAKLAR

- Tüm *F* bağımlılıklarını bulmak her zaman gerekli değildir. Bazen sadece *F* bağımlılığının kapanışa ait olup olmadığını bilmek yeterlidir.
- Örneğin, F bağımlılıkları olan R*(A1, A2, A3, A4, A5, A6, A7)* ilişkisel şeması dikkate alınırsa *F{A1 ---> A4, A1A2 ---> A4A5, A3A5 ---> A6 , A5 ---> A7}* ise, *A1A2 ---> A5A7*'nin F^+ 'nın bir elemanı olup olmadığının kontrol edilmesinin, önce F^+ 'nın belirlenmesini gerektirmediği açıktır.

Co-funded by the Erasmus+ Programme of the European Union

KAPANIŞLAR VE KAPAKLAR

- Doğrudan yöntemlerin kullanılması, kapatma özelliği kavramının tanıtılmasını gerektirir.
- *F, R(A)* ve $X \in P(A)$ içindeki *F* bağımlılıklarının kümesi olsun. *X* --- > *Y*'nin bir F-bağımlılığı olduğu herhangi bir *Y*, $Y \in P(A)$ için, bundan $Y \subseteq X^+$ çıkarsa, X^+ kümesine X'in kapanışı denir.
- Başka bir deyişle, bir *X* özelliğinin (muhtemelen bileşik) X^+ kapanışı, *X* --- > X^+ 'nın F^+'dan geldiği maksimum özelliktir.

KAPANIŞLAR VE KAPAKLAR

- F'nin hiçbir uygun alt kümesi F'ye eşdeğer değilse, F bağımlılıkları kümesine artıklık içermeyen kapak adı verilir.
- Bu, eğer F artıklık olmayan bir örtü ise, o zaman her *F', F' ⊂ F* için *(F')+ ≠ F+*'ya sahip olduğumuz anlamına gelir.
- Örneğin, *G = {A --- > B, B --- > C, AB --- > C, AC --- > B)* kümesinin, F bağımlılıkları kümesi *F = {A --- > B , B --- > C)*, çünkü *F ⊂ G* ve *F+= G+*.
- Bir dizi F-bağımlılığının fazlalık olmadan birden fazla kapsama sahip olabildiği durumların olduğunu belirtmek ilginçtir.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. İşlevsel bağımlılık nedir?
2. İşlevsel bağımlılıkların kuralları nelerdir?
3. DBMS'deki Fonksiyonel Bağımlılık türleri nelerdir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

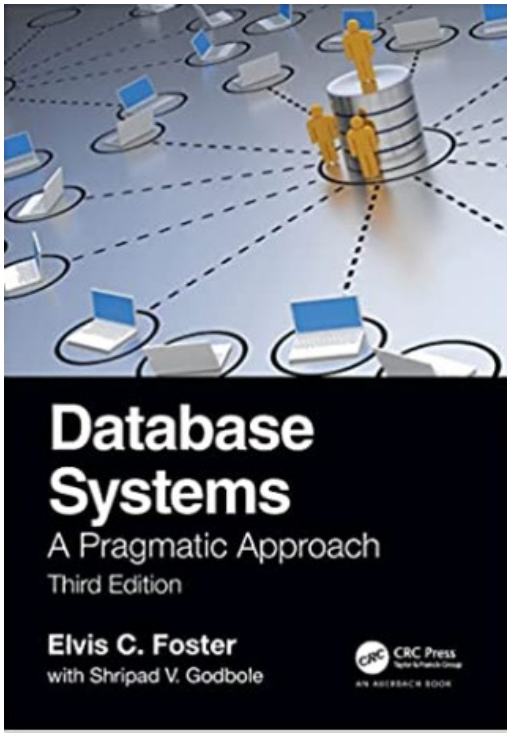
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

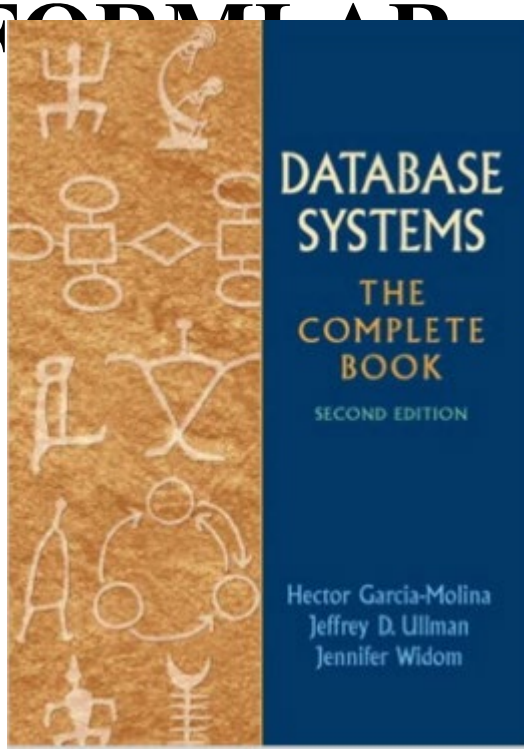
- **Modül 2. Veritabanında ilişkisel yaklaşım**
  - **Konu 4. Örnek bir veritabanının geliştirilmesi. Normalleştirme.**
    - Ders 1. İlişkisel şemaların normalleştirilmesi. Normal formlar.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

İLİŞKİSEL ŞEMALARIN NORMALLEŞTİRİLMESİ. NORMAL FORMLAR



Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.



Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

Azalov, P. (1991) Database, Sofia.

İlişkisel şemaların normalleştirilmesi. Normal formlar.

- Herhangi bir ilişkisel şema *R(A)* ile çalışmak, *A* nitelik kümesinin bir alt kümesi olan anahtarların kullanılmasını içerir.
- *R(A)* ilişkisel şemasına sahip her bir *r* ilişkisinin bireysel satırlarındaki bu niteliklerin değerleri, onları benzersiz bir şekilde tanımlar.
- Daha kesin olarak, bir *X* öznitelik kümesi, iki koşulu karşılıyorsa, F-bağımlılıkları *F* olan bir ilişkisel şema *R(A)* için bir anahtardır:

1. Teklik: $\{X \dashrightarrow A\} \in F+$.

2. Minimalite: *X*'in herhangi bir *Y* alt kümesi için, $Y \subset X$, elimizde $\{Y \dashrightarrow A\}$ *F*+'ya ait değildir.

İlişkisel şemaların normalleştirilmesi. Normal formlar.

- Benzersizlik ve minimallik özelliklerini karşılayan birkaç X kümesinin var olması mümkün olduğundan, ilişkilerle çalışırken fiilen kullanılan anahtarın **birincil anahtar** olduğu varsayılır ve bundan sonra yalnızca **anahtar** olarak anılacaktır.
- Bir önceki bölümde "iyi" olmayan ilişkisel şemaların var olduğunu gösteren örnekler tartışılmıştı. Bir sonraki bölümde ilişkisel şemaların yukarıda belirtilen dezavantajlarını gidermeye yönelik yöntemler sunulmaktadır.

NORMAL FORMLAR

- *İlişkisel şemanın normalleştirilmesi, yeni elde edilen ilişkisel şemalara belirli kısıtlamaların getirilerek bazı istenmeyen özelliklerin ortadan kaldırıldığı, dönüşümünü amaçlayan bir süreçtir.*
- "EMPLOYEE", "EMPLOYEE_1", "EMPLOYEE_2" ve "EMPLOYEE_3" ilişkilerini düşünün.
- İçerdikleri bilgilere göre bunların eşdeğer olduğunu varsayabiliriz.
- Ancak matematiksel nesneler olarak düşünüldüğünde tamamen farklıdırlar.
- Birincisi 2 üyeli, diğerleri ise sırasıyla 3, 5 ve 6 üyeli ilişkilerdir.

NORMAL FORMLAR

EMPLOYEE	IDENTIFICATION NUMBER	NAME
	4705127680 5212066341 5008156874	Petar Ivanov Maria Hristova Ivan Pavlov

EMPLOYEE_1	Code	Date	Name
	7680 6341 6874	470512 521206 500815	Petar Ivanov Maria Hristova Ivan Pavlov

NORMAL FORMLAR

EMPLOYEE_2	Code	Day	Month	Year	Name
	7680	12	5	47	Petar Ivanov
	6341	6	12	52	Maria Hristova
	6874	15	8	50	Ivan Pavlov

EMPLOYEE_3	Code	Day	Month	Year	First name	Last name
	7680	12	5	47	Петър	Иванов
	6341	6	12	52	Мария	Христова
	6874	15	8	50	Иван	Павлов

NORMAL FORMLAR

- Bir ilişkisel şema, kendisini oluşturan niteliklerin alanları atomik (basit) ise veya nitelikler atomikse ve tekrarlanan nitelikler yoksa birinci norm formundadır (1NF).
- *Atomiklik,* belirli bir durumda belirlenen bir kavramdır.
- Örneğin ÇALIŞAN ilişkisinde işlenen bilgiler sadece çalışanların kimlik numaraları ve isimlerine ilişkin ise INF'de olduğunu varsaymalıyız.
- Ancak bu ilişkiyle çalışırken kimlik numarasının bazı kısımlarını kullanmamız gerekirse, örn. G. gün, ay veya yıl, kimlik numarasının atomik olmayan (bileşik) bir nitelik olarak değerlendirilmesi gerekir ve bu nedenle ÇALIŞAN ilişkisi INF'de olmayacaktır.

NORMAL FORMLAR

- Bazı uygulamalarda "Ad" niteliği bileşik bir nitelik olabilir ve onun parçaları (ad ve soyadı) kullanılır.
- EMPLOYEE_3 ilişkisinin ilişki şeması yalnızca atomik nitelikleri içerir ve INF'dedir. Anahtarı oluşur ve nitelikleri içerir (Kod, Yıl, Ay, Gün).
- Bir nitelik, elemanlarının küme olması durumunda atomik de olmayabilir.
- PERSONEL ilişkisinde de durum böyledir. Bu ilişki INF'de değildir. INF'deki normalizasyonu PERSONNEL_1 ilişkisi tarafından verilmektedir.

Co-funded by the Erasmus+ Programme of the European Union

NORMAL FORMLAR

PERSONNEL	Name	Education
	Petar Ivanov Maria Hristova Ivan Plavlov	{Secondary technical} {Mathematical, engineering} {Economic}

PERSONNEL_1	Name	Education
	Petar Ivanov Maria Hristova Maria Hristova Ivan Pavlov	{Secondary technical} {Mathematical} {Engineering} {Economic}

Co-funded by the Erasmus+ Programme of the European Union

NORMAL FORMLAR

- Normalleştirilmiş ilişkilerin (INF'deki ilişkiler) kullanılmasına ilişkin çeşitli özel hususlar olabilir.
- En önemlilerinden biri, F bağımlılıklarının temsilinde doğruluk ihtiyacıdır.
- Örneğin, EMPLOYEE_1 ilişki şemasına "Yıldız işareti" özelliği eklenirse, F bağımlılığının varlığını ifade etmek mümkün değildir:

(Gün, Ay) → Yıldız burcu

NORMAL FORMLAR

- Ancak bu F bağımlılığı EMPLOYEE_2 ve EMPLOYEE_3 ilişkilerinin ilişki şemalarında temsil edilir.

INF'ye normalleştirmenin başka bir örneği:

EMPLOYEE (No, name, Child (firstname, age))
EMPLOYEE (No, name); CHILDREN(No, firstname, age)

No	NAME	CHILD	
		FIRSTNAME	AGE
500	DUPONT	ANDRE	10
501	DURAND	JEAN	11
501	DURAND	PIERRE	12
510	LEFEBVRE	PAUL	13
510	LEFEBVRE	JACQUES	14

EMPLOYEE	**No**	**NAME**
	500	DUPONT
	501	DURAND
	510	LEFEBVRE

CHILDREN	**No**	**FirstName**	**age**
	500	André	10
	501	Jean	11
	501	Pierre	12
	510	Paul	13
	510	Jacques	14

Co-funded by the Erasmus+ Programme of the European Union

NORMAL FORMLAR

- *R(A),* F-bağımlılığı *F* olan bir ilişkisel şema olsun. *X, X,Y ∈ P(A)* if *{X --->Y}* ∈ F^+ ise, bir Y niteliğinin bir *X, X,Y ∈ P(A)* niteliğine tam işlevsel bağımlı olduğu söylenir. ve eğer her X' alt kümesi için, X', *{X' --- > Y}* ∈ F^+ sağlanırsa.
- Örneğin, "Yıldız burcu" özelliği, niteliklere (Gün, Ay, Yıl) tam olarak F bağımlı değildir, ancak niteliklere (Gün, Ay) tam olarak F bağımlıdır.

NORMAL FORMLAR

- Bir F-bağımlılığı F'ye sahip bir ilişkisel şema R(A)'nın bir X niteliği, eğer anahtarın bir parçası ise, birincil nitelik olarak adlandırılır. Aksi halde birincil olmayan denir.
- Ancak şu ana kadar bir sorunla karşılaşmadım:
- EĞİTİM (Fakülte#, İsim, Şehir, Konu#, Marka)
- Bu ilişkinin anahtarı bileşiktir (Fakülte#, Konu#). F bağımlılıkları kümesi:
- (Fakülte#, Konu#) → Fakülte#, İsim, Şehir, Konu#, İşaret
- Fakülte# → Ad, Şehir

Co-funded by the Erasmus+ Programme of the European Union

NORMAL FORMLAR

EDUCATION	Faculty#	Name	City	Subjet #	Mark
	40385	Ivan Todorov	Pleven	M25	4.0
	40001	Teodosi Sabev	Kazanlak	M21	6.0
	41012	Rumyana Mineva	Kazanlak	C12	6.0
	40205	Hristo Petrov	Sofia	P09	5.5
	40385	Ivan Todorov	Pleven	M21	5.0
	40385	Ivan Todorov	Pleven	P09	6.0
	41012	Rumyana Mineva	Kazanlak	M25	5.5
	40205	Hristo Petrov	Sofia	M21	5.0
	41012	Rumyana Mineva	Kazanlak	M21	6.0

Co-funded by the Erasmus+ Programme of the European Union

NORMAL FORMLAR

- Açıkçası, Ad ve Şehir nitelikleri anahtarın F'sine tamamen bağımlı değildir. Bunun sonuçları nelerdir?
- 1. Öğrenci en az bir sınavı geçinceye kadar Konu# anahtar özelliğinin değeri tanımsız olduğundan o öğrenciye ait bilgiler (Fakülte#, Ad, Şehir) girilemez.
- 2. Öğrencinin adı ve şehri, geçilen sınav sayısı kadar tekrarlanır.
- 3. Bir öğrencinin tek sınavı iptal edilirse, bunun sonucunda o öğrenciye ait o sınavla ilgili olmayan bilgiler silinecektir.

NORMAL FORMLAR

- Bir ilişkisel şema, eğer INF'deyse ve birincil olmayan her öznitelik, anahtar üzerinde tam F bağımlılığındaysa, F bağımlılıkları kümesinde ikinci normal formdadır (2NF).
- Bu tür oluşturulmuş ilişkisel şemanın yukarıdaki dezavantajları, bunun yerine ikinci normal formdaki (2NF) iki ilişkisel şemayı ele alırsak ortadan kaldırılabilir.
- ÖĞRENCİ (Fakülte#, Adı, Şehir);
- STUD_MARK (Fakülte#, Konu#, Not).

NORMAL FORMLAR

- Bu ilişkisel şemalardaki F bağımlılıkları :

Fac# ---> Name, City
Fac#, Subject# ---> Mark

STUDENT	Fac#	Name	City
	40385	Ivan Todorov	Pleven
	40001	Teodosi Sabev	Kazanlak
	41012	Rumyana Mineva	Kazanlak
	40205	Hristo Petrov	Sofia

STUD_MARK	FAC#	SUBJECT#	MARK
	40385	M25	4.0
	40001	M21	6.0
	41012	C12	6.0
	40205	P09	5.5
	40385	M21	5.0
	40385	P09	6.0
	41012	M25	5.5
	40205	M21	5.0
	41012	M21	6.0

NORMAL FORMLAR

- Ortaya çıkan ilişkilerde, birincil olmayan her nitelik, anahtara tam olarak *F* bağımlıdır.
- *R(A),* karşılık gelen F-bağımlılıkları *F* kümesine sahip bir ilişkisel şema olsun.
- *X* ve *Z*'nin *A*'nın alt kümeleri olduğu varsayılır. *Y, Y* ⊂ *A, Z*'nin *XY*'ye ait olmadığı ve aşağıdaki özelliklerin geçerli olduğu nitelikler kümesi mevcutsa, bir *Z* niteliğinin *X*'e *geçişli olarak bağımlı* olduğu söylenir:

$$(X \dashrightarrow Y) \in F^+$$
$$(Y \dashrightarrow X) \in F^+$$
$$(Y \dashrightarrow Z) \in F^+ .$$

NORMAL FORMLAR

- **Geçişli bağımlılık örneği**

Auhtors			
Author_ID	Author	Book	Author_Nationality
Auth_001	Orson Scott Card	Ender's Game	USA
Auth_001	Orson Scott Card	Ender's Game	USA
Auth_002	Margaret Atwood	The Handmaid's Tale	Canada

Yukarıdaki örnekte YAZARLAR: **Kitap → Yazar:** Burada Kitap niteliği Yazar niteliğini tanımlar. Kitabın adını biliyorsanız yazarının adını da öğrenebilirsiniz. Ancak bir yazar bir Kitabı tanımlamaz çünkü bir yazar birden fazla kitap yazabilir. Örneğin yazarın Orson Scott Card adını bildiğimiz için kitabın adını hâlâ bilmiyoruz.

yazar → Author_Nationality : Benzer şekilde, yazar özelliği Author_Nationality'yi tanımlar, ancak bunun tersi geçerli değildir; uyruğunu bilmemiz yazarın kimliğini tespit edebileceğimiz anlamına gelmez.

Ancak bu tablo geçişli bağımlılığı ortaya koymaktadır:

Kitap → Author_Nationality: Kitabın adını biliyorsak "Yazar" sütununu kullanarak uyruğunu tanımlayabiliriz.

Co-funded by the Erasmus+ Programme of the European Union

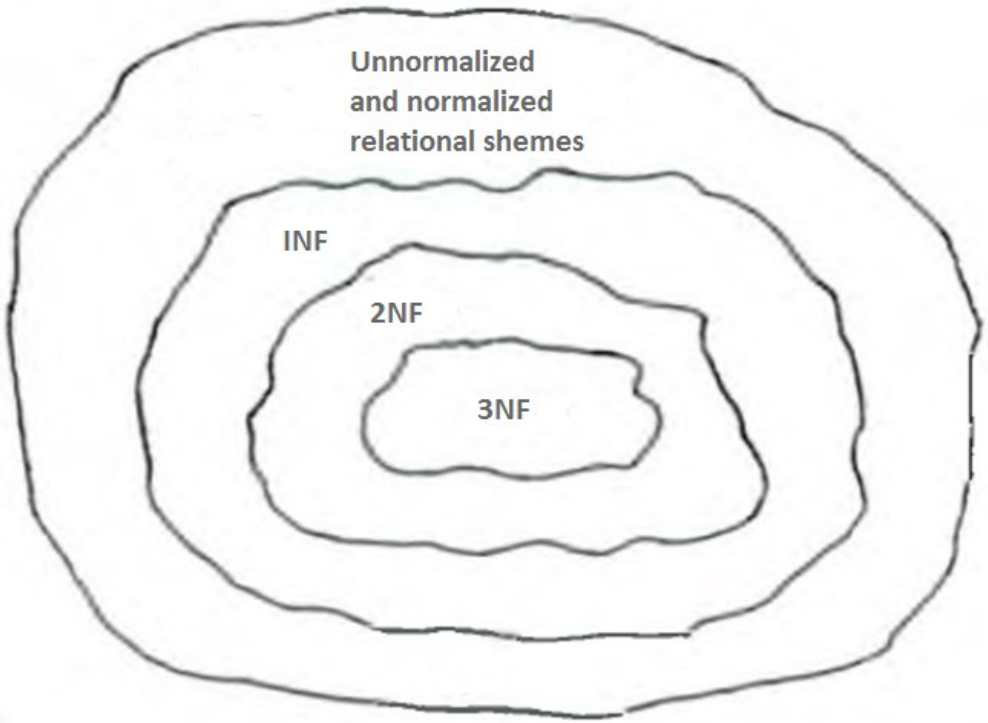
NORMAL FORMLAR

- Bir ilişkisel şema R(A), eğer INF'deyse ve hiçbir birincil olmayan nitelik anahtara geçişli olarak bağımlı değilse, F bağımlılıkları kümesinde üçüncü norm formundadır (3NF).
- 2NF'nin tanımı INF'nin varlığını gerektirir, yani. e. 2NF'de bulunan herhangi bir ilişki şeması aynı zamanda INF'de de vardır.
- 3NF'nin tanımı ilişkisel şemanın 2NF'de olmasını gerektirmez ancak 3NF'de yer alan herhangi bir ilişkisel şemanın da 2NF'de olduğu gösterilebilir.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

NORMAL FORMLAR



Input of the relation sets in 3NF, 2NF and IFN

NORMAL FORMLAR

- *R(A)* ilişkisel şemasının 3NF'de olduğunu ancak 2NF'de olmadığını varsayalım. $Z \in P(A)$ olsun ve *Z, R(A)*'nın *K* anahtarına tamamen F-bağımlı değildir.
- Bu, K' ---> Z'nin F-bağımlılığının geçerli olduğu K'nin bir *K'* alt kümesinin mevcut olduğu anlamına gelir.
- Ancak K bir anahtar olduğundan şu sonuç çıkar:

$$(K \dashrightarrow K') \in F$$

$$(K' \dashrightarrow K) \notin F^{+}$$

NORMAL FORMLAR

- Son olarak, yukarıda yapılan varsayımla çelişen geçişli bir *F* bağımlılığı elde edilir *(R(A)*, 3NF'de değildir).
- Bunun tersi ifade doğru değildir. Örneğin STUDENT_2 ilişkisi 2NF'dedir ancak 3NF'de değildir.

STUDENT_2	Fac_numb	Name	City	Region
	40385	Ivan Todorov	Pleven	Lovech
	40001	Teodosi Sabev	Kazanlak	Haskovo
	41012	Rumyana Mineva	Kazanlak	Haskovo
	40205	Hristo Petrov	Sofia	Sofia

NORMAL FORMLAR

- Bunun nedeni F bağımlılıklarıdır:
- Fac_numb Ad, Şehir, Bölge; - - - - >
- Şehir Bölgesi. - - - - >
- buradan geçişli F bağımlılığı gelir

NORMAL FORMLAR

- Bu geçişli bağımlılığa sahip olmanın etkileri nelerdir?
- 1. Plevne kasabasından tek bir öğrenci bile yoksa Bansko kasabasının Blagoevgrad bölgesinden olduğunun kaydedilmesi mümkün değildir.
- 2. Kazanlıklı öğrencilerin herhangi bir nedenle başka bir fakülteye taşınması durumunda Kazanlık kasabasının Stara Zagora bölgesinden olduğu bilgisi kaybolacaktır.
- 3. Kazanlak ilçesinin bölgesel merkez olacağı yeni bir bölgesel imar yapılması gerekiyorsa, o zaman STUDENT_2 ilişkisinde Kazanlıklı tüm öğrenciler kadar değişiklik yapılması gerekecektir. Bu da zaman kaybına yol açmakta ve verilerde tutarsızlıkların oluşmasına neden olacak koşulları yaratmaktadır, yani. e. DB'deki verilerin bütünlüğünü ihlal edecek koşullar vardır.

NORMAL FORMLAR

- STUDENT_2 ilişki şemasının yukarıda belirtilen dezavantajları geçişli bağımlılıktan kaynaklanmaktadır ve kaldırılmasıyla önlenmektedir.
- Bu, STUDENT_2'den türetilen STUDENT_3 ve ZONNING ilişkilerinin şemalarında yapılır. Bu iki ilişkisel şema 3NF'dedir.

СТУДЕНТ_3	Фак·ном	Име	Град
	40385	Иван Тодоров	Плевен
	40001	Теодоси Събев	Казанлък
	41012	Румяна Минева	Казанлък
	40205	Христо Петров	София

РАЙОНИРАНЕ	Град	Област
	Плевен	Ловеч
	Казанлък	Хасково
	София	София

Co-funded by the Erasmus+ Programme of the European Union

NORMAL FORMLAR

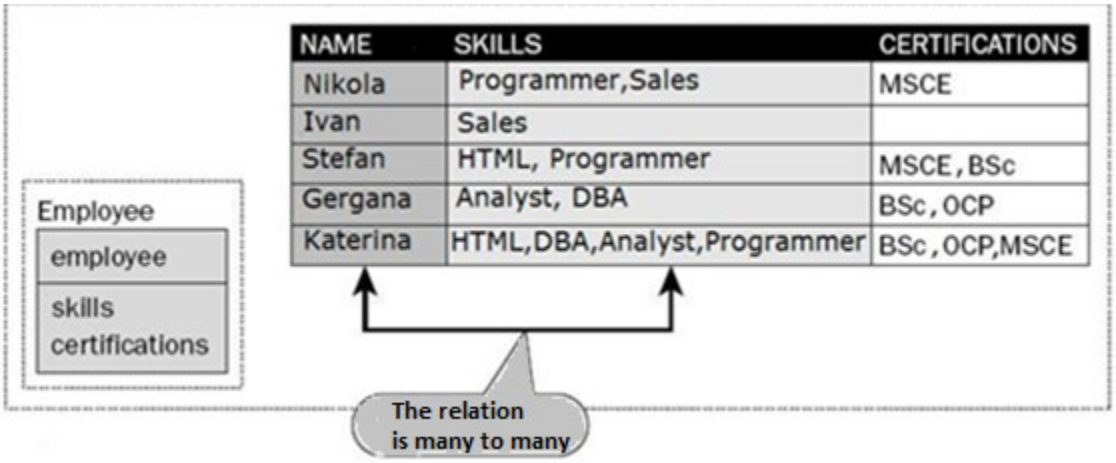
Dördüncü normal form (4NF)

Bir ilişki ancak çok değerli bağımlılıklar içermiyorsa 4NF'dedir.

Bir ilişkide üç nitelik (A, B ve C) olduğunda ve A'nın her değeri için iyi tanımlanmış bir B değerleri kümesi ve iyi tanımlanmış bir C değerleri kümesi olduğunda çok değerli bir bağımlılık vardır. B değerleri C'den bağımsızdır ve bunun tersi de geçerlidir.

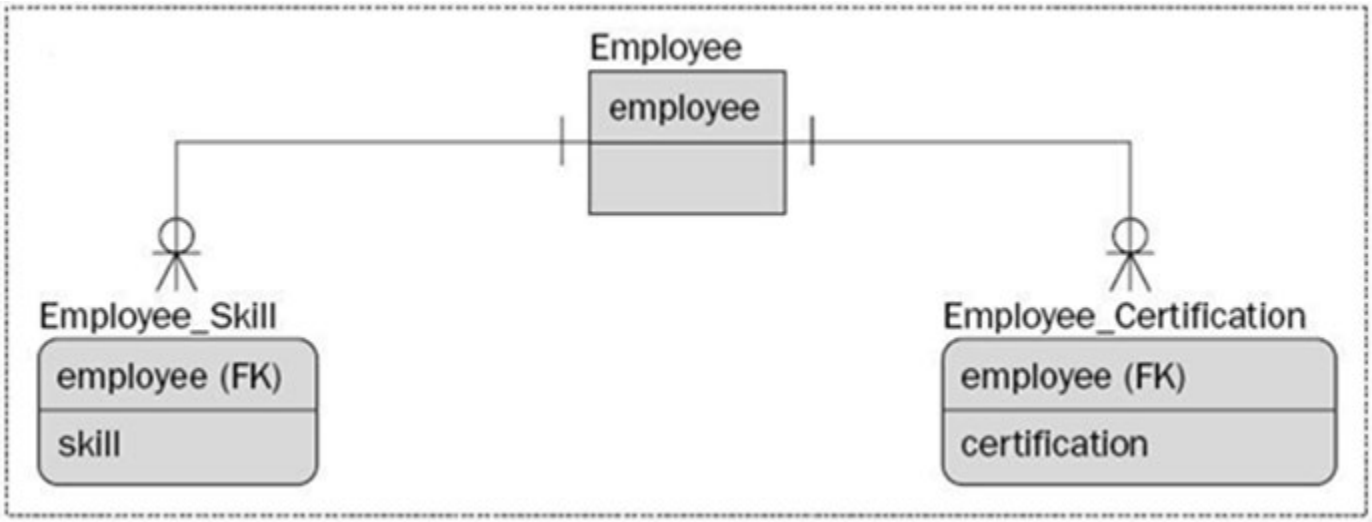
NORMAL FORMLAR

- Birden çok değerli alan, virgüllerle ayrılmış bazı türdeki değerlerin bir dizisini içeren bir alandır. Aşağıdaki şekil çalışanların sahip olduğu farklı becerilere ve sertifikalarına sahip Çalışan tablosunu göstermektedir. Beceri ve sertifikaların birbirinden bağımsız olmayıp, virgülle ayrılmış bir listeye eklendiğini de belirtmek gerekir.



NORMAL FORMLAR

- 4. normal formu uygulamak için, Çalışan tablosunu Employee_Skill, Employee ve Employee_Certification olmak üzere üç tabloya bölerek çok değerli bağımlılıkları kaldırıyoruz:



- Temel olarak 4. normal form, çok değerli öğelerin koleksiyonunu farklı tablo ve kayıtlara aktarır ve böylece her bir kayda doğrudan erişimi kolaylaştırır.

ÇÖZÜM

- İlişkisel şemaların normalleştirilmesi, belirli ilişkilerin değerleri ve içerikleriyle değil, doğrudan DB'deki verilerin anlambilimiyle ilgilidir.
- Belirli bir ilişkinin içeriğine göre işlevsel bağımlılıklar kurmak mümkün değildir ve bu da onun normal formunu kurmanın mümkün olmadığı anlamına gelir.
- DBMS, belirtilen işlevsel bağımlılıkları dikkate alarak veri bütünlüğünü koruması gereken sistemdir.
- Bununla birlikte, eğer bir ilişkisel şema 3NF'de ise, her ilişkideki elemanların benzersizliğini sağlamak için DBMS'nin yalnızca karşılık gelen ilişkinin anahtarını bilmesi yeterlidir.

ÇÖZÜM

- İlişkisel şema 3NF'de değilse, F bağımlılıklarının DBMS tarafından saklanması ve sürdürülmesi gerekir. Bu somut örneklerle gösterildi.
- Yukarıdaki hususlar, ilişkisel şemaların analiz edilmesi ve özellikle bunların 3NF'ye gönderilmesi ihtiyacını haklı çıkarmaktadır.

Bir sonraki bölümde ilişkisel şemaların normalleştirilmesine yönelik iki yaklaşım tartışılmaktadır.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Normalleşme Nedir?
2. Normalleştirmenin çeşitli biçimleri nelerdir?
3. Birinci Normal Form Nedir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

❑ Modül 2. Veritabanlarında ilişkisel yaklaşım

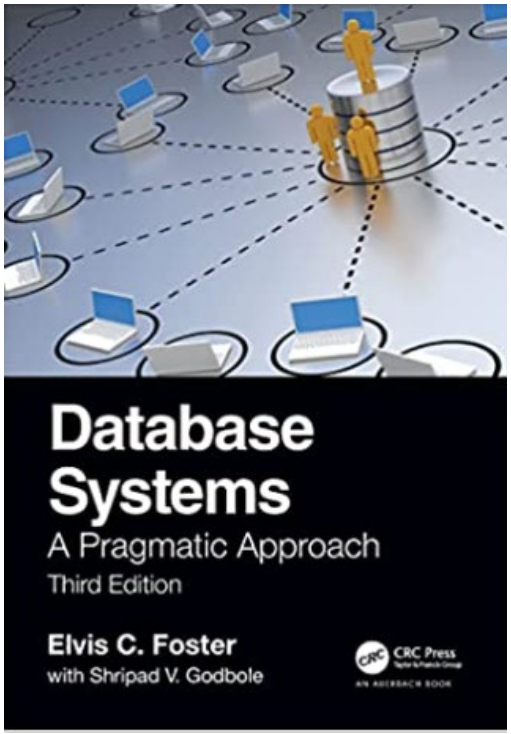
❑ Konu 4. Örnek bir veritabanının geliştirilmesi. Normalleştirme.

- ❑ Ders 2. İlişkisel şemaların ayrıştırılması ve sentezi. Ayrıştırma yoluyla normalleştirme algoritması. İlişkisel şemaların sentezi için algoritma.

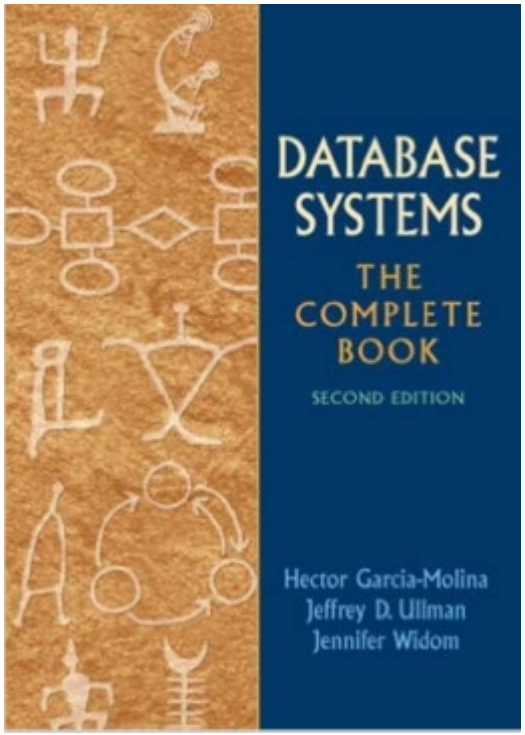
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

İLİŞKİSEL ŞEMALARIN AYRIŞTIRILMASI VE SENTEZİ. AYRIŞTIRMA İLE NORMALİZASYON ALGORİTMASI. İLİŞKİSEL ŞEMALARIN SENTEZİ İÇİN ALGORİTMA.



Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.



Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

Azalov, P. (1991) Database, Sofia.

Ayrıştırma yoluyla normalleştirme algoritması

- **İlişkilerin normalleştirilmesi için iki ana yöntem kullanılır - ayrıştırma ve sentez.**

Ayrıştırma yoluyla normalleştirme algoritması

R(A) ilişkisel bir şema olsun. *R* üzerinde bir ayrıştırma gerçekleştirmek, formun bir dizi ilişkisel şemasını tanımlamak anlamına gelir:

$$R' = \{R_1(A_1), R_2(A_2), \ldots, R_k(A_k) \mid A = \bigcup_{i=1}^{k} A_i\}$$

Başka bir deyişle, yeni ilişkisel şemalar *R*'nin bazı niteliklerine öznitelik olarak sahiptir ve bunların birleşimi *A* öznitelikleri kümesini verir ve herhangi iki ilişkisel şemanın özniteliklerinin kesişimi ya boş küme olabilir ya da boş olmayan olabilir.

Ayrıştırma yoluyla normalleştirme algoritması

- İlişkisel şema ayrıştırması yapma fikrinin amacı, *artıklık anormallikleri, güncelleme, ekleme ve silme anormallikleri* gibi istenmeyen anormalliklerin ortadan kaldırılması ihtiyacıdır.
- $R(A_1, A_2, A_3, A_4)$ belirli bir ilişkisel şema olsun ve R' onun ayrışımı olsun $R' = \{R_1(A_1, A_2), R_2(A_1, A_3, A_4)$. Ayrıca şu varsayılmaktadır:

1. r ilişkisel şemaya sahip bir ilişkidir ***R(A1, A2, A3, A4)***
2. $r_1 = \pi_{A1A2}(r)$, т.е. $r_1 : R_1(A_1, A_2)$
3. $r_2 = \pi_{A1A3\ A4}(r)$, т.е. $r_2 : R_2(A_1, A_3, A_4)$

Ayrıştırma yoluyla normalleştirme algoritması

- Bu şekilde formüle edilen varsayımlar altında, *r* ilişkisinin, r_1 ve r_2 projeksiyonları altında kurtarılabilir olduğunun söylenip söylenemeyeceği sorusunu sormak doğaldır.

- $s = r_1 \overset{\bowtie}{_{A_1}} r_2$, *olsun, s,* r_1 *ve* r_2 *'nin* A_1 *niteliğine göre doğal birleşimidir. Eğer ilişkisel şema R(A) ve buna karşılık gelen fonksiyonel bağımlılıklar kümesi F olan herhangi bir r için r=s ise, R(A)'nın ayrıştırılmasının F üzerinde kayıpsız bağlantı özelliğine sahip olduğu söylenir.*

Ayrıştırma yoluyla normalleştirme algoritması

- ***R(A)***, ***F*** fonksiyonel bağımlılıkları kümesine sahip bir ilişkisel şema olsun ve ***R'={R₁(A₁), R₂(A₂)}*** bunun bir ayrıştırması olsun.
- *r* bir ilişkisel şema R(A) ise ve r_i bir ilişkisel şema $R_i(A_i)$ ise, burada: $r_i = \pi_{Ai}(r)$, $i = 1, 2$.

hariç

1. $r \subseteq r_1 \bowtie r_2$
2. $r = r_1 \bowtie r_2$, only if
   $\{A_1' \cap A_2 \dashrightarrow A_1 \setminus A_2\} \in F^+$ or
   $\{A_1' \cap A_2^2 \dashrightarrow A_2^1 \setminus A_1^2\} \in F^+$

Co-funded by the Erasmus+ Programme of the European Union

Ayrıştırma yoluyla normalleştirme algoritması

- Bunu bazı örneklerle açıklayacağız.
- İlişkisel şema ÖĞRENCİ (Fac__numb, Name, Adres) dikkate alınır; burada Adres özelliği, öğrencinin çalışma sırasında yaşadığı adresi belirtir.
- Aynı adreste birden fazla öğrencinin yaşayabileceğini ve bir öğrencinin öğrenimi sırasında birkaç kez adres değiştirebileceğini dikkate alıyoruz.
- Bu, STUDENT ilişkisel şemasında tek bir işlevsel bağımlılığın belirtilmesi için bir neden verir:
- F={Fac_numb --→ Ad}

Ayrıştırma yoluyla normalleştirme algoritması

- Aşağıdaki ayrıştırma dikkate alınır:

$R' = \{R_1(\text{Fac\_numb}, \text{Ad}), R_2(\text{Fac\_numb}, \text{Adres})\}$

- STUD, STUDENT ilişkisel şeması ile bir ilişki olsun ve STUD1 ve STUD2 onun izdüşümleri olsun.

$$\text{STUD1} = \pi_{\text{Fac\_numb,Name}}(\text{STUD})$$
$$\text{STUD2} = \pi_{\text{Fac\_numb,Name}}(\text{STUD})$$

STUD	Fac_numb	Name	Address
	40251	Ivan Koychev	Sofia, Stud.grad, bl.25, A, room 128
	40326	Petar Dimov	Sofia, Stud. grad, bl.25, A, room 128
	40138	Ivan Koychev	Sofia, Vitosha bul, № 18, ap. 42

Ayrıştırma yoluyla normalleştirme algoritması

STUD1	Fac_numb	Name
	40251	Ivan Kouchev
	40326	Petar Dimov
	40138	Ivan Koychev

STUD2	Fac_numb	Address
	40251	Sofia, Stud. grad, bl. 25, A, room 128
	40326	Sofia, Stud. grad, bl. 25, A, room 128
	40138	Sofia, Vitosha bul, № 18, ap. 42

Ayrıştırma yoluyla normalleştirme algoritması

- STUD1 ve STUD2'nin doğal birleşimi *S* olsun

$$S = STUD1 \underset{Fac.number}{\bowtie} STUD2$$

- *S* = ÖĞRENCİ olduğu görülmektedir. *F*'de kayıpsız birleştirme ayrıştırması için koşullar olduğundan bu tesadüf değildir.

Really:

$$\{Fac\_numb, Name\} \cap \{Fac\_numb, Address\} = Fac\_numb$$

$$\{Fac\_numb, Name\} \setminus \{Fac\ numb, Address\} = Name$$

$$\{Fac\_numb \dashrightarrow Name\} \in F^{+} \text{ condition}$$

Co-funded by the Erasmus+ Programme of the European Union

Ayrıştırma yoluyla normalleştirme algoritması

- STUDENT ilişki şemasının diğer bir ayrıştırması şu şekildedir:

$R' = \{R_1(\text{Fac\_numb, Ad}), R_2(\text{Ad, Adres})\}$

- Yine STUD, STUDENT ilişkisel şeması ile bir ilişki olsun ve STUD1 ve STUD3 ilişkileri STUD'un izdüşümleri olarak aşağıdaki şekilde tanımlanır:

$$\text{STUD1} = \pi_{\text{Fac\_numb,Name}}(\text{STUD})$$
$$\text{STUD 3} = \pi_{\text{Name, Address}}(\text{STUD})$$

Ayrıştırma yoluyla normalleştirme algoritması

STUD3	Name	Address
	Ivan koychev	Sofia, Stud. grad, bl. 25, A, room 128
	Petar Dimov	Sofia, Stud. grad, bl. 25, A, room 128
	Ivan Koychev	Sofia, Vitosha bul., № 18, ap. 42

- STUD1 ve STUD3'ün doğal birleşimi STUD № ilişkisidir:

$$STUD_X = STUD1 \bowtie STUD3$$

Name

$STUD_X$	Fac_numb	Name	Address
	40251	**Ivan Koychev**	**Sofia, Stud.grad, bl. 25, A, room 128**
	40251	**Ivan Koychev**	**Sofia, Vitosha bul., № 18, ap. 42**
	40326	**Petar Dimov**	**Sofia, Stud.grad, bl. 25, A, room 128**
	40138	**Ivan Koychev**	**Sofia, Stud.grad, bl. 25, A, room 128**
	40138	**Ivan Koychev**	**Sofia, Vitosha bul., № 18, ap. 42**

Ayrıştırma yoluyla normalleştirme algoritması

- STUD_ ve STUD'un aynı olmadığı görülebilir. Ve bu gerçek tesadüfi değildir. Şunu fark etmeniz yeterlidir:

$$(Fac\_numb, Ad) \cap (Ad, Adres) = Ad$$

$$(Fac\_numb, Ad) / (Ad, Adres) = Fac\_numb$$

$$(Ad, Adres) / (Fac\_numb, Ad) = Adres$$

$$\{ Name \dashrightarrow Fac\_numb \} \notin F^+$$
$$\{ Name \dashrightarrow Address \} \notin F^+$$

- ***Ayrıştırmanın temeli, bir dizi fonksiyonel bağımlılık F'ye göre 3NF'de olmayan ilişkisel şemaları 3NF'de bulunan ilişkisel şemalara dönüştürmektir.***

Ayrıştırma yoluyla normalleştirme algoritması

- Bu dönüşüm iki adımda gerçekleştirilir. İlk adımda ilişkisel şemalar 2NF'e, ikinci adımda ise 3NF'e getiriliyor. İlk olarak ilişkisel şemaların 2NF'ye getirilmesi fikri tartışılacaktır.
- R(A), 2NF'de değil, 1NF'de olan belirli bir ilişkisel şema olsun. Bu şu demek:

a) a) bu ilişkisel şemanın anahtarı bileşiktir, K_1 ve K_2, $K_1 \in A$ ve $K_2 \in A$ ile gösterdiğimiz en az iki nitelikten oluşur.

б) A'nın X ve Y ile gösterdiğimiz iki alt kümesi vardır; $X \subset A$, $Y \subset A$, *(K1, K2) --->Y* ve *K1 --->X*, X özelliği anahtara tam işlevsel bağımlılık içinde değildir.

Ayrıştırma yoluyla normalleştirme algoritması

- Yukarıdaki açıklamalardan, 2NF'de olmayan ilişkisel şema $R(K_1, K_2, X, Y)$'yi iki ilişkisel şema ile değiştirmenin mümkün olduğu anlaşılmaktadır:

$$R_1(K_1, K_2, Y) \text{ and } R_2(K_1, X)$$

- *R1* ve *R2* 2NF'de değilse anlatılan prosedür tekrar uygulanmalıdır. Bu, orijinal ilişkisel şema, tamamı 2NF'de olan bir dizi ilişkisel şemaya dönüştürülene kadar devam eder. 2HF'de normalleştirme prosedürünü şu şekilde gösteririz:

```
DECOMP-2NF(A, R1, R2, F)
```

Ayrıştırma yoluyla normalleştirme algoritması

NEREDE

- *A* - orijinal ilişkisel şemadaki nitelikler kümesi;
- *R1, R2* - orijinal ilişkisel şemanın ayrıştırıldığı ilişkisel şemalar;
- *F* - A'daki işlevsel bağımlılıklar kümesi.
- İlişkisel şemaları 3NF'ye getirmek için aşağıdaki fikir uygulanabilir.
- *R(A)*, 2NF'de olan ancak 3NF'de olmayan *F* bağımlılıklarına sahip belirli bir ilişkisel şema olsun.

- Bu, *R(A)'*da geçişli fonksiyonel bağımlılıkların olduğu anlamına gelir; *A*'nın öznitelikler kümesi *K*, X_1, X_2 ve X_3 'ten oluşabilir; burada *K* bir anahtardır ve X_1, X_2, X_3 X_1 'in anahtar olmayan nitelikleridir --- $>X_2$ ve $\{X$ --- $>K) \in F^+$ X_3 özelliğinin mevcut olmaması göz ardı edilmemektedir.

Ayrıştırma yoluyla normalleştirme algoritması

Bu varsayımlar altında, orijinal ilişkisel şema $R(K, X_1, X_2, X_3)$, $R_1(K, X_1, X_3)$ ve $R_2(X_1, X_2)$olmak üzere iki ilişkisel şemaya ayrıştırılabilir. 2NF'den 3NF'ye normalizasyon prosedürünü şu şekilde gösterelim:

```
DECOMP-3NF(A,R1,R2,F)
```

- ***Ayrıştırma yoluyla tam normalleştirme algoritmasını*** düşünün. NORM_2(RA,F) ve NORM_3(RA,F) ile ilişkisel şema RA'nın sırasıyla ikinci ve üçüncü normal formda olup olmadığını kontrol eden iki Boole tipi işlevi belirtir.

Ayrıştırma yoluyla normalleştirme algoritması

```
const N =...;
type  A = (A1,A2,...,Ak);
      Attr = set of A;
      F-Dep = record
                X,Y:Attr
              end;
      FD = array[1..N] of F-Dep;

procedure DECOMP(RA:A;F:FD);
  var R1,R2 : A;
  begin
    if not NORM-2(RA,F)
      then begin
             DECOMP-2NF(RA,R1,R2,F);
             DECOMP(R1,F);
             DECOMP(R2,F)
           end
      else begin
             if not NORM-3(RA,F)
               then begin
                      DECOMP-3NF(RA,R1,R2,F);
                      DECOMP(R1,F);
                      DECOMP(R2,F)
                    end
           end
  end;
```

Ayrıştırma yoluyla normalleştirme algoritması

- İlişkisel şemaların ayrıştırma yoluyla normalleştirilmesine yönelik algoritmanın bazı **dezavantajları** vardır..
- Bu algoritmanın karmaşıklığı çok büyük. Bunun nedeni, ilişkisel bir şemadaki anahtar sayısının, özniteliklerin sayısı ve işlevsel bağımlılıkların sayısıyla birlikte katlanarak artması olabilir.
- Bir özelliğin **asallığını** kontrol etme görevi bundan daha basit değildir.
- Ayrıştırma bazen 3NF'de gerekenden çok daha büyük bir ilişkisel şemalar kümesiyle sonuçlanabilir.
- Bu düşünceler, ilişkisel şemaların normalleştirilmesi için **bir algoritmanın daha** dikkate alınmasına neden olmaktadır.

İlişkisel şemaların sentezi için algoritma

- Kısaca ilişkisel şema sentezi algoritmasının fikri şudur:
- İşlevsel bağımlılıklar kümesi *F* dikkate alınır ve bireysel bağımlılıklar sol taraflarında eşitliğe göre gruplandırılır.

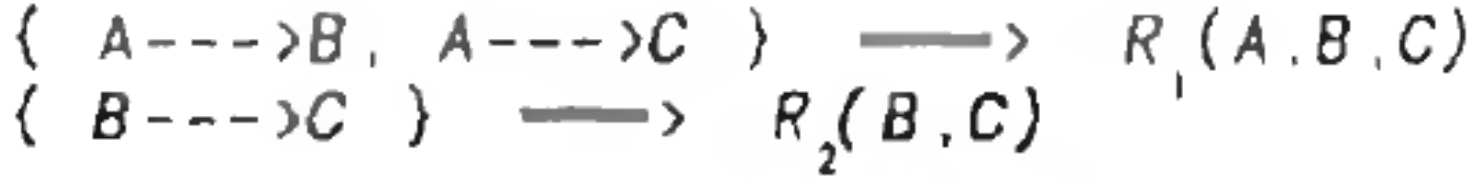
- Madde 1'den elde edilen her grup için, her bir grubun işlevsel bağımlılıklarında yer alan niteliklerin birleşiminden elde edilen bir dizi nitelik içeren ayrı bir ilişkisel şema tanımlanır.

- ***Örnek.*** {A, B, C} öznitelik kümesini ve F = {{A ---> B, A ---> C, B ---> C) işlevsel bağımlılıklar kümesini göz önünde bulundurun.

Co-funded by the Erasmus+ Programme of the European Union

İlişkisel şemaların sentezi için algoritma

- Algoritmanın 1. noktasından iki set *F* bağımlılığı elde edilir:



- 3NF'de olmadığı hemen anlaşılıyor, *A* ---> *B, B* ---> *C, A* ---> *C*'yi takip ettiğinden geçişli bir bağımlılık var.
- Alınan yanlış sonuç aşırı *F* bağımlılığından kaynaklanmaktadır. Bu yanlışlığı ortadan kaldırmak için orijinal algoritma fikrinde bazı iyileştirmeler yapılması gerekmektedir.

İlişkisel şemaların sentezi için algoritma

- $f = \{X \dashrightarrow Y\}$, F'nin işlevsel bir bağımlılığı olsun. Herhangi bir X' , $X' \in$ için Y, X'e tamamen işlevsel bağımlıysa, X niteliğine f'ye göre gereksiz olmayan bir nitelik adı verilir. X'ten şu sonuç çıkıyor:

$$((X-(X'))\dashrightarrow Y)\notin F^{+}$$

- Herhangi bir f, $f \in F$ için şunu takip ediyorsa, bir dizi fonksiyonel bağımlılık F'nin artıklık içermediğini hatırlatalım:

$$(F-(f))^{+} \neq F^{+}$$

- Bu iki kavram, normalleştirilmiş ilişkisel şemaların sentezi fikrinin belirlenmesinde büyük ölçüde rol oynar, dolayısıyla prosedürler olarak tanımlanırlar.

İlişkisel şemaların sentezi için algoritma

- REDUNDANT_ATR prosedürü, *B1* --- > *B2* fonksiyonel bağımlılığına göre bileşik öznitelik *B1*'deki tüm fazlalık öznitelikleri kaldırır. Bu prosedür, işlevsel bir bağımlılığın bir dizi işlevsel bağımlılığa üyeliğini belirleyen IS__MEMBER işlevini kullanır..

```
const   N =...;
        K =...;
type    A = (A1,A2,...,Ak);
        Attr = set of A;
        F_Dep = record
                    X,Y:Attr
                end;
        FD = array[1..N] of F_Dep;
```

İlişkisel şemaların sentezi için algoritma

```
procedure REDUNDANT_ATR(var B1:Attr;B2:Attr;F:FD);
  var I : integer;
      W : Attr;
      Z : A;
      f : F_Dep;
  begin
    Z:=A1;
    for I:=1 to K do
       begin
         if Z in B1
           then begin
                  W:=B1-[Z];
                   f.X:=W;
                   f.Y:=B2;
                   if IS_MEMBER(f,F)
                     then B1:=B1-Z
                end;
         Z:=succ(Z)
       end
  end;
```

İlişkisel şemaların sentezi için algoritma

- REDUNDANT_ATR prosedürünü F'nin fonksiyonel bağımlılıklarının her birinin sol tarafına uygulayarak, sol tarafları artıklık olmayan nitelikler olan bir dizi fonksiyonel bağımlılık G elde edilir. Bu yeni prosedürü ARTIK TÜM ATR(F, G) ile gösteriyoruz.
- REDUNDANT_ATR prosedürüne benzer şekilde, G'den gereksiz fonksiyonel bağımlılıkları kaldıran ve G'ye eşdeğer yeni bir H fonksiyonel bağımlılık seti ile sonuçlanan bir REDUNDANT_FDEP(G,H) prosedürü oluşturulabilir, yani G+=H+.
- Artık ilişkisel şemaları 3NF'de ***sentezlemeye yönelik algoritma*** açıklanabilir.

Co-funded by the Erasmus+ Programme of the European Union

İlişkisel şemaların sentezi için algoritma

1. REDUNDANT_ALL_ATR(F,G) prosedürü uygulanır.

2. REDUNDANT _FDEP(G,H) prosedürü uygulanır.

3. *H* fonksiyonel bağımlılıklar seti, her biri aynı sol tarafa sahip fonksiyonel bağımlılıklar içeren gruplara bölünmüştür.

4. Her bir işlevsel bağımlılık grubundan, işlevsel bağımlılıkların sol ve sağ taraflarında görünen niteliklerin birleşimini içeren bir ilişkisel şema tanımlanır. İlişkisel şemaların anahtarları, her grubun işlevsel bağımlılıklarının sol tarafında yer alan nitelikler olacak şekilde seçilir.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Bir ilişki şemasının ayrıştırılması nedir?
2. İlişkisel ayrıştırmanın özellikleri nelerdir?
3. İlişki şemasının sentezi nedir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

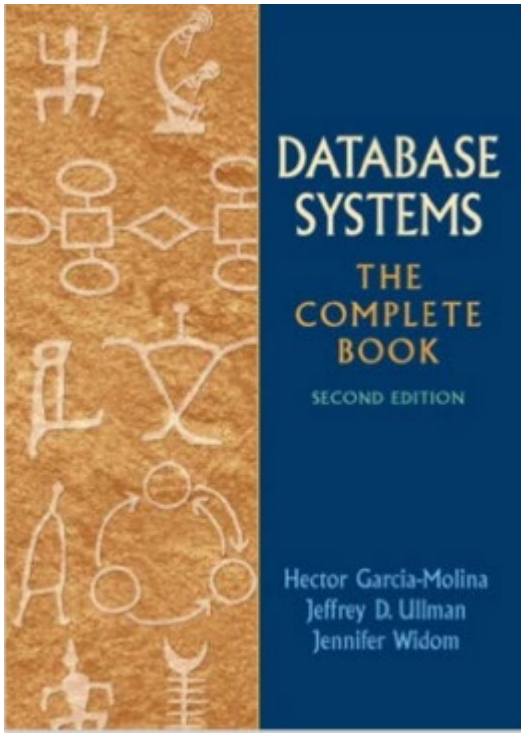
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

- **Modül 2. Veritabanlarında ilişkisel yaklaşım**
  - **Konu 5. Nesneye yönelik veritabanı sistemleri.**
    - Ders 1. Nesnelerin özü. Nesneye yönelik yaklaşımda temel kavramlar.

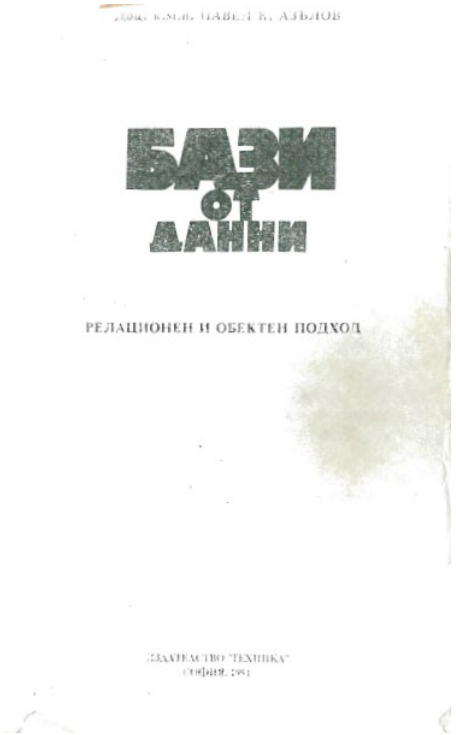
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

NESNELERİN ÖZÜ. Nesne yönelimli YAKLAŞIMDA TEMEL KAVRAMLAR.



Ullman, J., Widom, J (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.



Azalov, P. (1991) Database, Sofia.

NESNE DOĞALARI

- Genel olarak nesne kelimesi herhangi bir nesneyi ifade edebilir: bir kitap, bir bina, bir araba, bir işletme, bir şehir, bir bilgisayar vb. Nesneler maddi olabilir veya olmayabilir. Örneğin 19. ve 20. yüzyıllarda Avrupa'da yapılan kurtuluş savaşları tarihi objeler olarak görülebilir.
- Başka bir deyişle nesne doğrudan veya dolaylı olarak atıfta bulunulabilen her şeydir. Bilimin çeşitli alanlarında benzer genellemeler yapılmaktadır.
- İşte bilişimden bazı örnekler:
- 1. Nokta, doğru parçası, vektör, yay, daire, kesikli çizgi vb. genellikle **grafik nesne** adıyla özetlenir.

NESNE DOĞALARI

2. Tamsayı, gerçek büyüklükler basit nesnelerdir. İşaret ve Boole miktarları **basit nesne**lerdir.

3. Diziler, karakter dizileri, kayıtlar, kümeler, dosyalar vb. **bileşik nesne**lerdir.

4. Algoritmik yapılar dizisi, yineleme (döngü), ikili ve çok değişkenli seçim programlardaki **kontrol nesne**leridir.

5. Bir alt program aynı zamanda bir nesne olarak da görülebilir - yukarıda açıklanan nesneleri genelleştiren ve yalnızca *veri nesnelerinin* veya yalnızca bazı nesnelerin *işlem nesne*lerinin açıklamasını içeren veya aynı anda her iki türden nesneleri içeren bir **program nesnesi**.

Co-funded by the Erasmus+ Programme of the European Union

NESNE DOĞALARI

6. Metin parçası bir **metin nesnesi**dir, grafik görüntüsü bir **grafik nesnesi**dir, tablolar ve form da nesnelerdir ve hepsi ayrı ayrı ve belirli bir kombinasyon halinde daha karmaşık **bilgi nesnesi**ni - belgeyi - oluşturabilir

Bu örneklerin listesi uzatılabilir. Ancak bu genellemelerin her birine yüklenen anlama ve bundan elde edilebilecek faydaya bakmakta fayda var.

Bu nedenle **nesne kavramını** tanıtarak başlamak doğaldır.

Co-funded by the Erasmus+ Programme of the European Union

NESNE DOĞALARI

- **Nesne,** içine belirli davranış kuralları yerleştirilerek "ruhsallaştırılan" bazı veri türleri tarafından temsil edilen maddeden oluşur..
- Canlı organizmalar gibi nesneler de belirli bir ortamda bulunur ve bu durumda bu ortam ilgili *yazılım, bilgisayar ve telekomünikasyon sistemleri* tarafından sağlanır..
- Nesneler *etkin* (canlı olduğu varsayılır) veya *etkin olmayan* (canlı olmadığı varsayılır) olabilir. Aktif nesneler, yeni bir nesnenin "doğumu" veya bir nesnenin bir konumdan diğerine "taşınması" gibi çeşitli olaylara katılır.

Co-funded by the Erasmus+ Programme of the European Union

NESNE DOĞALARI

- Hem birbirlerine hem de dünyanın geri kalanına - kullanıcılara mesaj göndererek bilgi alışverişinde bulunarak "iletişim kurarlar".
- Tüm bu aktif yaşam, ayrıca **OM (Object Manager)** kısaltmasıyla anılan bir "**süper nesne**"nin rehberliği altındaki nesneler tarafından gerçekleştirilir.
- OM, nesnelere gömülü davranışa göre onların eylemlerini yöneten ve kontrol eden bir yazılım alt sistemidir.
- Kullanıcı ve nesne arasındaki bu tür bir ara seviye, kullanıcıyı tek tek nesnelere ilişkin belirli ayrıntıları bilmekten kurtarır.

NESNE DOĞALARI

- "Anatomik" anlamda her nesne bir kimlik parçası, bir vücut ve bir davranışla temsil edilir. "Baş" olarak da adlandırılan tanımlama kısmı, nesneyi diğer nesneler arasında tanımlar.
- Her nesnenin gövdesi bir dizi adlandırılmış özniteliklerden oluşur. Burada çeşitlilik çok fazla.
- Nesneler basit olabilir veya kendileri başka nesneler içerebilir, dolayısıyla bir **nesne hiyerarşisini** temsil edebilir. Her nesne, bir dizi kuralla temsil edilen bir davranışla karakterize edilir. Kurallar alt programlara benzer.

NESNE DOĞALARI

- Ancak aralarında bunları etkinleştirecek temel bir program yoktur. Kurallar, nesne yerel değişkenlerinin yanı sıra bu kuralların etkinleştirilebileceği koşulları da kullanır.
- Nesne yönelimli yaklaşımı kullanarak bir görevi çözmek, farklı nesnelerin oluşturulmasını gerektirir.
- Bunlardan bazıları verinin yapısı ve ona uygulanan karşılık gelen işlemlerle ilgili olarak benzer özelliklere sahip olacaktır. Bu, bu tür nesneleri ***sınıf*** adı verilen gruplar halinde gruplamak için bir önkoşuldur.
- Böylece her nesne bir sınıfın ayrı bir örneği olarak görülebilir.

Co-funded by the Erasmus+ Programme of the European Union

NESNE DOĞALARI

- Sınıf kavramının getirilmesi, aynı sınıftaki nesnelerin özelliklerinin tek bir tanımı için koşullar yaratır.
- Doğada her nesne genellikle bireyseldir ancak benzer nesnelerle ortak özelliklere de sahiptir.
- ***Örneğin***, bir elma sepetindeki her elma bir şekilde diğerlerinden farklıdır, ancak farklı türde nektar yapma tarifleri (yöntemleri) sepetteki veya dışındaki tüm elmalar için aynıdır.
- Bu anlamda tüm elmalar kümesi, örneğin armut sınıfından farklı, ayrı bir nesne sınıfı olarak görülebilir.

NESNE DOĞALARI

- Nesneleri sınıflara ayırma süreci bir tür soyutlamadır ve bu durumda ***genelleme*** olarak bilinir.

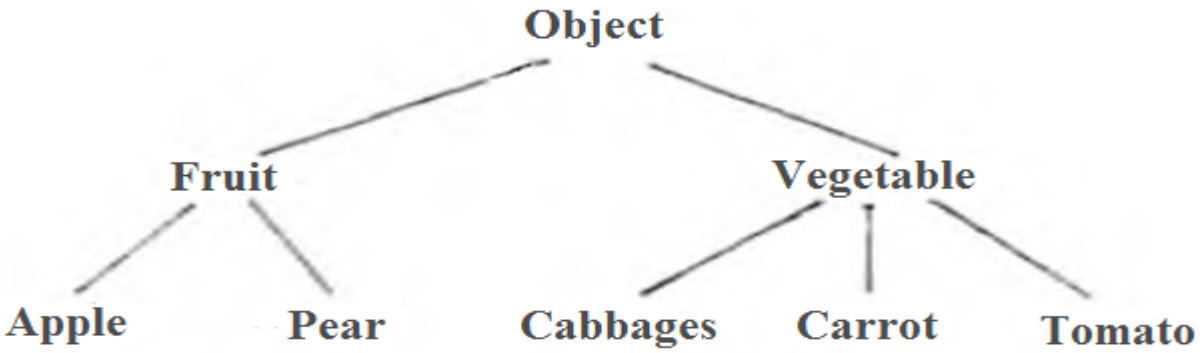


Fig. Example scheme of generalization of several object types

- Hiyerarşideki en yüksek sınıf, diğer tüm sınıflar için genelleştirici sınıf olan ***Object*** sınıfıdır.

NESNEYE YÖNELIK YAKLAŞIMDA TEMEL KAVRAMLAR

- Nesneye yönelik yaklaşımda temel kavram ***nesne*** kavramıdır. Bu kavramın tanımlanması zordur çünkü bir nesne yönelimli sistemden diğerine çok fazla değişiklik göstermektedir. Bu nedenle, çoğu zaman her yazar bunu belirtir ve ona belirli bir anlam verir.
- Nesne bilgidir ve işlenmesinin bir açıklamasıdır.
- Bir nesnenin, onu işlemek için "kapsüllenmiş" bir veri ve işlemler (yöntemler) birimi olduğu söylenebilir.
- Bir operasyonun uygulanması, onun nasıl uygulanacağına dair bilgi gerektirmez.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

NESNEYE YÖNELIK YAKLAŞIMDA TEMEL KAVRAMLAR

- Her nesne bir tür değer olarak düşünülebilir. *T* bazı veri türlerinden oluşan bir küme olsun ve *T1* ve *T2* iki tür *T* olsun.
- Eğer *T1* tipindeki her eleman (değer) *T2*'nin bir elemanı olarak kabul edilebiliyorsa, *T1*'in *T2*'nin bir ***alt tipi*** olduğu veya *T2*'nin de *T1*'in bir ***süper tipi*** olduğu söylenir.
- *T1* **alt tipi, üst tipi** *T2*'nin özelliklerini miras alır. Bu, *T2*'de tanımlanan veri yapılarının ve işlemlerinin, *T1* tipine yönelik veri yapıları ve işlemler olduğu anlamına gelir.
- Bir tür yalnızca başka bir türün alt türü olarak tanımlanabilir.

NESNEYE YÖNELIK YAKLAŞIMDA TEMEL KAVRAMLAR

- Böyle bir durumda ***basit kalıtımdan*** bahsediyoruz.
- Bir türün diğer birkaç türün alt türü olması da mümkündür ve bu durumda ***çoklu kalıtımdan*** bahsediyoruz.
- *T* türleri kümesindeki kalıtsallık, genellikle IsA ile gösterilen ikili bir ilişkidir. Asimetri, yansımama ve geçişlilik özellikleri onun için geçerlidir.
- IsA ilişkisi, *T*'de genellikle tür hiyerarşisi olarak adlandırılan bir hiyerarşiyi tanımlar.
- Sınıf terimi genellikle nesne yönelimli veritabanlarında tür anlamında kullanılır. ***Sınıfın*** farklı bir anlamı var.

NESNEYE YÖNELIK YAKLAŞIMDA TEMEL KAVRAMLAR

- Her ***sınıf***, zamanla değişen aynı türdeki nesnelerin bir kümesidir. Nesne sınıfları, gerçek dünyanın farklı konu alanlarındaki varlıkların model kümeleridir.
- *C* sınıfları kümesinde, asimetri, yansımama ve geçişlilik özelliklerinin geçerli olduğu ikili ilişkisel bir miras tanımlanabilir (SubsetOf ile gösterilir). Eğer *C1* sınıfı *C2*'nin alt sınıfı ise:

1. *C1, C2'nin* bir alt kümesidir.
2. *C1* eleman tipi, *C2* elemanlarının bir alt tipidir.

Co-funded by the Erasmus+ Programme of the European Union

NESNEYE YÖNELIK YAKLAŞIMDA TEMEL KAVRAMLAR

- SubsetOf ilişkisi, *C*'de ***sınıfların hiyerarşisi*** veya yalnızca ***sınıf hiyerarşisi*** adı verilen bir hiyerarşiyi tanımlar.
- **Kalıcılık** kavramı nesne yönelimli veritabanlarına özgüdür. **Kalıcılık**, herhangi bir nesnenin varoluş süresini belirleyen bir özelliğidir.
- Programlama dillerinde nesneler genellikle ilgili programın uygulanmasında oluşturulur ve bulunur.
- Bir sistemin, sistem varken oluşturulan nesnelerin **kalıcılığını** sürdürmesi gerekliliği, onu **nesne tabanı yönetimi** yetenekleriyle genişletmenin önemli bir önkoşuludur.

NESNEYE YÖNELIK YAKLAŞIMDA TEMEL KAVRAMLAR

- Nesneler kendilerini etkinleştiremez. Mesajlarla "harici olarak" etkinleştirilirler. Mesajlar prosedür odaklı sistemlerdeki fonksiyon ve prosedür çağrılarına benzer. Şunu işaret eden ifadelerdir:
- - mesajın alıcı nesnesi;
- - mesajın içeriği.
- Mesajın içeriği, bir argüman (nesne) listesi verilen alıcı nesnesine bir işlem uygulama talimatıdır.

NESNEYE YÖNELIK YAKLAŞIMDA TEMEL KAVRAMLAR

- Aynı işlemin farklı sınıflardaki nesnelere uygulanması doğaldır. Örneğin, hem tamsayılar hem de gerçek sayılar için "+" işlemini kullanmak veya farklı türdeki nicelikler için okuma ve yazma prosedürlerini kullanmak çok uygundur. Bu fikir nesne yönelimli sistemlerde yaygındır ve polimorfizm olarak bilinir.
- ***Polimorfizm***, aynı mesajların farklı nesnelere gönderilmesine olanak tanır; her biri kendi doğasına uygun bir şekilde "yanıt verir".
- Bir nesne ortamındaki her nesne diğerlerinden ayırt edilebilir. Bu, bireysel nesneyi benzersiz şekilde tanımlayan bir tanımlayıcı tarafından yapılır.

NESNEYE YÖNELIK YAKLAŞIMDA TEMEL KAVRAMLAR

- Bir nesnenin tanımlanması onun mevcut içeriğine, konumuna veya erişim şekline bağlı değildir.
- Nesne tanımlamasını sürdürmek, herhangi bir nesne sisteminin karakteristik bir özelliğidir. Genel olarak *nesne yönelimli sistemlerin temel özellikleri* şu şekilde özetlenebilir:
- 1. Nesneye yönelik bir sistem ortamında bir görevi çözmek şunları gerektirir:
- a) belirli bir görev için gereken nesne sınıflarının belirlenmesi;

NESNEYE YÖNELIK YAKLAŞIMDA TEMEL KAVRAMLAR

b) farklı nesne türlerine gerekli mesajların tanımlanması;

c) görevi çözen bir dizi mesaj oluşturmak.

2. İşlemler mesaj göndermekten ibarettir. Bir nesneye mesaj göndermenin sonucu da bir nesnedir.

3. Üst sınıflarının özelliklerini miras alan nesne sınıflarını tanımlamak için seçenekler mevcuttur.

4. Nesneye yönelik sistemler genişletilebilir. Sistem ve kullanıcı sınıfları eşdeğer statüye sahiptir.

NESNEYE YÖNELIK YAKLAŞIMDA TEMEL KAVRAMLAR

- Nesneye yönelik sistemlerin karakteristik özelliği olan yukarıdaki gereksinimlerin yalnızca bir kısmını karşılayan sistemler vardır.
- Bir sistem nesnelerin tanımlanmasına ve onlarla çalıştırılmasına izin veriyorsa, buna ***nesne tabanlı sistem*** denilmesi kabul edilir.
- Nesne tabanlı bir sistem, nesnelerin ayrı sınıflara gruplanmasına izin veriyorsa, o zaman ***sınıf tabanlı bir sistemdir***.
- Nesneye dayalı sistemlerin, özellikle nesneye dayalı sistemlerin bir alt kümesi olan ***sınıfa dayalı sistemlerin*** bir alt kümesi olduğu açıktır.

NESNEYE YÖNELIK YAKLAŞIMDA TEMEL KAVRAMLAR

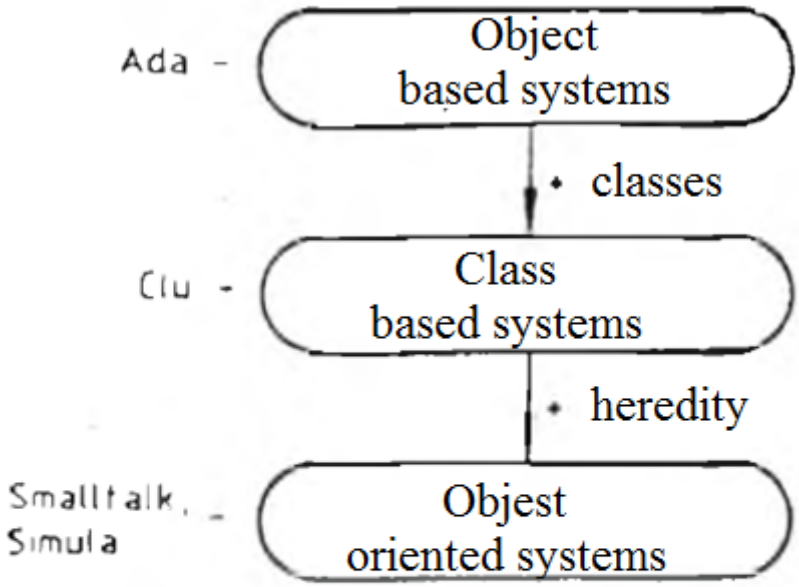


Fig. From object-based to object-oriented systems P.Wegner, OOPSLA, 168-181)

Co-funded by the Erasmus+ Programme of the European Union

ÇÖZÜM

- Veritabanlarında sınıflar arasında ifade edilebilen bağlantılar statik ve oldukça sınırlıdır. Çoğunlukla bireysel niteliklerin içeriğine atıfta bulunurlar ve ilişkisel sistemlerde iki veya daha fazla tablonun birleşimiyle uygulanırlar.
- Bireysel örnekler arasındaki bağlantılar, i. e. tablo satırları ifade edilemez. Bağlantılar yalnızca tüm tablo düzeyinde yapılır.
- Veritabanı yaklaşımından farklı olarak nesne yönelimli yaklaşımın özü, bir ortamda var olan tek bir nesnenin diğer nesnelerle bağlantılar kurarak etkileşime girebilmesi fikridir.

Co-funded by the Erasmus+ Programme of the European Union

ÇÖZÜM

- Bilgisayar sistemlerinin depolama ve aramaya yönelik temel yeteneklerinin kısa bir incelemesi, bunların geliştirilmesinin, bu sistemlerin kapsamının genişletilmesi, dolayısıyla uygulanabilecekleri konu alanlarının çeşitlendirilmesi ihtiyacıyla doğrudan ilişkili olduğunu göstermektedir.
- Bu, karmaşık gerçek dünya nesnelerinin bilgisayar modellerini oluşturma ihtiyacını doğurur ve bunları temsil etmek için karşılık gelen karmaşık veri yapılarını gerektirir.
- Bu sorunların çözümüne yönelik olası bir yaklaşım nesne yönelimli yaklaşımdır.

Co-funded by the Erasmus+ Programme of the European Union

ÇÖZÜM

- Veritabanı Yaklaşımı ile Nesneye Dayalı Yaklaşım arasında önemli ve çeşitli uyumsuzluk noktalarının olduğu açıkça görülmektedir. Bununla birlikte, bunların birlikte uygulanabileceği ve zaten uygulanmakta olduğu ortaya çıktı.
- Nesne yönelimli veritabanlarında DBMS'nin nesne yönelimli kavramlarla genişletilmesi önerilmektedir.
- Bunlar, soyut veri türlerinin tanıtılması, karmaşık nesnelerin işlenmesi ve nesne sınıflarının sınıf hiyerarşisi tarafından oluşturulan mirasın kullanılması olabilir.

Co-funded by the Erasmus+ Programme of the European Union

ÇÖZÜM

- Yaklaşımın önemli bir avantajı, büyük miktarda yapılandırılmış verinin işlenmesinde verimlilik sağlayan "Veritabanları" yaklaşımına yakın kalmasıdır.
- Öte yandan bu yaklaşım, nesnelerle çalışma yeteneğinin de eklendiği uygulama yelpazesini genişletmektedir.

Veritabanı Yaklaşımı ile Nesneye Dayalı Yaklaşımın tam olarak ortak kullanımına yönelik çaba, nesne yönelimli ve uygun sorgu arama olanaklarına sahip sistemler oluşturmaktır.

Sorular ve alıştırmalar:

1. "Nesne" kelimesinin açıklamasını yapabilir misiniz?
2. Nesneleri sınıflara ayırma sürecini nasıl adlandırırız?
3. Nesneye yönelik yaklaşımda temel kavram nedir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

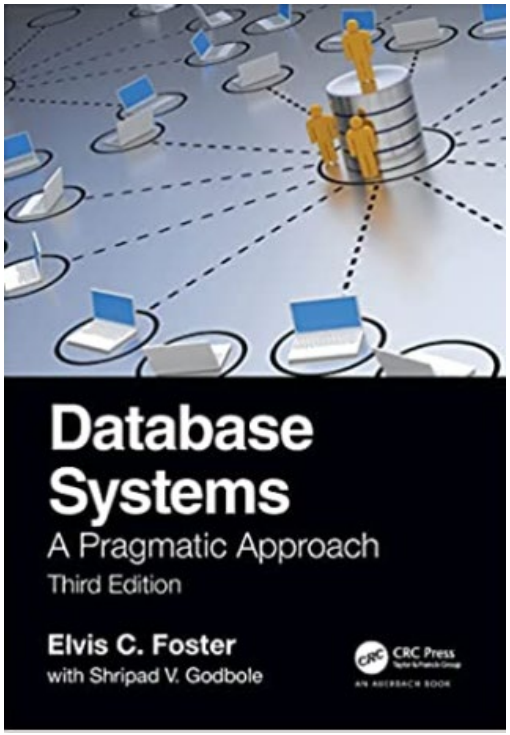
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

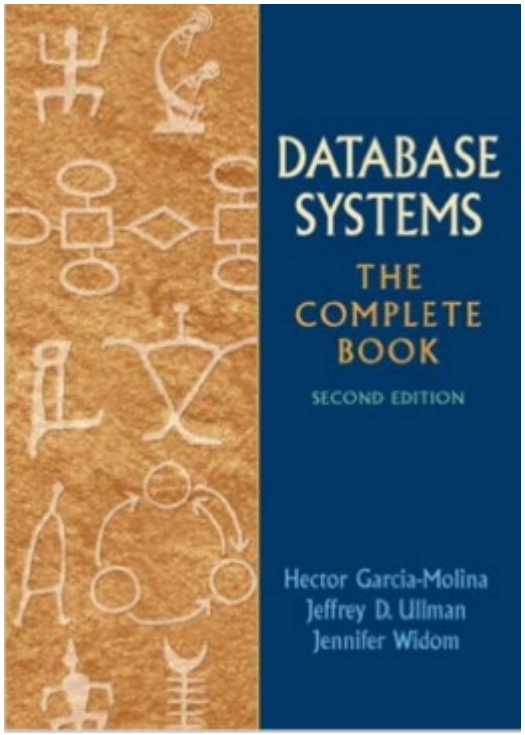
- **Modül 2. Veritabanlarında ilişkisel yaklaşım**
  - **Konu 5. Nesneye yönelik veritabanı sistemleri.**
    - Ders 2. Nesneye yönelik veritabanı yönetim sistemleri. Mimari.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

NESNE ODAKLI VERİTABANI YÖNETİM SİSTEMLERİ. MİMARİ.





Elvis C. Foster, Shripad V. Godbole. (2022) Database Systems. A Pragmatic Approach (3rd edition), CRC Press.

Ullman, J., Widom, J. (2009) DATABASE SYSTEMS The Complete Book (2rd ed), Upper Saddle River, New Jersey.

Azalov, P. (1991) Database, Sofia.

Nesneye yönelik DBMS(Veritabanı Yönetim Sistemi)

- **Nesneye yönelik** yaklaşımın uygulama alanlarından biri de veritabanlarıdır.
- Bunun temel nedeni, uzmanların **DBMS**'nin uygulanabilirlik kapsamını yapay zeka, görüntü ve ses işleme sistemleri, ofis sistemleri ve diğerleri gibi alanlarda genişletme arzusudur.
- Modern **DBMS**'lerin doğrudan bu amaçlarla kullanılamamasının çeşitli nedenleri vardır ancak bunlardan en önemlilerini şu şekilde özetlemek mümkündür:
- a) ***karmaşık veri nesnelerini*** temsil etme yeteneklerinin eksikliği;

Nesneye yönelik DBMS(Veritabanı Yönetim Sistemi)

b) modellenen gerçek nesnelerin davranışını ifade etmek için uygun araçların bulunmaması;

*c) modellenen konu alanının nesneleri ile meydana gelebilecek olayları kaydeden ve **işleyen mekanizmaların** eksikliği.*

DBMS'de belirtilen eksikliklerin üstesinden gelmek için bilinen iki yaklaşım vardır. Bunlardan biri **olan "aşağıdan yukarıya" yaklaşım**la amaç, geleneksel DBMS'yi nesne odaklı bir konseptle ve ilk etapta karmaşık bilgi nesnelerinin bakımıyla genişletmektir.

Nesneye yönelik DBMS(Veritabanı Yönetim Sistemi)

- **"Yukarıdan aşağıya" yaklaşım** olarak adlandırılan diğeri, nesne yönelimli programlama dillerini, var olmaya devam eden nesnelerin kalıcı nesnelerini ve programın yürütülmesinin tamamlanmasından sonra da sürdürme yeteneği ile genişletmeyi amaçlamaktadır.
- Her iki yaklaşımın da avantajları ve dezavantajları vardır. Bununla birlikte, nesne yönelimli bir DBMS'nin bu yaklaşımların ana fikirlerinin bir "sembiyozu" olması gerektiği konusunda oybirliğiyle bir görüş vardır. Bu, bir sistemin iki temel gereksinimi karşılaması durumunda nesne yönelimli bir DBMS (OODBMS) olduğunu varsaymamıza neden olur:

Nesneye yönelik DBMS(Veritabanı Yönetim Sistemi)

1. *Terimin geleneksel anlamında bir DBMS'dir.*
2. *Nesne yönelimlidir, nesne sınıflarını, kalıtım ve polimorfizmi, nesne tanımlamayı ve genişletilebilirliği destekler.*

TÜRLER, SINIFLAR VE NESNELER

Tür kavramı modern programlama dillerinde temeldir. Bir **tür,** değerler adı verilen bir dizi öğeyi ve bu değerlere uygulanan karşılık gelen işlemler kümesini birleştirir.

- ***Basit*** (atomik) ve ***bileşik*** (yapısal) olmak üzere iki tür tip kullanılır. Kompozit tipler hem basit hem de kompozit tipleri içerir.

Co-funded by the Erasmus+ Programme of the European Union

Nesneye yönelik DBMS(Veritabanı Yönetim Sistemi)

- Bu şekilde bir **tür** hiyerarşisi oluşturmak mümkündür. Bu amaçla dizi, liste, kayıt ve set gibi yapılandırıcılar kullanılır.
- Her değişken, aynı zamanda kendi **tür**ü olduğu varsayılan belirli bir **tür**ün değerlerini alan bir nicelik olarak kabul edilir.
- Bazı ortak özelliklere rağmen ***sınıf*** kavramı tip kavramından farklıdır. **Türler** esas olarak derleme sırasında programın doğruluğunu kontrol etmek için kullanılırken, ***sınıflar*** nesneleri oluşturmak ve sürdürmek için tasarlanmıştır.
- **Tür**lerin yanı sıra ***sınıflar*** da değerleri ayrı kümeler halinde sınıflandırır.

Nesneye yönelik DBMS(Veritabanı Yönetim Sistemi)

- Ancak **tür**lerden farklı olarak **sınıflar**, **nesneler** adı verilen, gerçekten oluşturulmuş örnekleri içeren başka bir kümeyi (sınıf uzantısı) destekler.
- Her **nesne,** yapısını tanımlayan belirli sayıda değişkenden oluşur. Bazı nesne yönelimli sistemlerde değişkenlerin türü programın yürütülmesi sırasında belirtilebilir.
- Bu doğal olarak daha fazla esneklik yaratır ancak hangi yöntemlerin (prosedürlerin) kullanılacağının önceden belli olmaması nedeniyle geç bağlantı adı verilen mekanizma için bir mekanizma gerektirir.

Nesneye yönelik DBMS(Veritabanı Yönetim Sistemi)

- **Nesne** yönelimli sistemlerde nesneler kapsüllenir. Bu, her nesnede iki ana bölümün varlığıyla ifade edilir - bir arayüz ve bir gerçekleştirme (uygulama) bölümü.
- Arayüz kısmı, **nesneye** uygulanabilecek bir dizi işlemi temsil eder.
- Uygulama kısmı, **nesnenin** yapısını ve her bir işlemin uygulanma şeklini açıklar.
- **Nesnelere** yalnızca arayüz kısımlarından erişilebilir, uygulama kısmı ise "gizlidir", erişilemezdir. Bu sayede nesneleri kullanan programları etkilemeden uygulama kısmında değişiklik yapılabilmesi için koşullar yaratılmış olur.

Nesneye yönelik DBMS(Veritabanı Yönetim Sistemi)

- Uygulama programlarının uygulama kısmındaki değişikliklerden korunması, uygulama bağımsızlığı veya veri bağımsızlığı olarak adlandırılan nesne yönelimli sistemlerin bir özelliğidir.

KARMAŞIK NESNELER VE GENİŞLETİLEBİLİRLİK

Temeli DBMS'nin kullanıldığı bazı standart dışı uygulamaların uygulanmasında, sistemin hem **karmaşık veri nesnelerini** tanımlama hem de bunların işlenmesine ilişkin ilgili işlemler için yeteneklerle genişletilmesine ihtiyaç vardır.

Karmaşık veya **bileşik veri nesneleri**, aynı zamanda karmaşık olabilen diğer nesnelerden oluşur.

Co-funded by the Erasmus+ Programme of the European Union

Nesneye yönelik DBMS(Veritabanı Yönetim Sistemi)

- **Karmaşık nesnenin** yapısı, çeşitli kurucular ve bunların bireysel bileşenler (alt nesneler) üzerine uygulanma sırası tarafından belirlenir.

- İlişkisel model, **karmaşık nesnelerin** tanımlanmasına ve bunlarla işlem yapılmasına izin vermez. İçinde izin verilen kurucular yalnızca kayıt (n-tuple) ve settir, ancak iç içe geçmelerine izin verilmez.

- Nesne yönelimli veri modellerindeki nesneler, kendilerini oluşturan bileşenlerin yapısında sınırlama olmayan **karmaşık nesnelerdir**.

Nesneye yönelik DBMS(Veritabanı Yönetim Sistemi)

- **Karmaşık nesnelerin** kullanımı, bunlar üzerinde çalışmak için ilgili yöntemlerin uygulanmasını gerektirir. Nesne yönelimli bir DBMS, nesne yönelimli programlama sistemi ile klasik bir DBMS'nin yeteneklerini birleştirdiğinden, bu işlemlerin programlama ortamı tarafından sağlandığı varsayılabilir.
- Ancak bu gibi durumlarda veri tabanından alınan verilerin ortama uygun forma dönüştürülmesi gerekli olacak ve bu da çok fazla zaman kaybına yol açacaktır. Sonuç olarak, karmaşık nesne işleme işlemleri, **DBMS** işlemlerinin doğal bir **uzantısın**ı ve tamamlayıcısını temsil etmelidir.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

Nesneye yönelik DBMS(Veritabanı Yönetim Sistemi)

NESNELERİN TANIMLANMASI VE KARARLILIĞI

Nesne yönelimli DBMS'de nesneler birbirinden ayırt edilebilir. Bu nedenle ***kimliklerinin belirlenmesi*** zorunluluğu.

- Programlama dillerinde nesnelerin ***tanımlanması*** adreslemeye ve DBMS'de birincil anahtarın değerine indirgenirken, nesne yönelimli DBMS'lerde nesnelerin ***tanımlanması*** onların fiziksel konumlarına veya özniteliklerinin belirli değerlerine bağlı değildir.
- Birçok sistemde, ***nesne tanımlama*** sorunu, her nesnenin, onu nesne ortamı boyunca benzersiz bir şekilde tanımlayan ve fiziksel konumundan bağımsız olan bir tanımlayıcısının sistematik olarak üretilmesiyle çözülür.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

Nesneye yönelik DBMS(Veritabanı Yönetim Sistemi)

- Nesne yönelimli DBMS'lerde bazı nesnelerin **kalıcılığı**, sistemdeki başka bir çalışma "oturumundan" sonra "var olma" yetenekleriyle ifade edilir, yani. tekrar tekrar kullanılabilirler.
- Nesnelerin sürdürülebilirliğine ilişkin önemli bir sorun, bazı nesnelerin **sürdürülebilir** olduğu, bazılarının ise sürdürülebilir olmadığı koşulların tanımlanmasıdır.
- Bu soruna farklı çözümler mümkün. Yaratılan her nesnenin **sürdürülebilir** hale geldiği varsayılabilir. Ancak bu, ara sonuç olarak elde edilen nesnelerin de bu özelliğe sahip olacağı anlamına gelir.

Nesneye yönelik DBMS(Veritabanı Yönetim Sistemi)

- Diğer bir olasılık ise **sürdürülebilirliği** sınıf düzeyinde belirlemektir, yani yalnızca bireysel sınıfların nesneleri **sürdürülebilir** olacaktır. Bu seçeneğe ek olarak, diğer sınıflardaki nesnelerin de sürdürülebilir olarak özel olarak belirtilmesi olanağı sağlanabilir.
- Üçüncü olasılık ise nesnenin **sürdürülebilirliğinin**, elde edildiği operasyonlara bağlı olmasıdır.

SORGU DİLLERİ

Klasik DBMS'de sık karşılaşılan bir durum, verinin asıl işlenmesinin Fortran, Pascal veya C gibi genel amaçlı bir programlama dili aracılığıyla anlatılması ve veri tabanından verilere erişimin bir veri aracılığıyla anlatılmasıdır. SQL gibi bir alt dil.

Nesneye yönelik DBMS(Veritabanı Yönetim Sistemi)

- Yukarıdaki yaklaşımın *önemli dezavantajları* vardır ve bu, nesne yönelimli DBMS geliştirmede dikkate alınması gereken hususlardan biridir.
- Nesne yönelimli DBMS'lerde sorgulama dillerine getirilen gereksinimler, çoğunlukla klasik DBMS'lere yönelik gereksinimlerdir.
- Daha önemli olanlara kısa bir genel bakış aşağıda verilmiştir:

1. Sorgu dili üst düzey bir dil olmalıdır. Bu, genellikle karmaşık bir nesne olan nesne türünden bağımsız olarak, sorgu dilinin işlenmesi için basit ifadeler sağlaması gerektiği anlamına gelir.

Nesneye yönelik DBMS(Veritabanı Yönetim Sistemi)

2. Sorgu dili belirli uygulamalardan bağımsız olmalı, yani ele alınan konu alanıyla ilgili olarak evrensel olmalıdır.

3. Sorgu dilinin kapatılması. Sorgu dili, nesne işleme araçlarına ek olarak sınıf hiyerarşisi ve kalıtımı oluşturma ve sürdürme, karmaşık nesneleri tanımlama, bazı işlemlerin sonucu olarak elde edilen nesneleri yönetme vb. araçları da sağlamalıdır.

Nesneye yönelik DBMS(Veritabanı Yönetim Sistemi)

4. Dilin gücünü sorgulayın. İlişkisel dillerin ifade gücüne ek olarak, nesne yönelimli DBMS'deki diller, özyinelemeli nesnelerin tanımlanmasına ve işlenmesine ve aynı zamanda karşılık gelen sınıf hiyerarşisi boyunca gezinmeye izin vermelidir.

*5. **Sorgu optimizasyonu**. Nesne yönelimli DBMS'deki sorgu dili, sistem optimizasyonu fırsatlarına izin vermelidir.*

Nesne yönelimli DBMS mimarisi

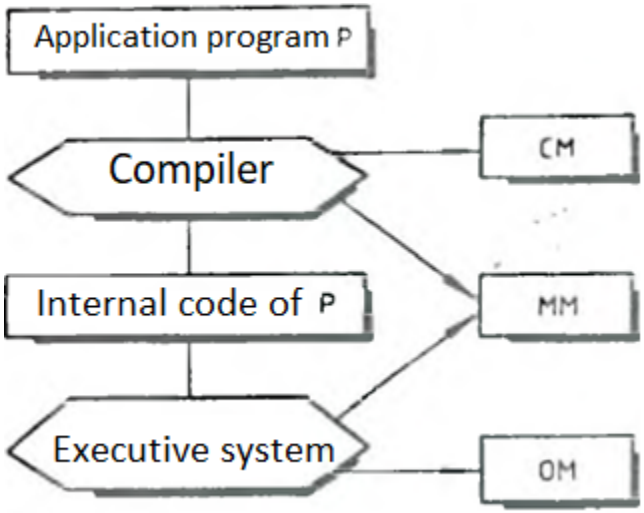
- Nesneye yönelik bir DBMS'nin klasik bir DBMS'ye göre sahip olduğu avantajlar, bu tür sistemler oluşturulurken aşılması gereken yeni ve zor sorunların ortaya çıkmasına neden olmaktadır.
- Her ne kadar nesne yönelimli DBMS'nin mimarisi ve işlevlerine ilişkin hâlâ birleşik bir kavram olmasa da, halihazırda oluşturulmuş DBMS'lerin bir kısmına dayanarak, onları oluşturan ana bileşenlere bakacağız.

TEMEL BİLEŞENLER

Nesneye yönelik bir DBMS'nin temel bileşenleri çoktur ve bunların her biri ayrı bir alt sistem olarak görülebilir.

Nesne yönelimli DBMS mimarisi

- Bunlar, daha önce **OM (nesne yöneticisi)** olarak anılan nesnelerin yöneticisi, **MM** yöntemlerin yöneticisi **(yöntem yöneticisi)** ve **CM -** sınıfların yöneticisi (**sınıf yöneticisi**).



Nesne yönelimli bir DBMS'nin temel bileşenleri

Nesne yönelimli DBMS mimarisi

- **OM**'nin en önemli işlevleri veritabanındaki nesneleri oluşturmak ve yok etmek, depolamak ve sürdürmektir. **OM**'nin etkinleştirilmesi, **OM**'nin mesajın adını ve ona katılan nesnelerin tanımlayıcılarını bildiği mesajların gönderilmesiyle gerçekleştirilir.
- Belirtilen yöntemi yürütmek için **OM**'nin öncelikle nesnelerin ait oldukları sınıflara ait olduğunu belirlemesi gerekir.

Nesne yönelimli DBMS mimarisi

- **CM**'nin görevleri, nesnelerin sınıf tanımlarını oluşturmak ve saklamak ve bunlar arasında ilişkiler kurmak, böylece sınıfların sınıf hiyerarşisini oluşturmaktır.
- Klasik bir DBMS'nin işlevlerine atıfta bulunulduğunda, **CM**'nin işlevleri, DBMS kataloğunun bakımına kadar bunlara yakındır.
- **MM**'nin amacı kaynak ve nesne kodunda yöntemler oluşturmak, depolamak ve sürdürmektir.
- Yöntemler sistem (sistemin kendisinde yerleşiktir) ve kullanıcıdır.

Nesne yönelimli DBMS mimarisi

- Yöntemler, belirli bir uygulama için ihtiyaç duyulan yeni oluşturulan sınıflar içerisinde kullanıcı tarafından tanımlanır. Kullanıcı yöntemleri belirli bir ***nesne yönelimli programlama dilinde*** yazılmıştır.
- **Nesne yönelimli DBMS mimarilerinin** altında yatan kavram çeşitliliğini tek bir noktada birleştirmek zordur. Bu nedenle burada sadece bunların karakteristik bileşenleri sunulmaktadır.
- Nesnelerin yalnızca ilgili programın yürütülmesi sırasında var olduğu nesne yönelimli programlama dillerinden farklı olarak, **nesne yönelimli DBMS**'ler kalıcı nesnelerle çalışma olasılığı ile karakterize edilir.

Co-funded by the Erasmus+ Programme of the European Union

Nesne yönelimli DBMS mimarisi

- Her nesnenin, hem RAM'de hem de harici bellekte erişilebilen kendi tanımlayıcısıyla benzersiz bir şekilde tanımlandığı varsayılır.
- Birçok uygulamada bu tanımlayıcı bir çift addır; bir dosyanın adı ve nesneyi içeren bir kaydın adı. Dört bayt olarak kodlanabilirler.
- Dosya adı **11** bit ve kayıt adı **21** bit olmak üzere dosya sayısı **2048**'e kadar çıkabileceği gibi her birindeki kayıt sayısı da iki milyonun üzerinde olabilir.
- Harici bellekte bulunan nesneler **RAM**'dekilerle aynı duruma sahiptir. Bu amaçla, işletim belleğinde bulunan nesneler özel bir tabloya kaydedildiği için sanal bir bellek tutulur.

Nesne yönelimli DBMS mimarisi

- Her nesne fiziksel olarak iki parçadan oluşur. Bunlardan biri sistemiktir ve şunları içerir: nesnenin ait olduğu sınıf hakkında bilgi; söz konusu nesnenin bağlı olduğu nesnelerin bir sayacı, bunların bir alt kümesidir ve nesnenin değişikliğe uğrayıp uğramadığını ve nesneye imhası için komut gönderilip gönderilmediğini belirten iki bayrak.
- Diğer nesnelerin n-tuple'ları olan veya nesne kümelerini temsil eden nesneler, kendilerini oluşturan nesnelerin yalnızca temel değerlerini veya tanımlayıcılarını içerir.

Nesne yönelimli DBMS mimarisi

- Kümelerin yapılar olarak değişebilirliği nedeniyle, bunların nesne uzayında temsili, kayıtları değişken veya belirsiz uzunlukta olan dosyaları işlemek için uygun bir bellek ayırma stratejisinin yanı sıra uygun dosya sistemi yeteneklerini gerektirir.

İlişkisel yaklaşımı benimsemenin ve ilişkisel şemaların kullanımının basitliğine rağmen bazı yapısal sınırlamalar yeni arayışlara neden olmaktadır. Bunlardan biri, daha zengin anlambilime sahip yeni veri modellerinin geliştirilmesidir. Bunlar örneğin Varlık-İlişki veri modeli, Semantik Veri Modeli ve ***Nesneye Dayalı Veri Modelidir****.*

Co-funded by the Erasmus+ Programme of the European Union

OODBMS'in Faydaları

01.	**Reusability of Code:** A tested system component can be reused in the design of another component.
02.	**Stability and Reliability:** Software can be constructed from tested components. Organizations can be assured of guaranteed performance of software.
03.	**Increased Sophistication:** More complex systems can be constructed.
04.	**Understandability:** Designer and user think in terms of object and behavior rather than low-level functional details. This results in more realistic modeling that is easier to learn and communicate.
05.	**Faster Design:** Most RAD tools and contemporary CASE tools are object-oriented to some degree. Also, code reusability enhances faster development.
06.	**Higher Quality Design:** New software can be constructed by using tested and proven components.
07.	**Easier Maintenance:** Since systems are broken down into manageable component objects, isolation of system faults is easy.
08.	**Dynamic Lifecycle:** I-OO-CASE tools integrate all stages of the software development life cycle (SDLC).
09.	**Interoperability:** Generic classes may be designed for multiple systems.
10.	**Design Independence:** Classes may be designed to operate and/or communicate across different platforms.
11.	**Clarity:** OT promotes better communication between IS professionals and business people.
12.	**Better CASE Tools:** OT leads to the development of more sophisticated and flexible CASE and RAD tools.
13.	**Better Machine Performance:** OT leads to more efficient use of machine resources.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. OODBMS ile ilgili aşağıdakilerden hangisi doğrudur?

   A. Karmaşık veri türlerini Web'de saklama yeteneğine sahiptirler.

   B. Tüm uygulamalarda RDBMS'yi geride bırakıyorlar.

2. Aşağıdakilerden hangisi aynı türdeki elemanların sıralı bir koleksiyonudur?

   A. Ayarla

   B. Çanta

   C. Liste

   D. Sözlük

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

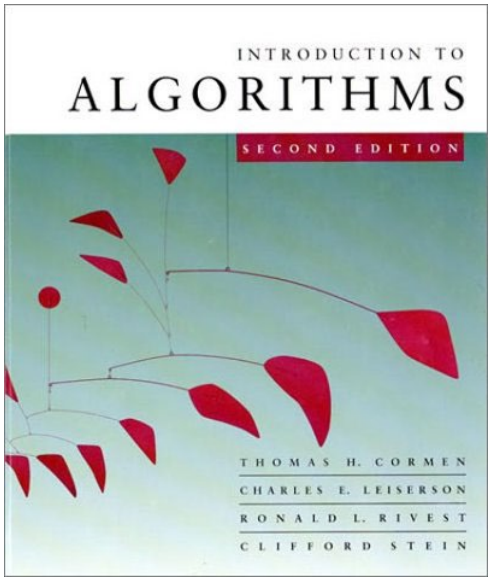
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

- **Modül 3. Sorgu optimizasyonu için algoritmalar ve veritabanlarındaki uygulamaları**
  - **Konu 1. Algoritmalara giriş**
    - Ders 1. Hesaplamada algoritmaların rolü. Bir teknoloji olarak algoritmalar.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

HESAPLAMADA ALGORİTMALARIN ROLÜ. BİR TEKNOLOJİ OLARAK ALGORİTMALAR.



Cormen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2009). Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill.

Co-funded by the Erasmus+ Programme of the European Union

Giriş

- Algoritmalar nelerdir?
- Algoritma çalışmak neden faydalıdır?
- Bilgisayarlarda kullanılan algoritmaların rolü nedir?
- Bu dersimizde bu soruların cevaplarını vermeye çalışacağız.

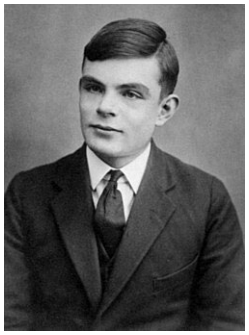
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

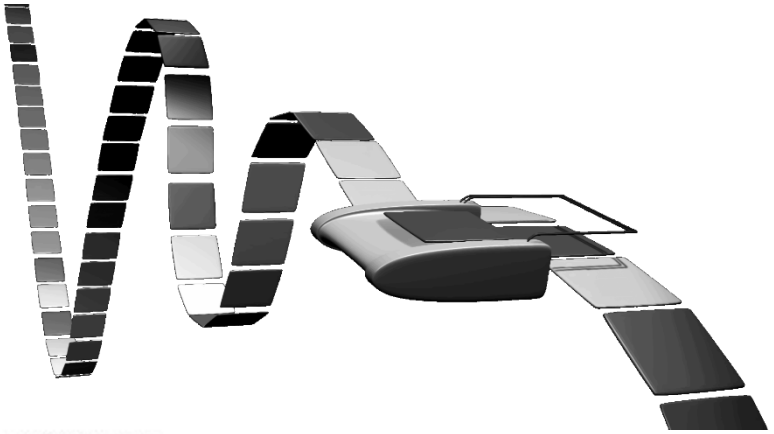
Algoritmalar

- Algoritma (alim Al-Khawarizmi'nin adından gelir) matematik, bilgisayar bilimi, dilbilim ve diğer alanlardan gelen, ***sonlu bir talimat dizisine*** veya bir sorunu çözmek için kullanılan bir adım prosedürünün açık bir açıklamasına atıfta bulunan bir terimdir. hesaplamaya veya veri işlemeye.

Alan Mathison Turing



Turing makinesi



Algoritmalar

Turing makinesi, İngiliz matematikçi Alan Turing tarafından 1936'da tanımlanan hayali bir bilgisayardır. Turing'in çalışması, algoritma kavramının (mekanik, biçimsel veya etkili prosedür olarak da bilinir) ilk doğru tanımıdır. Bilgisayar bilimlerinde esas olarak algoritmaların hesaplanması ve karmaşıklığının yanı sıra matematiksel mantık alanlarındaki temel sonuçları kanıtlamak için kullanılır.

Turing makinesi dört bileşenden oluşur :

1. **Hafıza** – Hücrelerden oluşan potansiyel olarak sonsuz bant, her hücreye sonlu bir alfabenin sembolü yazılır. Alfabe özel bir boş sembol (genellikle “0” olarak yazılır) ve bir veya daha fazla başka sembol içerir. Makinenin çalışmasının her anı bant sınırlıdır, ancak gerekirse sola veya sağa boş sembolü içeren yeni hücreler ekleyebiliriz.
2. **Kafa-** Hesaplamanın her anında bandın belirli bir hücresinin üzerinde olduğunu düşünün. Her dokunuşta kafa, üzerinde bulunan hücrenin sembolünü okur, yeni bir sembol yazar ve talimata ve okunan sembole göre bant üzerinde sola veya sağa hareket eder.
3. **Program** - Çağdaş bilgisayarlardan farklı olarak bellekten ayrı, sınırlı bir talimat listesi. Her talimat, kafanın alfabeden i'inci harfi okuması durumunda ne yapılması gerektiğine dair bir dizi göstergedir. Her gösterge, kasete hangi sembolün yazılması gerektiği, bir sonraki adımda hangi talimatın uygulanacağı ve kafanın nereye (sola veya sağa) hareket ettirileceği bilgilerini içerir.
4. **Kayıt-** Aktif talimatın numarasını içeren kayıt (program sayacı). Bir talimatın başlangıç olduğu söylenir, yani hesaplama, program sayacına numarasının girilmesiyle başlar. Son bir talimat daha var; ona ulaşıldığında hesaplama durur.

Co-funded by the Erasmus+ Programme of the European Union

Algoritmalar

- Gayri resmi bir algoritma, bir veya daha fazla değeri girdi olarak alan ve başka bir değeri çıktı olarak dışarı aktaran, iyi tanımlanmış herhangi bir hesaplama prosedürüdür.
- Başka bir deyişle algoritmalar, iyi tanımlanmış bir görevi gerçekleştirmek için kullanılan tarifler gibidir.
- Bu satırda, örneğin Fibonacci sayı dizisini hesaplayan bazı blok kodlar, belirli bir algoritmanın yürütülmesidir.
- İki sayıyı toplamaya yönelik basit bir işlev bile, çok basit olsa bile bir algoritma olarak düşünülebilir.

Algoritmalar

- Algoritmalar öncelikle çalışılmalıdır çünkü belirli bir teknolojiye bağlı değildirler.
- Algoritmaları anlamak için Java, C++ veya herhangi bir yeni teknoloji bilmenize gerek yok.
- Belirli bir sorunu (benzer özelliklere sahip belirli programlama dilleri ile) çözmek için izlenmesi gereken adımlar kavramına yönelik bir anlayışa sahip olmayı gerektirir.
- Algoritmalar bilgisayar bilimleri alanı için önemlidir çünkü bir problemin çözümünde en iyi yola ulaşmak için farklı hesaplama yollarının analizinin yapılmasını sağlarlar.

Algoritmalar

- **"En iyi yol"** ile en az **kaynakla** çözüme en hızlı şekilde ulaşabilen yol kastedilmektedir.
- Algoritma, iyi tanımlanmış bir hesaplama problemini çözmek için güçlü bir araçtır.
- Örneğin bir sayı dizisini artan düzende sıralamamız gerekebilir.
- Bu sorun pratikte sıklıkla ortaya çıkmakta ve birçok standart tekniğin ve analiz aracının uygulamaya konması için iyi bir zemin oluşturmaktadır.

Algoritmalar

- Sıralama problemini şu şekilde tanımlayabiliriz :
- ***Giriş :*** n sayıdan oluşan bir dizi : $\langle a_1, a_2, \ldots, a_n \rangle$
- ***Çıktı :*** değişiklik (yeniden sıralama) $\langle a'_1, a'_2, \ldots, a'_n \rangle$, böylece $a'_1 \leq a'_2 \leq \cdots \leq a'_n$.

Örnek:

Giriş : 8 2 4 9 3 6

Çıktı : 2 3 4 6 8 9

Co-funded by the Erasmus+ Programme of the European Union

Algoritmalar

- Böyle bir girdi dizisine sıralama probleminin bir örneği denir.
- Genel olarak bir problemin örneği, problemin çözümünü hesaplamak için gerekli olan girdilerden (problem için uygulanan kısıtlamaların karşılanması) oluşur.
- Birçok program bunu bir ara adım olarak kullandığından, sıralama bilgisayar bilimlerinde temel bir işlemdir.
- Sonuç olarak kullanabileceğimiz çok sayıda iyi sıralama algoritmasına sahibiz.

Algoritmalar

- Belirli bir aktiviteyi gerçekleştirmek için en iyi algoritmanın hangisi olduğu daha çok aşağıdaki gibi diğer faktörlere bağlıdır:
- sıralanacak öğe sayısı
- elemanların değerleri üzerinde olası kısıtlamalar
- Bilgisayarın mimarisi
- depolama aygıtlarının türü: ana bellek, diskler vb.

Her doğru girdi için doğru çıktıyla duruyorsa bir algoritmanın doğru olduğu söylenir.

Doğru algoritmanın belirli bir hesaplama problemini çözdüğünü söylüyoruz.

Co-funded by the Erasmus+ Programme of the European Union

Algoritmalar

- Yanlış algoritma bazı girdi durumlarında hiç durmayabilir veya yanlış cevapla durabilir.
- Beklenenin aksine yanlış algoritmalar, eğer hata oranlarını kontrol edebilirsek bazen yararlı olabiliyor.
- Ancak genellikle doğru algoritmaları kullanmaya ve derlemeye çalışmalıyız.
- Bir algoritma, izlenecek hesaplama prosedürünün doğru bir açıklaması sağlandığı sürece bir bilgisayar programı olarak tanımlanabilir.

Co-funded by the Erasmus+ Programme of the European Union

Algoritmalar

- Algoritmalar hangi problemleri çözer?
- Algoritmaların pratik uygulamaları her yerdedir ve aşağıdaki örnekleri içerir:

- Biyoenformatikte gen dizilimini tanımlamaya yönelik algoritmalar

- İnternet, dünyanın her yerindeki insanların bu bilgilere hızlı bir şekilde erişmesini ve bu bilgileri geri almasını sağlar. İnternet nesneleri, akıllı algoritmalar kullanarak bu büyük miktarda veriyi yönetebilir ve işleyebilir.

Co-funded by the Erasmus+ Programme of the European Union

Algoritmalar

- Algoritmalar hangi problemleri çözer?
- Algoritmaların pratik uygulamaları her yerdedir ve aşağıdaki örnekleri içerir:

- E-ticaret, mal ve hizmetlerin elektronik ortamda akdedilmesine ve alışverişinin yapılmasına olanak sağlar ve kredi kartı numaraları, şifreler, banka ekstreleri gibi kişisel bilgilerin gizliliğine bağlıdır.

- E-ticarette kullanılan teknolojiler arasında sayısal algoritmalara ve sayı teorisine dayanan kriptografi ve dijital imzalar bulunmaktadır.

Co-funded by the Erasmus+ Programme of the European Union

Algoritmalar

- Algoritmalar hangi problemleri çözer?
- İmalat ve diğer ticari işletmelerin çoğu zaman kıt kaynakları en faydalı şekilde tahsis etmesi gerekir.
- Bir petrol şirketi beklenen karı en üst düzeye çıkarmak için kuyularını nereye yerleştireceğini bilmek isteyebilir.
- Bir siyasi aday, seçimleri kazanma şansını artırmak için bir reklam kampanyasına parayı nereye harcayacağını belirlemek isteyebilir.
- Bir İnternet servis sağlayıcısı, müşterilerine daha etkin hizmet verebilmek için ek kaynakların nereye yerleştirileceğini belirlemek isteyebilir.

Co-funded by the Erasmus+ Programme of the European Union

Algoritmalar

- Algoritmalar hangi problemleri çözer??
- **Bunların hepsi daha sonraki derslerde inceleyeceğimiz doğrusal programlama kullanılarak çözülebilecek problem örnekleridir.**
- Bir bilgisayar mühendisinin veya programcının bilmesi gereken en önemli algoritmalardan bazıları :
  - İkili arama – tek boyutlu bir dizide (veya sıralanan herhangi bir veri yapısında) belirli bir değeri, her adımda verilerin yarısını hariç tutarak bulmaya yönelik bir teknik.
  - Dijkstra'nın algoritması – Dijkstra'nın algoritması, bir grafiğin düğümleri arasındaki en kısa yolu bulmak için kullanılır ve yol ağını temsil edebilir.
  - Floyd-Warshall - Floyd-Warshall algoritması, ağırlıklı odaklı bir grafikteki tüm köşe çiftleri arasındaki en kısa yolları bulmak için kullanılır.
  - Kruskal algoritması – açgözlü algoritma türüdür. Bu, her adımda bir kümenin maksimum (veya minimum) öğesinin (yerel olarak) seçilmesiyle uygulanan herhangi bir algoritmadır ve amaç, genel maksimuma ulaşmaktır.

Bir teknoloji olarak algoritmalar

- Bilgisayarların sonsuz hızlı olduğunu ve bilgisayar belleğinin sonsuz büyüklükte olduğunu varsayalım. Algoritma öğrenmek için bir neden var mı?
- Algoritmamızın belirli bir süre sonra sona erdiğini ve doğru cevabı bulduğunu göstermekten başka bir neden olmasa bile cevap evettir.
- Bilgisayarlar sonsuz hızlı olsaydı, herhangi bir uygun problem çözme yöntemi işe yarardı.
- Muhtemelen uygulamanızın yazılım mühendisliği uygulamalarının en iyileri arasında olmasını isteyeceksiniz (örn. uygulamanız iyi tasarlanmış ve belgelenmiş olmalıdır).

Co-funded by the Erasmus+ Programme of the European Union

Bir teknoloji olarak algoritmalar

- Bu nedenle bilgisayar zamanı ve belleği sınırlıdır.

- Bu kaynakları akıllıca kullanmanız gerekir ve zaman veya hafıza açısından verimli algoritmalar kullanmak bunu yapmanıza yardımcı olacaktır.

- Aynı sorunu çözmek için geliştirilen farklı algoritmaların etkinlikleri genellikle tamamen farklılık gösterir.

Bir teknoloji olarak algoritmalar

- Genel olarak sistem performansı hem hızlı donanım seçimine hem de verimli algoritmaların kullanımına bağlıdır.
- Algoritmalar modern bilgisayarlarda kullanılan çoğu teknolojinin temelini oluşturur.
- Üstelik bilgisayarların giderek artan yetenekleriyle, onları daha büyük sorunları çözmek için kullanıyoruz.
- Sağlam bir bilgi tabanına ve algoritmik tekniğe sahip olmak, gerçekten yetenekli programcıları ve mühendisleri ayıran özelliklerden biridir.

Algoritma özellikleri

Giriş verileri

- Her algoritmanın açıklamasında miktarlar kullanılır. Bunlardan bazıları sabit, bazıları ise değişkendir.
- Kullanılmadan önce değişkenlere başlangıç değerleri adı verilen değerlerin verilmesi gerekir.
- Değişkenlerden bazıları başlangıç değerlerini algoritmanın kendi açıklaması içerisinde alır. Diğer değişkenler, algoritmanın yürütülmesi sırasında başlangıç değerlerini girdi verileri olarak alırlar.

Co-funded by the Erasmus+ Programme of the European Union

Algoritma özellikleri

Giriş verileri

- Bu şekilde değişkenler algoritmanın parametreleri olarak hareket eder.
- Örneğin Öklid algoritmasında M ve N değişkenlerinin değer olarak aldığı doğal sayılardır.
- Bazı algoritmaların girdi kümesinin boş küme olduğu durumlar vardır.

Co-funded by the Erasmus+ Programme of the European Union

Algoritma özellikleri

Çıkış verileri

- Her algoritmanın yürütülmesi belirli bir sonuç elde etmeyi amaçlamaktadır.
- Çoğu zaman bu sonuç, algoritmada kullanılan bazı niceliklerin uç değeridir.
- Bu son değerler başlangıç değerlerine bağlıdır, yani. e. giriş verileri üzerinde.
- Öklid algoritmasındaki çıktı sonucu, a (veya b) değişkeninin uç değeridir.

Co-funded by the Erasmus+ Programme of the European Union

Algoritma özellikleri

Algoritmalar aşağıdaki önemli özelliklere sahiptir:

1. Büyüklük – algoritma, tek tek belirli görevleri değil, homojen bir görev sınıfını çözmek için kullanılır. Örneğin, Öklid algoritması herhangi iki doğal sayının en büyük ortak bölenini bulmayı sağlar. Algoritmaların uygulanmasının etkinliği, büyüklüklerinden kaynaklanmaktadır. Bu sayede bir algoritma farklı durumlarda tekrar tekrar uygulanabilir. Her algoritma, algoritmanın uygunluğu olarak adlandıracağımız belirli bir veri sınıfına uygulanabilir.

Co-funded by the Erasmus+ Programme of the European Union

Algoritma özellikleri

Algoritmalar aşağıdaki önemli özelliklere sahiptir :

2. Kesinlik – algoritmanın açıklaması açıktır ve hiçbir belirsizlik içermez. Gerçekleştirilecek eylemleri açıkça tanımlar. Algoritma, belirli veriler (harici değişkenlerin değerleri) verildiğinde, uygulayıcının kim olduğuna bakılmaksızın aynı sonucu üretir. Kesinlik, algoritmaların çok çeşitli uygulayıcılar tarafından yürütülmesine olanak tanır. Algoritmalardaki talimatların her biri belirsizliği önleyecek şekilde yazılmalıdır.

Co-funded by the Erasmus+ Programme of the European Union

Algoritma özellikleri

Algoritmalar aşağıdaki önemli özelliklere sahiptir :

3. Etkinlik - başlangıç verilerinin kabul edilebilir değerleri göz önüne alındığında, sınırlı sayıda temel işlemin istenen sonuca yol açacağının garantisi. Algoritmanın sonlu sürede sonlandırılması. Algoritmaların bu özelliği onların pratik olarak uygulanmasına olanak sağlar. Bir algoritmanın sonlu yürütme süresi, her algoritmanın sonlu sayıda adımdan oluştuğu ve her adımın (her temel eylemin) sonlu bir zamanda yürütüldüğü varsayımının bir sonucudur.

Algoritma özellikleri

Algoritmalar aşağıdaki önemli özelliklere sahiptir :

4. Sağduyululuk - algoritma sınırlı sayıda temel reçeteden oluşur. Çoğu zaman, "temel reçete" yerine aynı anlama gelen başka terimler kullanılır - talimat, komut, operasyon veya operatör.

Algoritma, belirli girdi verilerinden başlayarak ve izin verilen işlemleri uygulayarak, belirli bir şekilde, sonlu sayıda adımda istenen sonuca götüren, belirli nesneler üzerinde toplu bir görevi çözmeye yönelik bir prosedürdür.

Co-funded by the Erasmus+ Programme of the European Union

Çözüm

- Algoritmaların bilim ve mühendislikte olduğu kadar günlük yaşamdaki rolü de çok büyüktür.
- Algoritmalar, bilimsel sonuçların açıklanması için uygun ve uygun bir biçimdir; başta matematik alanında olmak üzere yeni keşfedilen çeşitli yöntemlerin raporlarının algoritma biçiminde basıldığı çok sayıda dergi vardır.
- Algoritmaların büyük önemi onlara olan ilgiyi de belirliyor. Bilim ve teknolojinin her alanından uzmanlar, hem pratikte anında uygulama hem de bilimin gelişimi için çeşitli görevleri çözecek algoritmalar arıyor.
- **Algoritmalar, veritabanı sorgu optimizasyonunda (bir sorguyu işlemek için uygun bir yürütme stratejisi seçme süreci) yaygın olarak kullanılır.**

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Algoritmalar nedir ve algoritmaları incelemek neden faydalıdır?

2. Algoritmaların hesaplamadaki rolü nedir?

3. Algoritmaların temel özellikleri nelerdir?

Co-funded by the Erasmus+ Programme of the European Union

Dinlediğiniz için teşekkürler!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

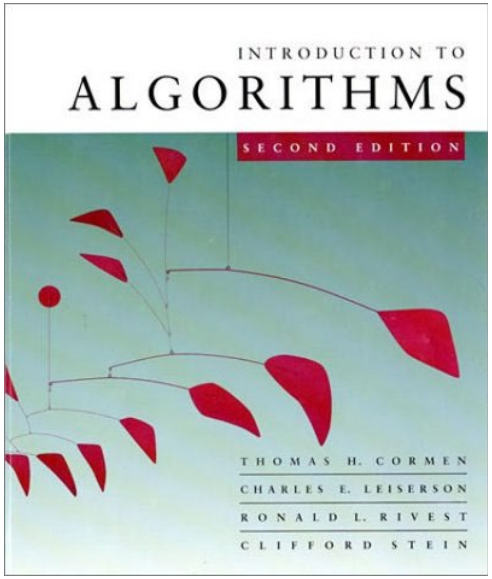
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

- **Modül 3. Sorgu optimizasyonu için algoritmalar ve veritabanlarındaki uygulamaları**
  - **Konu 1. Algoritmalara giriş**
    - Ders 2. Algoritmaların tasarımı ve analizi

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

ALGORİTMALARIN TASARIMI VE ANALİZİ



Carmen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2009). Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill.

Algoritmaların analizi

- Bir algoritmayı analiz etmek, algoritmanın ihtiyaç duyduğu kaynakları tahmin etmekten gelir.
- Bazen bellek veya bilgisayar donanımı gibi kaynaklar önemlidir ancak çoğu zaman **hesaplama süresini** belirlemek isteriz.
- Genel olarak, bir sorunu çözmek için birden fazla aday algoritmayı analiz ederken en verimli olanı belirlememiz gerekir.
- Böyle bir analiz, problemin çözümüne uygun birden fazla aday gösterebilir ancak analiz sürecinde bazı algoritmaları çöpe atabiliriz.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Algoritmaların analizi

Eklemeye göre sıralama analizi

6 5 3 1 8 7 2 4

Eklemeli sıralama prosedürü için gereken süre öncelikle girdinin boyutuna bağlıdır: bin sayıyı sıralamak, üç sayıyı sıralamaktan daha uzun sürer.

Ayrıca, eklemeye göre sıralama, aynı boyuttaki iki giriş dizisini, halihazırda nasıl sıralanmış olduklarına bağlı olarak sıralamak için farklı zamanlara sahip olabilir.

- Genel olarak, algoritmanın çalışma süresi girdinin boyutuyla birlikte artar, dolayısıyla geleneksel olarak analiz, algoritmanın çalışma süresinin girdinin boyutunun bir fonksiyonu olarak tanımlanmasından oluşur.

Algoritmaların analizi

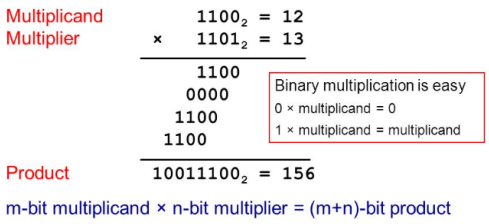
Eklemeye göre sıralama analizi

- Bunu yapabilmek için "**runtime**" ve "**input size**" terimlerini daha dikkatli tanımlamamız gerekiyor.
- Girdinin büyüklüğü fikri incelenen probleme bağlıdır.
- Sıralama gibi birçok problem için en doğal ölçü girdi başına öğe sayısıdır; belirli bir sıralama dizisinin boyutu n.

Algoritmaların analizi

Eklemeye göre sıralama analizi

- Çarpma gibi diğer birçok problem için
- iki tam sayı, giriş boyutunun en iyi ölçüsü
- temsil etmek için gereken toplam bit sayısıdır
- girdi ikili formattadır.



- Bazen girdi boyutunu bir yerine iki sayıyla tanımlamak daha uygundur.
- Örneğin, algoritma girişi bir grafikle ilişkiliyse, girişin boyutu grafiğin köşe ve kenar sayısıyla açıklanabilir.
- Çalıştığımız her problem için hangi girdi ölçü boyutunun kullanılacağını belirtmeliyiz.

Algoritmaların analizi

Eklemeye göre sıralama analizi

- Belirli bir görev için bir algoritmanın çalışma süresi, yürütülen ilkel işlemlerin veya "**adımların**" sayısıdır.
- Adım kavramını makineden bağımsız olacak şekilde tanımlamak uygundur.
- Kaynak kodun her adımının yürütülmesinin sabit bir zaman aldığını varsayarsak, her adım için farklı bir süreye ihtiyacımız olacaktır.
- Bu bakış açısı RAM modeliyle tutarlıdır ve aynı zamanda çoğu bilgisayarda sözde kodun nasıl uygulanacağını da yansıtır.

Algoritmaların analizi

Eklemeye göre sıralama analizi

- Eklemeye göre sıralama analizimizde her ikisini de dikkate almak gerekir: dizi öğelerinin zaten sıralandığı **en iyi durum** ve giriş dizisinin ters sıralandığı **en kötü durum**.
- Algoritmanın en kötü durum zamanlaması bize her giriş için çalışma süresine ilişkin bir üst sınır verir.
- Bunu bilmek algoritmanın asla daha uzun sürmemesini sağlar.

Co-funded by the Erasmus+ Programme of the European Union

Algoritmaların analizi

Eklemeye göre sıralama analizi

- Bazı algoritmalar için en kötü durum oldukça sık meydana gelir.
- Örneğin, belirli bir bilgi parçası için bir **veritabanı aranırken**, arama algoritması, bilginin veritabanında mevcut olmadığı en kötü duruma düşecektir.
- Bazı uygulamalarda eksik bilgilerin aranması oldukça yaygındır.

Algoritmaların analizi

Sahte kod

- Sözde kod, bilgisayarın donanımıyla ilgisi olmayan ve programın kullanılabilmesi için bilgisayar kodunun yeniden yazılmasını gerektiren bir program kodudur.
- Katı sözdizimi kuralları yoktur, bilgisayarlar için değil insanlar için tasarlanmıştır.
  - **Algoritmanın üst düzey açıklaması**
  - **İnsan diline göre daha yapılandırılmış**
  - **Programdan daha az ayrıntıyla**
  - **Algoritmaların tanımlanması için tercih edilen gösterim**
  - **Bir program yazarken sorunları gizler.**

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Algoritmaların analizi

Örnek: Bir dizinin en büyük elemanını bulma.

Algorithm arrayMax(A, n)

Input: array A of n integers

Output: maximum element of A

currentMax <- A[0]

for i <- 1 **to** n−1 **do**

if A[i] > currentMax **then** currentMax <- A[i]

return currentMax

Co-funded by the Erasmus+ Programme of the European Union

Algoritmaların analizi

Sahte kod ayrıntıları

- Operatörleri yönetme

if…then…[else…]
while…do…
repeat…until…
for…do…
Boşluk parantezlerin yerini alır.

- Yöntem tanımlama

Algorithm method(arg[, arg…])
Input…
Output…

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Algoritmaların analizi

Sahte kod ayrıntıları

- Bir işlevi/yöntemi çağırma
  var.method (arg[, arg…])
- Değer dönüşü

  return expression
- İfade

  <- Assignment (like = in C++)
  = Equality testing (like == in C++)
  n^2 - degrees, etc. mathematical notations are allowed.

Co-funded by the Erasmus+ Programme of the European Union

Algoritmaların analizi

Temel işlemler (ilkeller)

- İfade hesaplaması.
- Bir değişkene değer atamak.
- Bir dizide indeksleme.
- Bir işlevin çağrılması.
- Bir işlevden bir değer döndürün.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Algoritmaların analizi

Temel işlemleri sayma

Sözde kodu kontrol ederek, giriş boyutunun bir fonksiyonu olarak algoritma tarafından gerçekleştirilen maksimum ilkel işlem sayısını belirleyebiliriz..

Example

Algorithm arrayMax(A, n)	**Number of operations**
currentMax <- A[0]	2
for i <- 1 **to** n-1 **do**	$2+n$
if A[i] > currentMax **then**	$2(n-1)$
currentMax <- A[i]	$2(n-1)$
{increment counter i}	$2(n-1)$
return currentMax	1
	$7n-1$

Algoritmaların analizi

Çalışma süresinin tahmini

- ArrayMax algoritması en kötü durumda 7n-1 temel işlemi gerçekleştirir.
- Tanımlamak:

a = en hızlı temel işlemin çalışma süresi.

b = en yavaş temel işlemin çalışma süresi.

ArrayMax için en kötü durum *T(n)* olsun. O halde $a(7n - 1) \leq T(n) \leq b(7n - 1)$ olur.

Bu nedenle çalışma süresi *T(n)* iki doğrusal fonksiyonla sınırlıdır.

Co-funded by the Erasmus+ Programme of the European Union

Algoritma tasarımı

- Pek çok kullanışlı algoritma yapı olarak özyinelemelidir: belirli bir sorunu çözmek için, yakından ilişkili alt problemlerle ilgilenmek üzere kendilerini bir veya daha fazla kez yinelemeli olarak çağırırlar.
- Bu algoritmalar tipik olarak "böl ve yönet" yaklaşımını izler: problemi orijinal probleme benzer ancak boyutları daha küçük olan birkaç alt probleme bölerler ve alt problemleri yinelemeli olarak çözerler ve daha sonra başlangıçtaki probleme bir çözüm oluşturmak için çözümleri birleştirirler.

Algoritma tasarımı

- Böl ve yönet paradigması, yinelemenin her düzeyinde üç adım içerir :

- Problemi, ana problemin daha küçük kopyaları olan bir dizi alt probleme **bölmek;**
- Alt problemleri **ele almak** ve yinelemeli olarak çözmek. Alt problem boyutları yeterince küçükse alt problemleri basit bir şekilde çözün.
- Alt problemlerin çözümlerini başlangıç probleminin çözümünde **birleştirmek**.

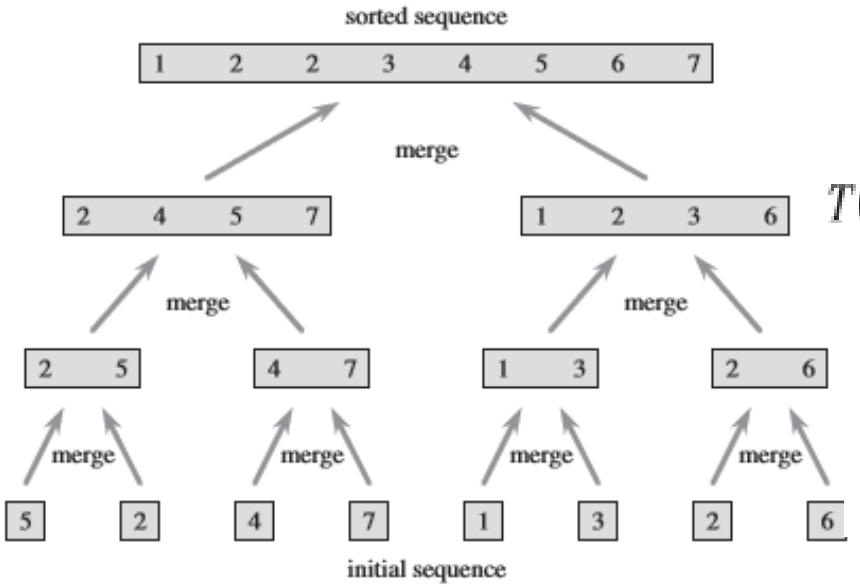
Birleştirerek sıralama (birleştirerek sıralama) algoritması böl ve yönet paradigmasını kullanır.

Co-funded by the Erasmus+ Programme of the European Union

Algoritma tasarımı

- Bir algoritmanın kendisi özyinelemeli bir çağrı içerdiğinde, çalışma süresini genellikle n boyutunda bir problemin çalışma süresini daha küçük girdilerdeki çalışma süresi cinsinden tanımlayan bir yineleme veya yineleme denklemiyle tanımlayabiliriz.
- Daha sonra yinelemeyi çözmek ve algoritmanın performansına ilişkin sınırlar sağlamak için matematiksel araçları kullanabiliriz.

Algoritma tasarımı



$$T(n) = \begin{cases} \Theta(1) & \text{if } n \le c\,, \\ aT(n/b) + D(n) + C(n) & \text{otherwise}\,. \end{cases}$$

böl ve fethet

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1\,, \\ 2T(n/2) + \Theta(n) & \text{if } n > 1\,. \end{cases}$$

birleştirme sıralaması

Co-funded by the Erasmus+ Programme of the European Union

Özet

- **Analiz kısmı girdi büyüklüğü, zaman karmaşıklığı ve asimptotik büyüme hızı sınırları kavramlarını içerir.**
- **En kötü durum karmaşıklığının dışında, ortalama karmaşıklığın analizi de etkilenir.**
- **Algoritmaların karmaşıklığını tahmin etme teknikleri ve bu tekniklerin klasik örnekler üzerinde kullanımı tartışılmaktadır.**
- **Zaman karmaşıklığı ve bellek karmaşıklığı gibi farklı algoritmalar analiz edilir.**

Co-funded by the Erasmus+ Programme of the European Union

Özet

- "Tasarım" kısmı, verimli algoritmaların oluşturulmasını kolaylaştırmak için tasarlanmış bir dizi algoritmik şemayı sunar: "böl ve yönet", "dinamik programlama", "açgözlü", "grafik geçişi", "geri izleme" ve diğer birçok algoritmik teknik ve yaklaşım .
- Tipik görevleri çözmek için şemalar uygulanır.
- Yaklaşık algoritmaların derlenmesine yönelik şemalar gösterilmiştir.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Algoritma analizi nedir?
2. Algoritma tasarımı nedir?
3. Sahte kod nedir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

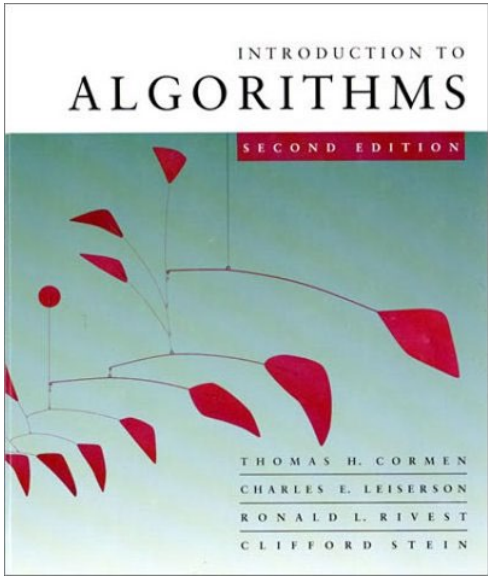
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

- **Modül 3. Sorgu optimizasyonu için algoritmalar ve veritabanlarındaki uygulamaları**
  - **Konu 1. Algoritmalara giriş**
    - Ders 3. Algoritmaların karmaşıklığı. Karmaşıklık türleri ve tahminleri

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

ALGORİTMALARIN KARMAŞIKLIĞI. KARMAŞIKLIK TÜRLERİ VE DEĞERLENDİRİLMESİ.



Carmen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2009). Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill.

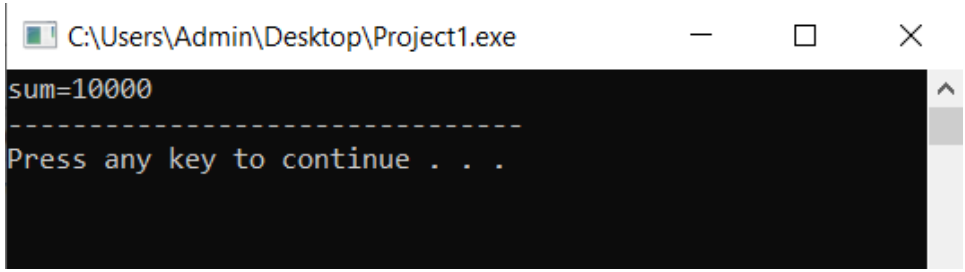
Giriş

- Algoritma değerlendirmesi ve karmaşıklığı konusu önemlidir ve gözden kaçırılmamalıdır.
- Bir bilgisayar algoritmasını değerlendirirken genellikle onun üç özelliğiyle ilgileniriz:
  - **Sadelik (ve zarafet);**
  - **Doğruluk;**
  - **Haslık**.
- Bunlardan ilkini herkes sezgisel olarak (ve biraz da öznel olarak) "ölçebilir"ken, son ikisi çok daha **derin bir analiz** gerektirir..

Algoritmaları analiz etme teknikleri

- *Bir algoritmanın aşağıdaki program parçasını düşünün*:

```
1) n = 100;
2) sum = 0;
3) for (i = 0; i < n; i++)
4) for (j = 0; j < n; j++)
5) sum++;
```



Yukarıdaki algoritmanın ne kadar hızlı çalışacağıyla ilgileniyoruz. Deneysel olarak yapabileceğimiz şey, işi bitirmenin ne kadar süreceğini kontrol etmektir.

Algoritmaları analiz etme teknikleri

- Davranışını daha genel olarak incelemek için onu n'nin diğer değerleri için çalıştırabiliriz.
- Giriş veri boyutu ile yürütme hızı arasındaki bağımlılık

Input size	Running time
10	0,000001 sec.
100	0,0001 sec.
1000	0,01 sec.
10000	1,071 sec.
100000	106,543 sec.
1000000	10663,6 sec.

- n'yi 10 kat artırdığımızda programın yürütme süresi 100 kat artar.

Algoritmaları analiz etme teknikleri

- Bu parçayı daha derinlemesine inceleyelim

```
1)    n = 100;
2)    sum = 0;
3)    for (i = 0; i < n; i++)
4)      for (j = 0; j < n; j++)
5)        sum++;
```

  - 1) ve 2) numaralı satırlarda sabit zaman alan statik bir başlatma vardır. a ile gösterelim.
  - i = 0 ve i++ işlemleri ve ayrıca i < n kontrolü için yine sabit bir süre gereklidir (her biri işlemcinin sabit sayıda talimatını temsil eder), bunu b, c, d ile göstereceğiz. .
  - 4. satırda j = 0, j < n ve j++ işlemleri için gerekli süreleri e, f, g ile belirtiyoruz.
  - Son olarak, 5. satırdaki işlem de sabit zaman gerektirir: h olsun.

Algoritmaları analiz etme teknikleri

- Bu şekilde tanıtılan gösterimlerle, algoritmanın toplam çalışma süresini keyfi bir n değeri için hesaplamak zor değildir:
-

$$a + b + n.c + n.d + n.(e + n.f + n.g + n.h) =$$
$$= a + b + n.c + n.d + n.e + n.n.f + n.n.g + n.n.h =$$
$$= n^2.(f + g + h) + n.(c + d + e) + a + b$$

Bunu hatırlat *a, b, c, d, e, f, g, h* sabitlerdir. belirtelim :

$i = f + g + h$

$j = c + d + e$

$k = a + b$

(here i and j have nothing in common with the variables used in the fragment above). Thus the algorithm is running for time:

$$i.n^2 + j.n + k$$

Algoritmaları analiz etme teknikleri

- i, j ve k sabitleri algoritmanın haslığı açısından önemlidir ancak belirleyici değildir. Pratikte bir algoritmanın etkililiğini incelediğimizde onlarla ilgilenmeyiz. Bu sabitler öncelikli olarak programımızın makine temsiline ve aynı zamanda üzerinde gerçekleştirildiği makinenin hızına bağlıdır.
- Üstelik algoritmamızın davranışını incelerken tek değerli j'yi bile göz ardı edebiliriz. n ve k'yi seçin ve yalnızca n'nin en çok katıldığını tutun.
- Bu "basitleştirmenin" amacı, belirli bir algoritma için yalnızca en önemli özelliği bırakmaktır; e. , çalışma zamanının en çok bağlı olduğu özellik, i. e. , giriş veri boyutu arttıkça en hızlı büyüyen.

Algoritmaları analiz etme teknikleri

- Giriş verilerinin boyutu n'ye bağlı olarak verilen iki algoritma A1 ve A2'nin çalışma süresini gösteren iki fonksiyonu ele alalım: f = 2. n2 ve g = 200. n.
- g'nin katsayısı f'ninkinden çok daha büyük olmasına rağmen, n sabit bir değeri geçtiğinde (bu durumda n > 100), A2 algoritmasının görevi A1'den daha hızlı çözeceğini görmek kolaydır.
- Ayrıca n arttıkça iki algoritmanın çalışma süreleri arasındaki oran A2 lehine artmaktadır.
- Asimptotik olarak A2 algoritması daha hızlıdır ve karmaşıklığı doğrusaldır, A1 algoritması ise ikinci derecedendir.

Algoritmaları analiz etme teknikleri

- **Karmaşıklık nedir?**
- Kısaca karmaşıklık, bir algoritmayı yürütmek için gereken sürenin veya belleğin, giriş veri boyutunun değiştirilmesiyle nasıl değiştiğidir.
- Algoritmaların karmaşıklığını resmi olarak değerlendirirken, onların n noktasında sonsuza doğru olan davranışlarıyla ilgileneceğiz. Bir algoritmanın karmaşıklığını f : N → N tipindeki fonksiyonlarla tanımlayacağız.

Definition 1.27. $O(F(n)) = \{f(n) \mid \exists c\ (c > 0), \exists n_0(c): \forall n > n_0 : 0 \leq f(n) \leq c.F(n)\ \}$

I.e. *O(F(n))* is a set of *f* functions for which such constant *c (c > 0)* that $f(n) \leq c.F(n)$ exists for all *big enough* values of *n*, i.e. constant n_0 exists (eventually depending on *c*), for which the above inequation is satisfied for each $n > n_0$. In this way *O(F)* defines the set of all functions that *increse* **not** *faster* than *F*.

Algoritmaları analiz etme teknikleri

- **Asimptotik gösterim**
- 1. **O(F)** F, i'den daha hızlı artmayan tüm f fonksiyonlarının kümesini tanımlar. e. n'nin yeterince büyük tüm değerleri için f (n) <= cF (n) şeklinde bir c > 0 sabiti vardır.

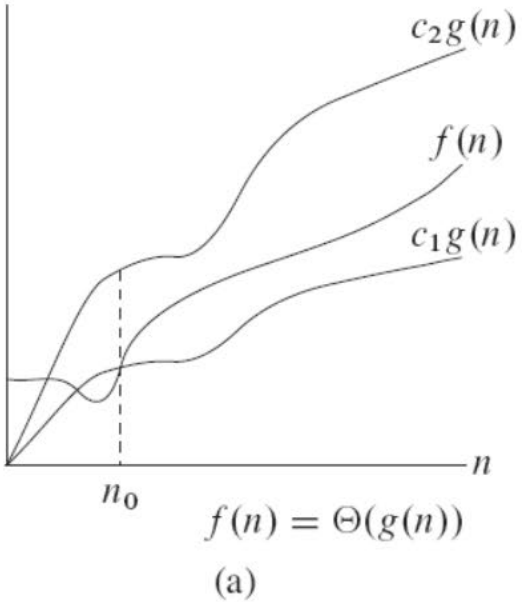
- 2. ***Theta (F)*** F kadar hızlı artan tüm f fonksiyonlarının kümesini tanımlar (sabit bir çarpana izin vererek), i. e. n'nin yeterince büyük tüm değerleri için c1F(n) <= f (n) <= c2F(n) olacak şekilde c1 > 0 ve c2 > 0 sabitleri mevcuttur.

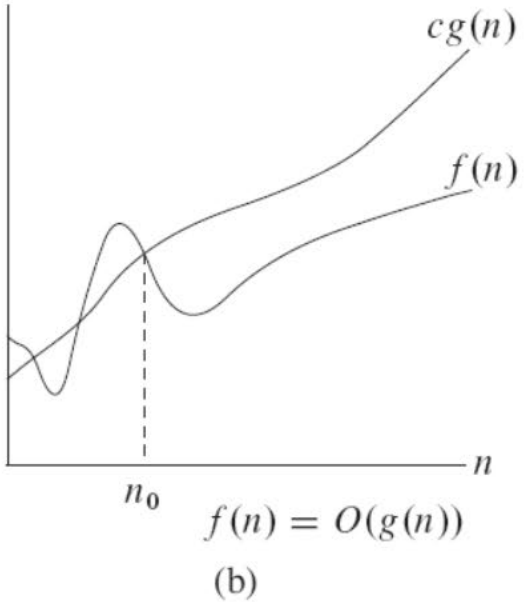
- 3. **Omega (F)** F, i'den daha yavaş artmayan tüm f fonksiyonlarının kümesini tanımlar. e. n'nin yeterince büyük tüm değerleri için f (n) >= cF (n) şeklinde bir c > 0 sabiti vardır.

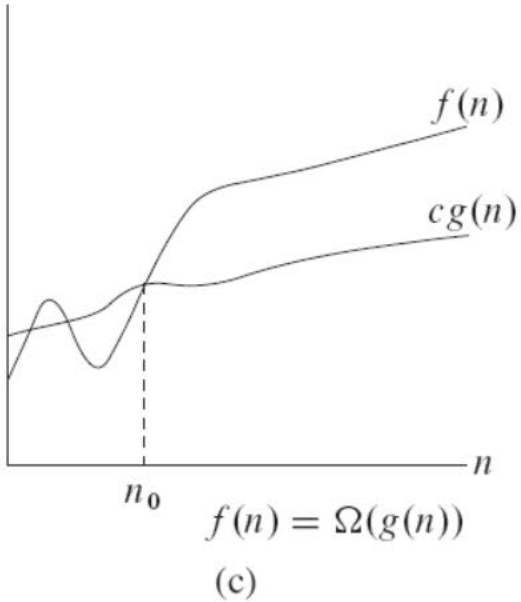
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Algoritmaları analiz etme teknikleri







Bir algoritmanın karmaşıklığı, verilen fonksiyonun büyüme hızıyla birlikte artar

En kötü durum

En iyi durum

Algoritmaları analiz etme teknikleri

- Temel bir işlemin karmaşıklığı sabittir, yani. e. Ç(1). Temel bir işlemin ne olduğunu tanımlamak kolay değildir. Temel olarak temel bir işlem, işlenen verinin boyutundan bağımsız olarak sabit bir zamanda gerçekleştirilen işlemdir. Genel durumdaki temel işlemler, örneğin atama, toplama, çarpma vb.'dir.
- Ancak yüz basamaklı sayılarla çalışırken çarpma işlemini temel bir işlem olarak düşünmek iyi bir fikir değildir. Trigonometrik fonksiyonları (sinüs, kosinüs vb.), üsleri, logaritmayı temel işlemler olarak düşünmek iyi değildir.

Algoritmaları analiz etme teknikleri

Operatörlerin sırası

Bir operatör dizisinin zaman karmaşıklığı, bunlardan daha yavaş olanın karmaşıklığı tarafından belirlenir.

Resmi olarak, F1 karmaşıklığına sahip s1 operatörünün ardından F2 karmaşıklığına sahip s2 operatörü geliyorsa şunu yazabiliriz:

$$T(s_1) \in O(F_1),\ T(s_2) \in O(F_2) \Rightarrow T(s_1; s_2) \in O(\max(F_1, F_2))$$

This is equivalent to the rule:

$$f_1 + f_2 \in (\max(f_1, f_2))$$

Algoritmaları analiz etme teknikleri

Operatörlerin bileşimi

- Bir operatörü başka bir operatörün kapsamına yerleştirirken karmaşıklık, karmaşıklıklarının çarpımı olarak hesaplanır,

$$T(s_1) \in O(F_1),\ T(s_2) \in O(F_2) \Rightarrow T(s_1\{s_2\}) \in O(F_1.F_2))$$

This is equivalent to the rule:

$$f_1.f_2 \in O(f_1.f_2)$$

Algoritmaları analiz etme teknikleri

if-yapıları

```
if (p)
   s1;
else
   s2;
```

- Eğer p, s1 ve s2'nin karmaşıklıkları O(P), O(F1), O(F2) ise, gösterilen parçanın karmaşıklığı max(O(P), O(F1), O(F2)) olur. , P, F1 ve F2 arasında en hızlı büyüyen fonksiyonun karmaşıklığı.

Algoritmaları analiz etme teknikleri

Döngüler

Döngüye bakalım :

```
fact = 1;
 for (i = 1; i <= n; i++)
   fact *= i;
```

- Döngünün gövdesinin n'den bağımsız olarak sabit bir c zamanı aldığını düşünebiliriz.
- For döngüsü operatörünün karmaşıklığı O(n)'dir. Daha sonra tüm döngünün karmaşıklığına ilişkin kompozisyon kuralından O(c.n), i elde ederiz. e. Açık).
- Burada, döngüden önce ilk başlatmanın karmaşıklığını eklemeliyiz (karmaşıklığı O(1) olan), burada tutarlılık kuralına göre şunu elde ederiz: O(1+n). Sonunda karmaşıklığın O(n) olduğu ortaya çıkıyor.

Algoritmaları analiz etme teknikleri

İç içe geçmiş döngüler

```
sum = 0;
for (i = 0; i < n; i++)
  for (j = 0; j < n; j++)
    sum++;
```

- Birbirinden bağımsız sayaçlara sahip iki veya daha fazla iç içe geçmiş döngünün karmaşıklığı kolaylıkla elde edilebilir. Yukarıdaki parçadan iki iç içe döngü olması durumunda f ∈ n olur. O(g), burada g iç döngünün karmaşıklığıdır. Ama g ∈ O(n), o halde f ∈ O(n.n), f ∈ O(n2).

Algoritmaları analiz etme teknikleri

Örnek

```
sum = 0;
for (i = 0; i < n; i++)
  for (j = 0; j < n; j++)
    if (i == j)
      for (k = 0; k < n; k++)
        sum++;
```

- If operatörü n2 kez çalıştırılacaktır, ancak i == j kontrolünün sonucunun yalnızca n katı doğru olacaktır. En içteki döngünün karmaşıklığı doğrusal olduğundan, toplam O(n2) karmaşıklığını elde ederiz.

Algoritmaları analiz etme teknikleri

Logaritmik karmaşıklık

Program parçasına bakalım:

```
for(sum = 0, h = 1; h < n; h *= 2)
  sum++;
```

- Burada h 1,2,4 değerlerini alır 1,2,4,. . . , 2^k,. . . n'ye ulaşana kadar. Böylece sum++ $\log_2 n$ kez yürütülür ve algoritmanın karmaşıklığı $O(\log_2 n)$ olur.
- Bu tür algoritmik karmaşıklık, her adımda arama aralığının iki (neredeyse) eşit parçaya bölündüğü ikili aramaya da sahiptir.

Algoritmaları analiz etme teknikleri

Özyineleme

- Genel durumda özyinelemeyi analiz etmek önemsiz değildir. Tipik olarak, algoritmanın karmaşıklığı için T(n) = f(T(n–1)) biçiminde bir bağımlılık elde edilir.

- Faktöriyel

İşlevi düşünün :

```
unsigned fact(unsigned n)
{ if (n < 2)
    return 1;
  return n*fact(n-1);
}
```

Bu durumda özyineleme, for tipindeki tek bir döngüye eşdeğerdir; burada karmaşıklık nedeniyle kolayca O(n) elde ederiz.

Algoritmaları analiz etme teknikleri

Özyineleme

- ***Fibonacci sayıları***
- Ancak işler her zaman bu kadar basit değildir. Örneğin özyinelemeyle son derece verimsiz bir şekilde bulunabilen Fibonacci sayılarını ele alalım:

```
unsigned fib(unsigned n)
{
  if (n < 2)
    return 1;
  return fib(n-1) + fib(n-2);
}
```

n = 0 ve n = 1 için sabit zaman karmaşıklığımız var: bir temel kontrol ve sonuç döndürme. Diğer durumda yine aynı kontrolü yaparız, ancak bu kez bunu iki özyinelemeli dönüşüm izler.

Algoritmaları analiz etme teknikleri

Özyineleme

- ***Fibonacci sayıları***

Genel olarak aşağıdaki formüller geçerlidir:

$$T(0) = T(1) = O(1)$$
$$T(n) = T(n-1) + T(n-2) + O(1),\ n \geq 2$$

The above dependencies strongly remind of these from the Fibonacci numbers definition:

$$f_0 = f_1 = 1$$
$$f_n = f_{n-1} + f_{n-2}$$

It follows immediatlely that $T(n) \geq f_n$. However for the Fibonacci numbers the following independencies are valid: $\left(\frac{3}{2}\right)^{n-1} \leq f_n \leq 2^n,\ n \geq$ (It is proves slightly by induction). So it turns out that *T(n)* increases expotentially.

Algoritmaları analiz etme teknikleri

Özyineleme

- ***Fibonacci sayıları***

Ancak, önceden hesapladığımız bir şeyi yeniden hesaplamayacak kadar akıllıysak, algoritmanın karmaşıklığını O(n)'ye indirgeyebiliriz:

```
unsigned long f[MAX] = {0,0,0,...};
unsigned long fib(unsigned n)
{
  if (0 == f[n])
    if (n < 2)
      f[n] = 1;
    else
      f[n] = fib(n-1) + fib(n-2);
    return f[n];
}
```

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Algoritmaların tipik karmaşıklıkları

Complexity	Sign	Description
constant	O(1)	A constant number of steps (say 1, 5, 10, or some other number) is required to perform an operation, and this number is independent of the input data volume.
logarithmic	O(log(N))	To perform an operation on N elements requires a number of steps of the order of log(N), where the base of the logarithm is usually 2. For example, an algorithm with complexity O(log(N)) for N = 1,000,000 will take about 20 steps (to constant precision).
linear	O(N)	To perform an operation on N elements requires approximately as many steps as there are elements. Approximately 1,000 steps are needed for 1,000 items. The number of elements and the number of operations are linearly dependent, for example the number of steps is about N/2 or 3*N for N elements.
Log-Linear	O(n*log(n))	It takes approximately N*log(N) steps to perform an operation on N elements. Approximately 10,000 steps are needed for 1,000 elements.

Co-funded by the Erasmus+ Programme of the European Union

Algoritmaların tipik karmaşıklıkları

Complexity	Sign	Description
quadratic	$O(n^2)$	An operation requires N^2 number of steps, where N characterizes the input data volume. For example, an operation on 100 items requires 10,000 steps. If the number of steps is quadratic in the input data volume, then the complexity is quadratic.
cubic	$O(n^3)$	An operation requires N^3 steps, where N characterizes the amount of input data. For example for a 100 elements approximately 1 000 000 steps are executed.
exponential	$O(2^n)$, $O(N!)$, $O(n^k)$, …	To perform an operation or computation, a number of steps is required that is exponentially related to the size of the input data. For example, at N=10 the exponential function 2^N has a value of 1024, at N=20 it has a value of 1 048 576, and at N=100 the function has a value that is a number of about 30 digits.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

Algoritma analizi için özel teknikler

Barometre kullanma

Aşağıdaki program parçasını tekrar düşünün :

```
unsigned sum = 0;
for (i = 0; i < n; i++)
  for (j = 0; j < i*i; j++)
    sum++;
```

Yukarıda toplamların özelliklerini kullanarak karmaşıklığını tanımladık. Buna başka bir şekilde de yaklaşabiliriz: Uygun bir talimat (barometre) seçeriz ve bunun kaç kez yürütüldüğünü izleriz.

Bu bizi seçilen talimatla ilgisi olmayan diğer tüm talimatları analiz etme endişesinden kurtarır. Barometre nasıl seçilir?

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

Algoritma analizi için özel teknikler

Barometre kullanma

Aşağıdaki program parçasını tekrar düşünün :

```
unsigned sum = 0;
for (i = 0; i < n; i++)
  for (j = 0; j < i*i; j++)
    sum++;
```

Bu, en azından programdaki diğer talimatlar kadar sık yürütülen bir talimat olmalıdır.

Yukarıdaki program parçasında bunun için uygun bir aday sum++'dır. Bu arada, parça yürütüldükten sonra sum değişkeninin değeri bize sum++ komutunun yürütme sayısını verecektir.

Algoritma analizi için özel teknikler

- Bilgisayar algoritmalarını analiz ederken çoğunlukla **en kötü** veya **ortalama durumdaki** davranışlarını inceleriz ve **en iyi durumda** nasıl davrandıklarını neredeyse hiç umursamayız.
- Bir algoritmanın en kötü durum karmaşıklığını belirlemek amacıyla analiz edilmesinde yaygın olarak kullanılan bir teknik, her adımda mümkün olan en kötü sonucun meydana geldiğini varsaymaktır.
- Bu bize O(...)'nun doğru karmaşıklığını verir, ancak her zaman doğru bir tahmin vermez: sonuçlar genellikle oldukça kötümserdir ve pratikte algoritma en kötü durumda bile çok daha hızlı çalışır.
- Bunun nedeni, en kötü durumun her zaman her adımda ortaya çıkmayabilmesidir. Dolayısıyla genel karmaşıklığın tahmin edilenden önemli ölçüde daha iyi olduğu ortaya çıkabilir.

Algoritma analizi için özel teknikler

- **Algoritmaların ve programların karmaşıklık tahminini sağlam bir teorik temele oturtma fırsatını bize vermelerine rağmen, asimptotik tahminlere karşı bir miktar şüpheci olmalıyız.**
- Temel özellikleri, n'nin sonsuz büyümesi altında algoritmanın davranışıyla ilgilenmeleridir. Ancak gerçek bilgisayar dünyasında sonsuz büyüklükte sayılar yoktur.
- Bu, asimptotik fonksiyon tarafından verilen tahminin yetersiz olabileceği anlamına gelir: örneğin, sabitleri gizlediği için veya n'nin bu kadar büyük değerleriyle ilgilenmediğimiz için.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Algoritmaların karmaşıklığı nedir?
2. Zaman karmaşıklığı nedir?
3. Uzay karmaşıklığı nedir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

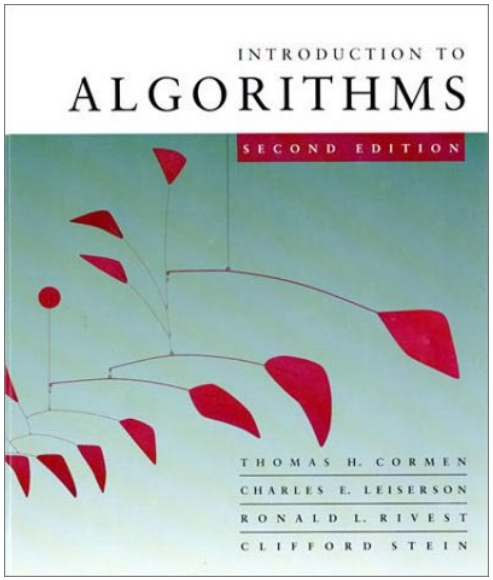
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

- **Modül 3. Sorgu optimizasyonu için algoritmalar ve veritabanlarındaki uygulamaları**
  - **Konu 2. Algoritma Tasarımında Stratejiler**
    - Ders 1: Algoritmalarda Böl ve Fethet Paradigmasını

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

ALGORİTMALARDA PARADİGMA'YI BÖL VE FETHEDİN



Cormen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2009). Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill.

Tarih

- Karmaşık bir görevi, ayrı ayrı ele alınması daha kolay olan ve çözümü ilk göreve kolay bir çözüm oluşturmaya olanak tanıyan daha basit birkaç göreve bölme fikrinin kökleri, antik çağlara kadar uzanır.
- İmparatorluğun komşu savaşan kabilelerine karşı dış politikasının temel ilkesi olarak "Böl ve yönet"i formüle eden ve teşvik eden Roma İmparatorluğu döneminde zirveye ulaştı.
- Böl-ve-fethet algoritmasının fikri, bir problemi, kolayca çözülebilecek kadar küçük hale gelene kadar sırayla (genellikle yineleme yoluyla) iki veya daha fazla alt probleme bölmektir.

Co-funded by the Erasmus+ Programme of the European Union

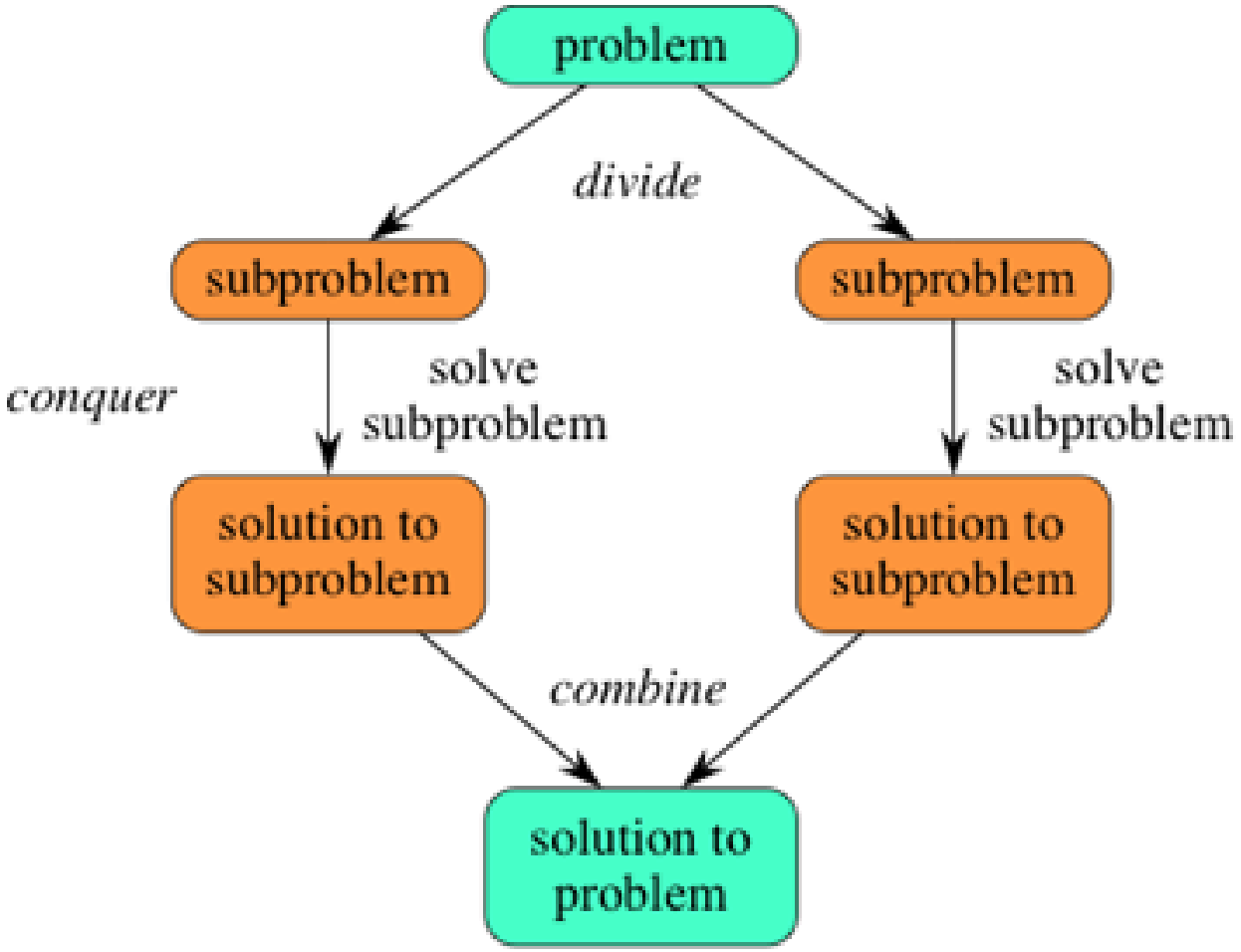
Tasarım paradigması böl ve yönet

1. Problemin (görevin) alt problemlere (alt görevlere) bölünmesi.

2. Problemin küçük alt problemlerini yinelemeli olarak çözün.

3. Bu çözümleri daha küçük örneklerde birleştirerek orijinalde (daha büyük örnekte) bir çözüm elde edin.

Böl ve Fethet algoritması



- **Böl ve Fethet algoritması**

Böl ve yönet örnekleri

Böl ve yönet algoritmalarının tipik örnekleri şunlardır:

- **Sıralama algoritmaları: birleştirme sıralaması ve hızlı sıralama;**
- **İkili arama ağacı;**
- **Bir sayıyı bir kuvvete yükseltmek;**
- **Fibonacci sayıları;**
- **Matris çarpımı: Strassen algoritması.**
- **Büyük Veri Analizi için böl-yönet yöntemi kullanılabilir.**

Co-funded by the Erasmus+ Programme of the European Union

Böl ve fethet

Sıralamayı birleştir:

1. Böl: önemsiz

2. Conquer: iki alt dizinin özyinelemeli sıralaması

3. Birleştir: sıralanan iki alt diziyi doğrusal zamanda birleştirir.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Böl ve fethet

Sıralamayı birleştir :

Adım adım örnek

6 5 3 1 8 7 2 4

Birleştirme sıralamasının bir satırı nasıl sıraladığını gösteren bir örnek.

Böl ve fethet

Sıralamayı birleştir– Temel teorem:

$$T(n) = a\,T(n/b) + f(n)$$

Case 1: $f(n) = O(n^{\log_b a - \varepsilon})$, constant $\varepsilon > 0$
$\Rightarrow T(n) = \Theta(n^{\log_b a})$.

Case 2: $f(n) = \Theta(n^{\log_b a} \lg^k n)$, constant $k \geq 0$
$\Rightarrow T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.

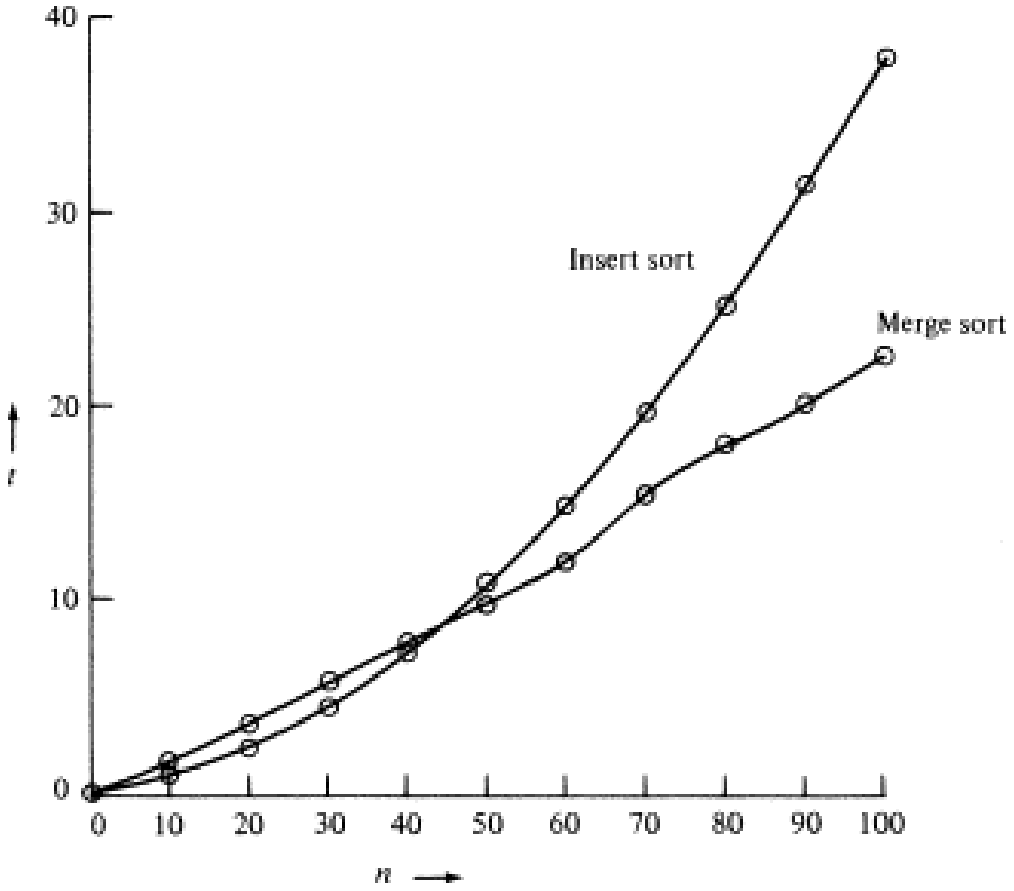
Case 3: $f(n) = \Omega(n^{\log_b a + \varepsilon})$, constant $\varepsilon > 0$,
and regularity condition
$\Rightarrow T(n) = \Theta(f(n))$.

Merge sort: $a = 2,\ b = 2 \Rightarrow n^{\log_b a} = n^{\log_2 2} = n$
$\Rightarrow$ Case 2 $(k = 0) \Rightarrow T(n) = \Theta(n \lg n)$.

Co-funded by the Erasmus+ Programme of the European Union

Böl ve fethet

Sıralamayı birleştir



Co-funded by the Erasmus+ Programme of the European Union

Böl ve fethet

Ikili arama :

Sıralanmış bir dizide bir öğe arayın:

- Böl: **ortadaki öğeyi kontrol etme**

- Fetih: **1 alt dizide yinelemeli arama**

- Birleştir: önemsiz

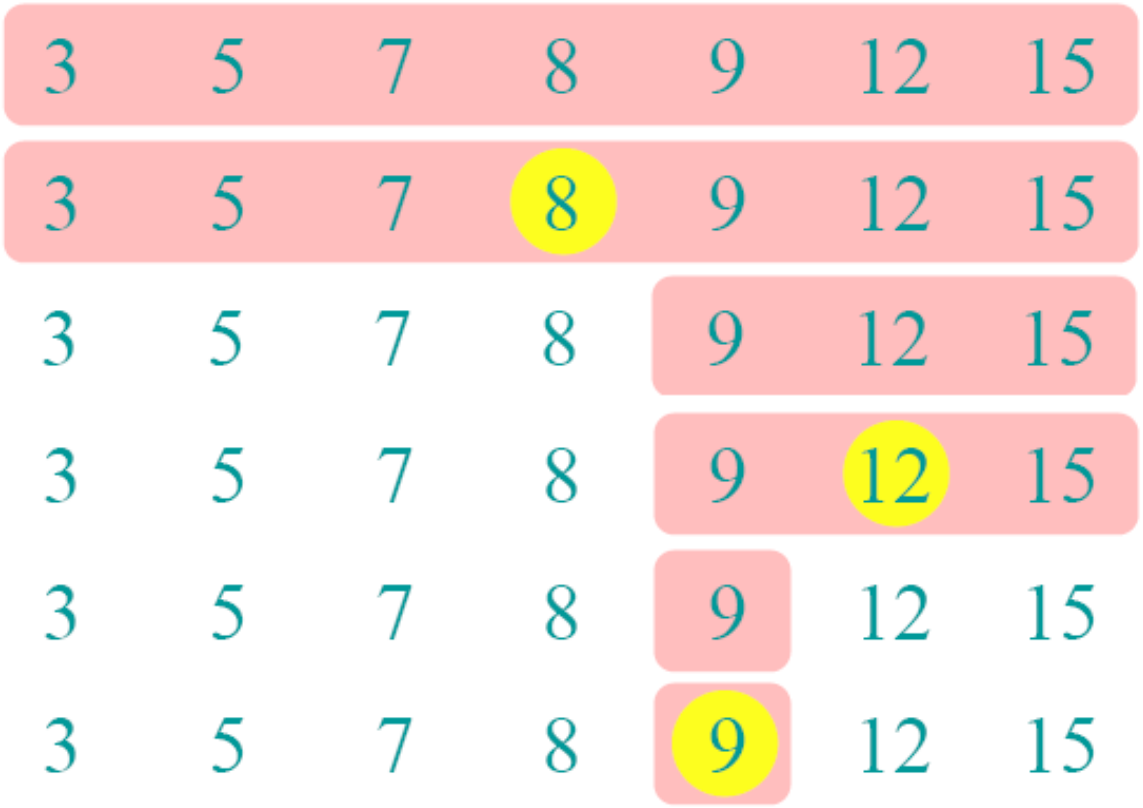
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Böl ve fethet

Ikili arama :

Örnek: 9 aranıyor:



Böl ve fethet

Ikili arama :

$$T(n) = 1\,T(n/2) + \Theta(1)$$

 # subproblems

subproblem size

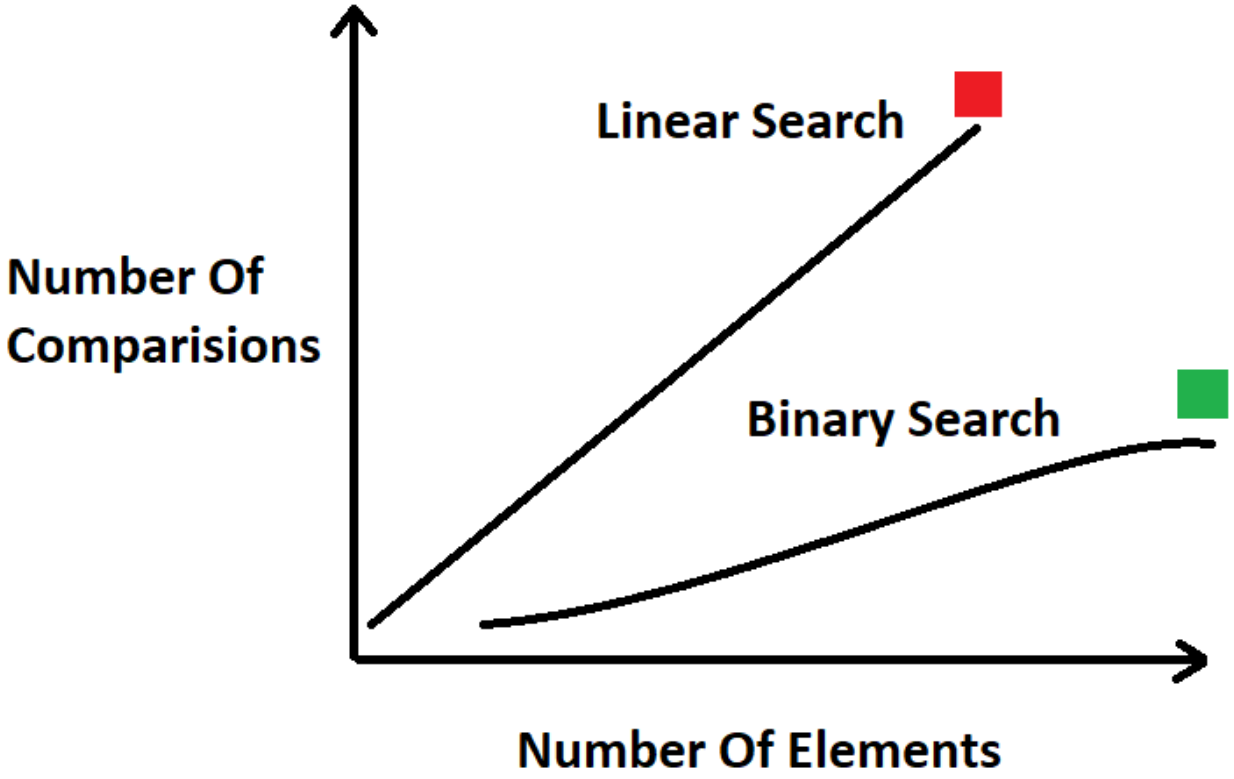
work dividing and combining

$$n^{\log_b a} = n^{\log_2 1} = n^0 = 1 \Rightarrow \text{CASE 2 } (k = 0)$$
$$\Rightarrow T(n) = \Theta(\lg n) .$$

Co-funded by the Erasmus+ Programme of the European Union

Böl ve fethet

Ikili arama :



Böl ve fethet

Bir sayıyı bir kuvvete yükseltmek :

Sorun: hesapla a^n, nerede $n \in \mathbb{N}$.

Saf algoritma: $\Theta(n)$.

`temp=power(base,expo/2)?`

Böl ve fethet algoritması:

Approach	Time Complexity	Space Complexity
Naive Iterative Approach	O(n)	O(1)
Divide and Conquer	O(n)	O(log n)
Optimized Divide and Conquer	O(log n)	O(log n)

$$a^n = \begin{cases} a^{n/2} \cdot a^{n/2} & \text{eğer } n \text{ çift sayıysa} \\ a^{(n-1)/2} \cdot a^{(n-1)/2} \cdot a & \text{eğer } n \text{ tek sayıysa} \end{cases}$$

$$T(n) = T(n/2) + \Theta(1) \Rightarrow T(n) = \Theta(\lg n)$$

Böl ve fethet

Fibonacci sayıları:

Özyinelemeli tanım :

$$F_n = \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F_{n-1} + F_{n-2} & \text{if } n \geq 2. \end{cases}$$

0 1 1 2 3 5 8 13 21 34 ⋯

Algorithm 1: F() Recursive

```
Data: N
Result: F(N)
if N == 0 or N == 1 then
    return N
end
return F(N-1) + F(N-2)
```

Saf özyinelemeli algoritma :

(üstel zaman), burada

altın oran.

$$\Omega(\phi^n)$$

$$\phi = (1+\sqrt{5})/2$$

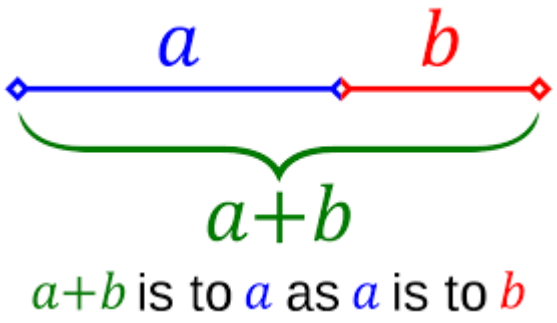
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Böl ve fethet

Altın Oran (Altın oran, Altın katsayı veya İlahi oran olarak da bilinir), daha uzun parçanın uzunluğunun, örneğin "a"nın, daha kısa parçanın uzunluğuna oranıdır, örneğin "b" eşittir toplamlarının "(a + b)" daha uzun uzunluğa oranı. Yunan harfi φ ile gösterilir ve yaklaşık olarak 1.618'e eşit bir değere sahiptir. .

Her Fibonacci sayısının bir öncekine oranı 1, 61803'e eğilimlidir. . . Sayı 1. 61803. . . bir "altın oran"dır. Büyük Yunanca harf φ (phi) ile gösterilir.



$$\frac{a+b}{a} = \frac{a}{b} = 1.618\ldots = \varphi$$

Altın oran araması, sıralanmış bir dizideki bir öğeyi aramak için başarıyla uygulanan bir böl ve yönet tekniğidir.

Böl ve fethet

Fibonacci sayılarının hesaplanması :

Aşağıdan yukarıya doğru :

Her değeri oluşturan bir satırda önceki iki değeri toplayarak $F_0, F_1, F_2, \ldots, F_n$ 'yi hesaplıyoruz.

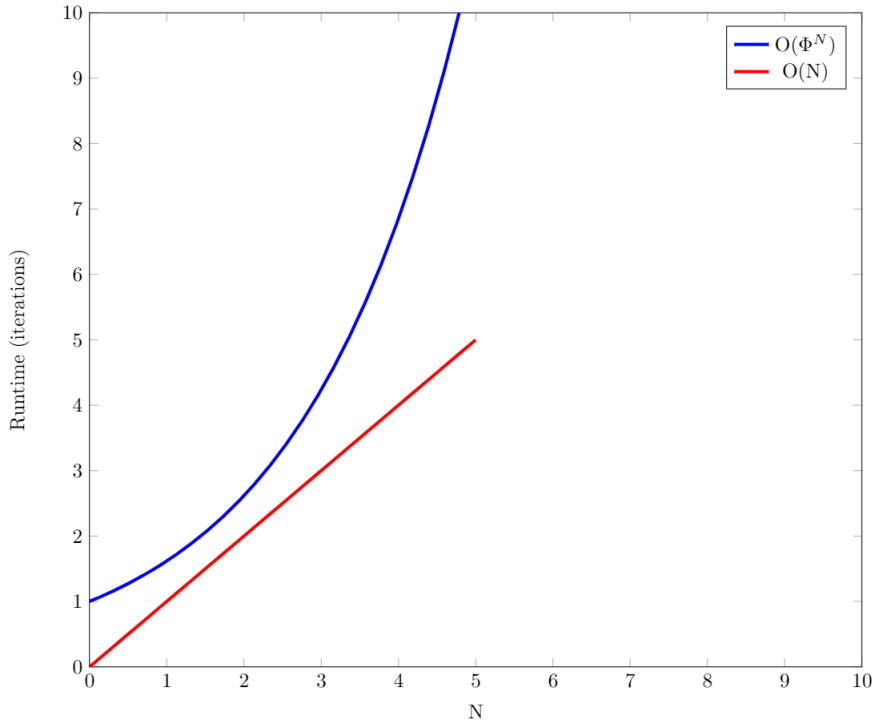
Süre : $\Theta(n)$.

Algorithm 4: F() Bottom-Up

```
Data: N
Result: F(N)
if N == 0 or N == 1 then
    return N
end
A ⟵ 0
B ⟵ 1
for I ← 2 to N do
    Temp ⟵ A + B
    A ⟵ B
    B ⟵ Temp
end
return B
```

Böl ve fethet

Fibonacci sayılarının hesaplanması :



Böl ve fethet

Matris çarpımı :

$$\left.\begin{array}{l}\textbf{Input:} \quad A=[a_{ij}],\ B=[b_{ij}].\\ \textbf{Output:} \quad C=[c_{ij}]=A\cdot B.\end{array}\right\} \quad i,j=1,2,\ldots,n.$$

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix}$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Böl ve fethet

Matris çarpımı:

Standart algoritma

Algoritmaların, özellikle de biyoinformatik alanındakilerin verimliliğinin artırılması, geniş çaplı bir etkiye sahip olabilir. Matris çarpımı biyoenformatikte sıklıkla karşılaşılan böyle bir sorundur.

$$\begin{aligned}
&\textbf{for } i \leftarrow 1 \textbf{ to } n \\
&\quad \textbf{do for } j \leftarrow 1 \textbf{ to } n \\
&\qquad \textbf{do } c_{ij} \leftarrow 0 \\
&\qquad\quad \textbf{for } k \leftarrow 1 \textbf{ to } n \\
&\qquad\qquad \textbf{do } c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}
\end{aligned}$$

$$\text{Running time} = \Theta(n^3)$$

Böl ve fethet

Matris çarpımı:

Böl ve yönet algoritması

Fikir : $n \times n$ matrix $= 2 \times 2$ matrix of $(n/2) \times (n/2)$ submatrices:

$$\begin{bmatrix} r & s \\ t & u \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

$$C = A \cdot B$$

$$\left.\begin{aligned} r &= ae + bg \\ s &= af + bh \\ t &= ce + dg \\ u &= cf + dh \end{aligned}\right\} \begin{array}{l} 8 \text{ mults of } (n/2) \times (n/2) \text{ submatrices} \\ 4 \text{ adds of } (n/2) \times (n/2) \text{ submatrices} \end{array}$$

Böl ve fethet

Matris çarpımı:

Böl ve yönet algoritması

Fikir: $n \times n$ matrix $= 2 \times 2$ matrix of $(n/2) \times (n/2)$ submatrices:

$$\begin{bmatrix} r & s \\ t & u \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

$$C = A \cdot B$$

$$\left.\begin{aligned} r &= ae + bg \\ s &= af + bh \\ t &= ce + dh \\ u &= cf + dg \end{aligned}\right\}$$

recursive

8 mults of $(n/2) \times (n/2)$ submatrices

4 adds of $(n/2) \times (n/2)$ submatrices

Böl ve fethet

Matris çarpımı:

Böl ve yönet algoritması

$$T(n) = 8\,T(n/2) + \Theta(n^2)$$

 # submatrices

submatrix size

work adding submatrices

$$n^{\log_b a} = n^{\log_2 8} = n^3 \Rightarrow \text{CASE 1} \Rightarrow T(n) = \Theta(n^3).$$

Basit bir algoritmadan daha iyi değil.

Böl ve fethet

Matris çarpımı:

Böl ve yönet algoritması, Strassen Algoritması

- Multiply 2×2 matrices with only 7 recursive mults.

$$P_1 = a \cdot (f - h)$$
$$P_2 = (a + b) \cdot h$$
$$P_3 = (c + d) \cdot e$$
$$P_4 = d \cdot (g - e)$$
$$P_5 = (a + d) \cdot (e + h)$$
$$P_6 = (b - d) \cdot (g + h)$$
$$P_7 = (a - c) \cdot (e + f)$$

$$r = P_5 + P_4 - P_2 + P_6$$
$$s = P_1 + P_2$$
$$t = P_3 + P_4$$
$$u = P_5 + P_1 - P_3 - P_7$$

7 mults, 18 adds/subs.
Note: No reliance on commutativity of mult!

Böl ve fethet

Matris çarpımı:

Böl ve yönet algoritması, Strassen Algoritması

- Multiply 2×2 matrices with only 7 recursive mults.

$$
\begin{aligned}
P_1 &= a \cdot (f - h) \\
P_2 &= (a + b) \cdot h \\
P_3 &= (c + d) \cdot e \\
P_4 &= d \cdot (g - e) \\
P_5 &= (a + d) \cdot (e + h) \\
P_6 &= (b - d) \cdot (g + h) \\
P_7 &= (a - c) \cdot (e + f)
\end{aligned}
$$

$$
\begin{aligned}
r &= P_5 + P_4 - P_2 + P_6 \\
&= (a + d)(e + h) \\
&\quad + d(g - e) - (a + b)h \\
&\quad + (b - d)(g + h) \\
&= ae + ah + de + dh \\
&\quad + dg - de - ah - bh \\
&\quad + bg + bh - dg - dh \\
&= ae + bg
\end{aligned}
$$

Böl ve fethet

Matris çarpımı:

Strassen algoritması

1. Bölme: A Ve B alt matrislere bölme $(n/2)\times(n/2)$
   Çarpmanın bir kısmı + ve - ile değiştirilebilir.
2. Fetih: alt matrislerin 7 çarpımının gerçekle $(n/2)\times(n/2)$
3. Birleştir: C için alt matrislerin + ve – değerlerini kullanırız

$(n/2)\times(n/2)$

$$T(n) = 7\,T(n/2) + \Theta(n^2)$$

Co-funded by the Erasmus+ Programme of the European Union

Böl ve fethet

Matris çarpımı:

Strassen algoritması

Strassen algoritması, hızlı matris çarpımı için doğrusal cebirde kullanılan bir algoritmadır. Büyük matrisler için klasik olandan daha hızlıdır. 1960'ların sonlarında Alman matematikçi Volker Strassen tarafından keşfedildi.

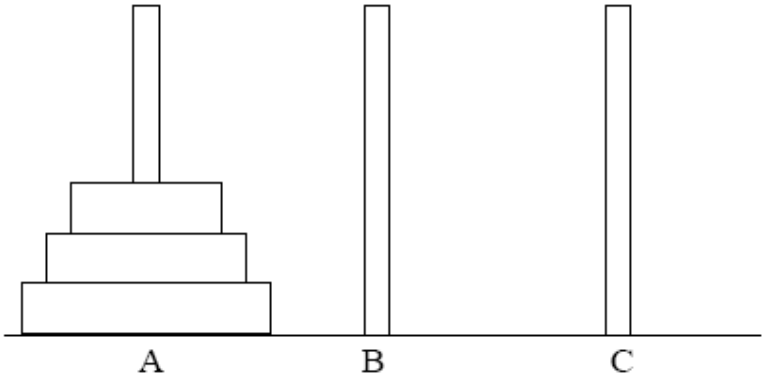
Boyutu N x N olan iki matrisin çarpımı için elde edilen matrisin hesaplanması Θ(n3) şeklindedir. Strassen bunu Θ(nlog7)'ye yükseltmeyi başardı.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

Böl ve fethet

Hanoi Kulesi Görevi :

Farklı çaplarda n sayıda disk ve A, B ve C olmak üzere üç sütun vardır. Diskler, boyutları azalarak birinci sütuna dizilir ve bir kule oluşturur.



Aşağıdaki kurallara tabi olarak ilk sütundan son sütuna aktarılmalıdırlar:

1. Her hamlede bir disk hareket ettirilebilir ve bu disk sütunlardan birinin en üstündeki disk olmalıdır.
2. Daha büyük çaplı bir disk, daha küçük çaplı bir diskin üzerine yerleştirilemez.

Co-funded by the Erasmus+ Programme of the European Union

Böl ve fethet

Hanoi Kulesi Görevi :

- Fikir şu; n sayıda diski A sütunundan C sütununa taşımak için, n-1 diski A sütunundan B sütununa taşımamız, ardından n numaralı diski (aralarında en büyüğü olan) A sütunundan diğer sütuna taşımamız gerekiyor. C sütunu ve son olarak kalan n-1 diski B sütunundan C sütununa taşıyın.
- Böylece, doğal olarak görevi, daha küçük boyuttaki (bölme) diğer iki örneğini çözmeye indirgedik; bunların birleşimi, ek bir işlemle (en büyük diske aktarım) birlikte aradığımız çözümü (fethetmek) verir.

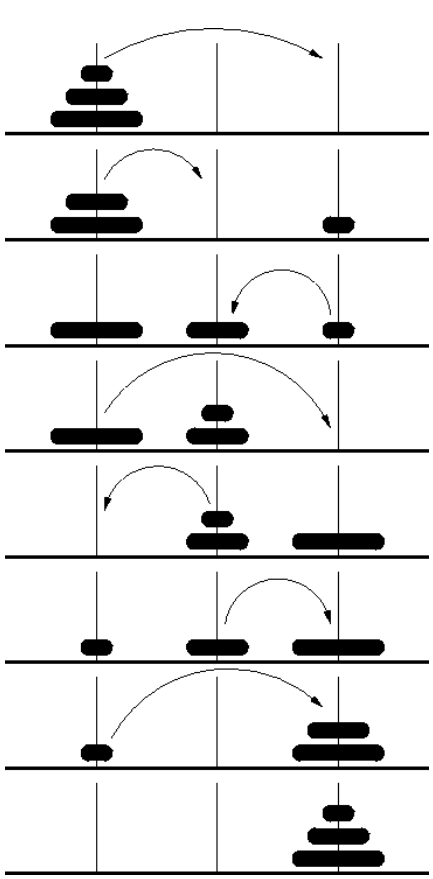
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Böl ve fethet

Hanoi Kulesi Görevi :

- 3 diskli örnek



1 alt problem

+

2 alt problem

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Böl ve fethet

Hanoi Kulesi Görevi:

- 4 diskten oluşan Hanoi kulesinin nasıl çözüldüğünü izleyin *Video demonstration:*

Co-funded by the Erasmus+ Programme of the European Union

Böl ve fethet

Çözüm:

- Böl ve yönet, birkaç güçlü algoritma tasarım tekniğinden yalnızca biridir.
- Böl ve yönet algoritmaları yinelemeler kullanılarak analiz edilebilir.
- Böl ve yönet stratejisi genellikle verimli algoritmalara yol açar.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Böl ve yönet stratejisi nedir?

2. Böl ve fethet stratejisini kullanan sıralama algoritmasını belirtebilir misiniz?

3. Böl ve yönet yöntemiyle çözülen diğer bazı popüler problemleri belirtebilir misiniz?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz !

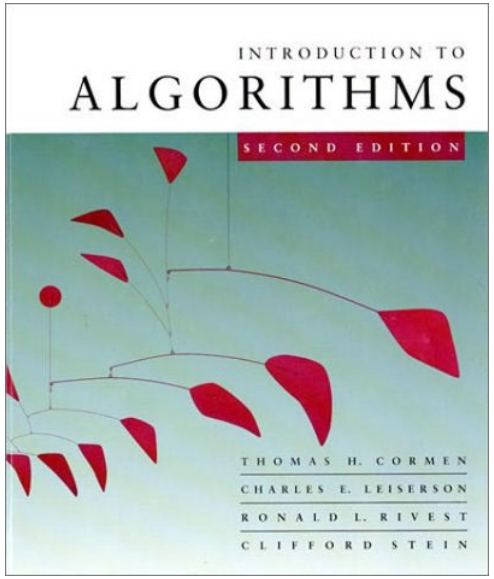
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

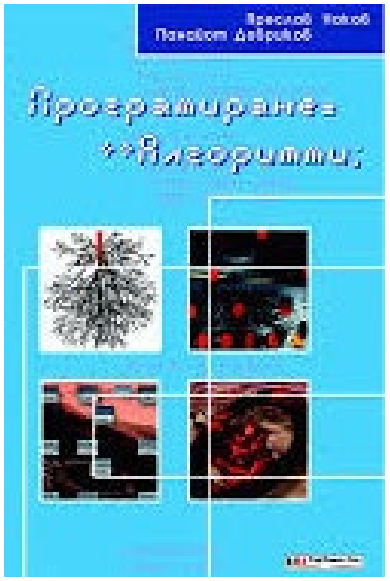
Co-funded by the Erasmus+ Programme of the European Union

VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

- **Modül 3. Sorgu optimizasyonu için algoritmalar ve veritabanlarındaki uygulamaları**
  - **Konu 2. Algoritma Tasarımında Stratejiler**
    - Ders 2. Dinamik programlama

DİNAMİK PROGRAM





Cormen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2009) [1990]. Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill.

Preslav Nakov, Panayot Dobrikov (2002). Programming = ++Algorithms). Top Team Co, София.

Dinamik program

- Dinamik programlama DP (optimizasyon, DO), böl ve yönet yöntemi gibi, alt problemlerin çözümlerini birleştirerek problemleri (görevi) çözer. (Bu bağlamda "programlama", yazılı bilgisayar kodunu değil, tablo şeklinde bir yöntemi ifade eder).
- "Böl ve Fethet" dersinde gördüğümüz gibi, bu algoritmalar problemi (görevi) birbiriyle bağlantısız alt problemlere böler, alt problemleri yinelemeli olarak çözer ve daha sonra orijinal problemi çözmek için bunların çözümlerini birleştirir.

Dinamik program

- Bunun tersine, alt problemler örtüştüğünde, yani alt problemler alt alt problemleri paylaştığında dinamik programlama uygulanır.
- Bu bağlamda böl-yönet algoritması gereğinden fazla iş yapar, sık karşılaşılan alt sorunları tekrar tekrar çözer.
- Dinamik bir programlama algoritması, her alt problemi yalnızca bir kez çözer ve ardından cevabını bir *tablo* biçiminde kaydeder, böylece her alt problemi çözerken cevabı her seferinde yeniden hesaplama işinden kaçınır.

Co-funded by the Erasmus+ Programme of the European Union

Dinamik program

- Optimizasyon problemleri için genellikle dinamik programlama kullanırız. Bu tür sorunların *birçok olası çözümü* olabilir. Her çözümün bir değeri vardır ve biz en uygun (minimum veya maksimum) değere sahip çözümü bulmak istiyoruz.
- Böyle bir çözümü "optimal çözüm" yerine "*problemin optimal çözümü*" olarak adlandırıyoruz çünkü optimal değerin elde edilebileceği birçok çözüm olabilir.

Co-funded by the Erasmus+ Programme of the European Union

Dinamik program

- Dinamik bir programlama algoritması geliştirirken dört adımlı bir sıra izliyoruz :

1. Optimal çözümün yapısını karakterize edin.
2. Optimum çözümün değerini yinelemeli olarak tanımlayın.
3. Tipik olarak aşağıdan yukarıya bir modelde optimal bir çözümün değerini hesaplayın.
4. Hesaplanan bilgilerden en uygun çözümü oluşturun.

Dinamik program

- 1-3 arasındaki adımlar, bir sorunu çözmek için dinamik programlamanın temelini oluşturur.
- Eğer çözümün kendisiyle değil de sadece optimal çözümün değeriyle ilgileniyorsak o zaman 4. adımı atlayabiliriz.
- ***Tanım.*** *Tekrarlanan hesaplamalardan kaçınmak için halihazırda çözülmüş alt problemlerin çözümlerinin sonuçlarıyla bir tablonun doldurulduğu programlama tekniğine dinamik programlama denir.*

Co-funded by the Erasmus+ Programme of the European Union

Dinamik program

- Dinamik programlama, başlangıçtaki problemin alt problemlerini daha küçük boyutta çözmeye ve önceden hesaplanan sonuçları güvenli hale getirmeye dayanır, bellek pahasına hız kazanılır.
- Bazı durumlarda sabit bir belleğe ihtiyaç duyulurken (Fibonacci sayıları), diğer durumlarda ihtiyaç duyulan bellek doğrusal (Sırt Çantası problemi), ikinci dereceden (optimal matris çarpım problemi) ve bazen daha da büyük olabilir.

Dinamik program

- Ayrıca dinamik programlamanın her zaman uygulanabilir olmadığı da unutulmamalıdır.
- Bir yandan, başlangıç probleminin çözümü her zaman alt problemlerin bir kısmının veya tamamının çözüm sonuçlarının birleştirilmesiyle elde edilemez.
- Öte yandan, böyle bir kombinasyon mümkün olsa bile dikkate alınması gereken alt görevlerin sayısı kabul edilemeyecek kadar fazla olabilir.
- Buna, açıklanan yöntem kullanılarak çözülebilecek sorunları karakterize eden açık bir kriterin bulunmaması da eklenmelidir.

Dinamik program

- Bir dizi görev için standart algoritmaların dinamik programlamadan çok daha verimli olduğu ve diğerleri için bu yöntemin hiç uygulanamadığı ortaya çıktı.
- Yöntemin uygulanması için iki gerekli koşul vardır: optimal çözüm altyapısı ve örtüşen alt problemler.

- Optimal çözüm altyapısı, başlangıç probleminin optimal çözümünün, alt problemlerin optimal çözümlerinin bir fonksiyonu olarak bulunabileceği anlamına gelir. Bu, alt görev kümesinin güçlü bir şekilde sınırlandırılmasına, dolayısıyla yöntemin daha yüksek bir verimliliğine yol açar.

Dinamik program

  - Dinamik programlamanın uygulanması için ikinci gerekli koşul alt görevlerin örtüşmesidir. Dinamik programlama, örtüşen alt problemlerden akıllıca faydalanabilir, her alt problemin çözümünü yalnızca bir kez hesaplayabilir, böylece çözülen alt problemlerin gerçek sayısını ciddi şekilde sınırlayabilir.

- Yeni bir alt problemin gerçekten çözülmesine ne kadar az ihtiyaç duyulursa, yöntem o kadar verimli olur. Alt görevlerin örtüşmemesi, dinamik programlama uygulamasının uygunsuz olduğunun kesin bir göstergesidir. Böyle bir durumda böl-yönet yöntemini denemek daha iyidir.

Co-funded by the Erasmus+ Programme of the European Union

Dinamik program

- Genellikle bir problemin çözümünde dikkate alınan nesneler, elemanların kısmen veya tamamen doğrusal olarak sıralandığı kümelerdir.
- Bu tür nesnelerin örnekleri arasında dizeler, çeşitli kombinatoryal konfigürasyonlar (permütasyonlar, kombinasyonlar vb.), sıralı arama ağaçlarının yaprakları, bir çizgi üzerindeki noktalar, bir çokgenin köşeleri vb. yer alır.
- Bu durumlarda dinamik programlama genellikle verimli bir algoritmayla sonuçlanır.

Dinamik program

- Tipik olarak dinamik programlamada problemler aşağıdan yukarıya doğru yinelemeli olarak çözülür. e. , ilk olarak önemsiz alt problemler (daha fazla ayrıştırılmadan doğrudan çözülenler) dikkate alınır, ardından tüm alt problemlerin zaten çözüldüğü daha büyük alt problemler vb. dikkate alınır.
- Bu yaklaşımın ana dezavantajı, başlangıçtaki problemden daha küçük boyuttaki tüm alt problemlerin çözülmesini gerektirmesidir; bu da gereksiz hesaplamalara yol açar, çünkü hesaplama sürecinde bazı alt problemlerin gerekli olmadığı ortaya çıkabilir.

Dinamik program

- Dinamik programlama, ilişkisel veritabanı yönetim sistemlerinde birleştirme sorgularını optimize etmek için iyi bir yöntemdir ve neredeyse tüm ticari optimize ediciler bu amaç için dinamik programlamaya güvenir.
- Dinamik programlama genellikle hafife alınır veya daha doğru bir ifadeyle hafife alınır. Bazı programcılar bunun karmaşık ve dağınık olduğunu düşünüyor.
- Aslında en basit ama etkili algoritmik tekniklerden biridir.
- Bu tür bilgiler somut örneklere dayanarak en kolay şekilde öğrenilebildiğinden, klasik bir örneği ele alacağız.

Co-funded by the Erasmus+ Programme of the European Union

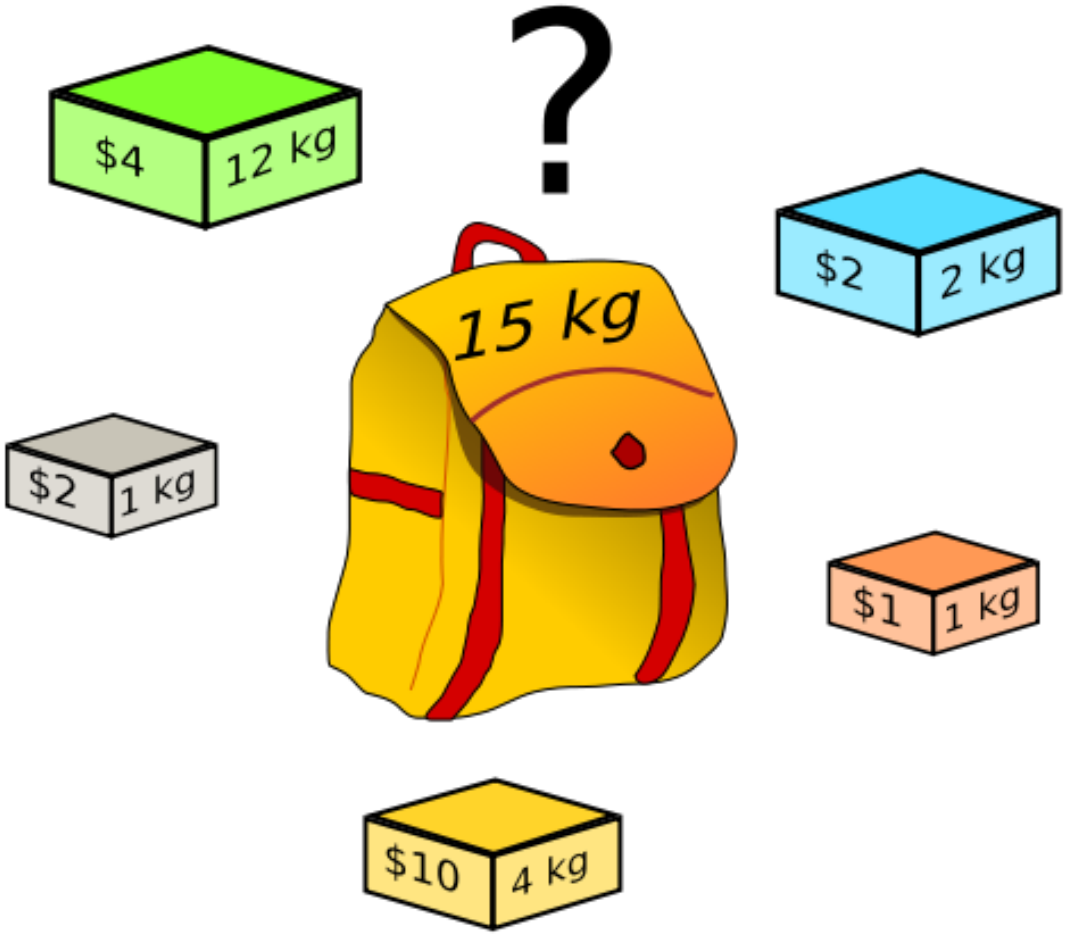
Sırt çantası sorunu

- Bu, dinamik programlama dünyasındaki en ünlü problemlerden biridir ve yöntem kullanılarak verimli bir şekilde çözülmüştür.
- Sırt çantası problemi literatürde birçok farklı formülasyonla karşımıza çıkmakta ve farklı çözümler önermektedir.
- En popüler versiyonunda (0-1 sırt çantası problemi), **dinamik programlamayla** çözülen problemlerin klasik bir örneğidir.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Sırt çantası sorunu



Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Sırt çantası sorunu

Given a knapsack with a capacity of M kilograms and N items for each of which are given two weights m_i and a value c_i. Choose set of items whose total value is maximum and the sum of their weights does not exceed M. M, N, c_i and m_i are natural numbers $(1 \leq i \leq N)$.

The task comes down to finding the maximum of the sum:

$$\sum_{i=1}^{N} x_i c_i$$

under restricted conditions:

$$\sum_{i=1}^{N} x_i m_i \leq M,$$

$$c_i > 0, m_i > 0, x_i \in \{0,1\}, i = 1,2,\ldots,n$$

Restrictions on c_i and m_i are restrictions on condition of the task (*on the concrete example*) while these on x_i should be taken as restrictions on the *solution sought*.

Solution: We will define a recurrent target function $F(i)$, that is a solution for a knapsack with a capacity i.

Than:

$$F(i) = \begin{cases} 0 & i = 0 \\ \max\limits_{j=1,2,\ldots,N; m_j \leq i} \left[c_j + F(i - m_j) \right] & i > 0 \end{cases}$$

Sırt çantası sorunu

Yukarıdaki yinelenen formülden yola çıkarak F'yi hesaplamak için bir fonksiyon uygulayabiliriz. Bu şekilde bir ve aynı öğeyi birden çok kez seçebileceğimizi belirteceğiz. Bu nedenle her F(i) için, bu maksimum değere yol açan somut öğeleri içeren [i] kümesini koruyacağız. Ancak, [i] hedef fonksiyon kümesinin maksimum değerini oluşturan somut bir öğe kümesi bulalım, tekrar aynı öğeyi dahil etmemizi engelleyecektir.

Olası bir yaklaşım, fonksiyon değerini yinelemeli olarak hesaplamak ve tekrarlanan hesaplamaları önlemek için bir tablo kullanmaktır (ezberleme). Zaten hesaplanmış değerlerin tablosu, eğer zaten hesaplanmışsa, hedef fonksiyonun değerini veya NOT_CALCULATED özel değerini içerecektir; aksi halde. Bazı i = 1, 2, … N için F(i) değerine her ihtiyaç duyduğumuzda, önce Fn[i]'nin (kaydedilen tablo değeri) NOT_CALCULATED'dan farklı olup olmadığını kontrol edeceğiz, bu durumda Fn değerini alıyoruz [i] tablodan. Programın giriş verilerini kodunda sabit olarak ayarlayacağız. Ana iş, hedef fonksiyonun değerlerini hesaplayan yinelemeli fonksiyon F () tarafından gerçekleştirilir.

Sırt çantasının kapasitesinin tüm eşyaları almaya izin verip vermediğini kontrol eden hesaplama () fonksiyonu ile yapılır ve eğer böyle değilse F ()'ye döner.

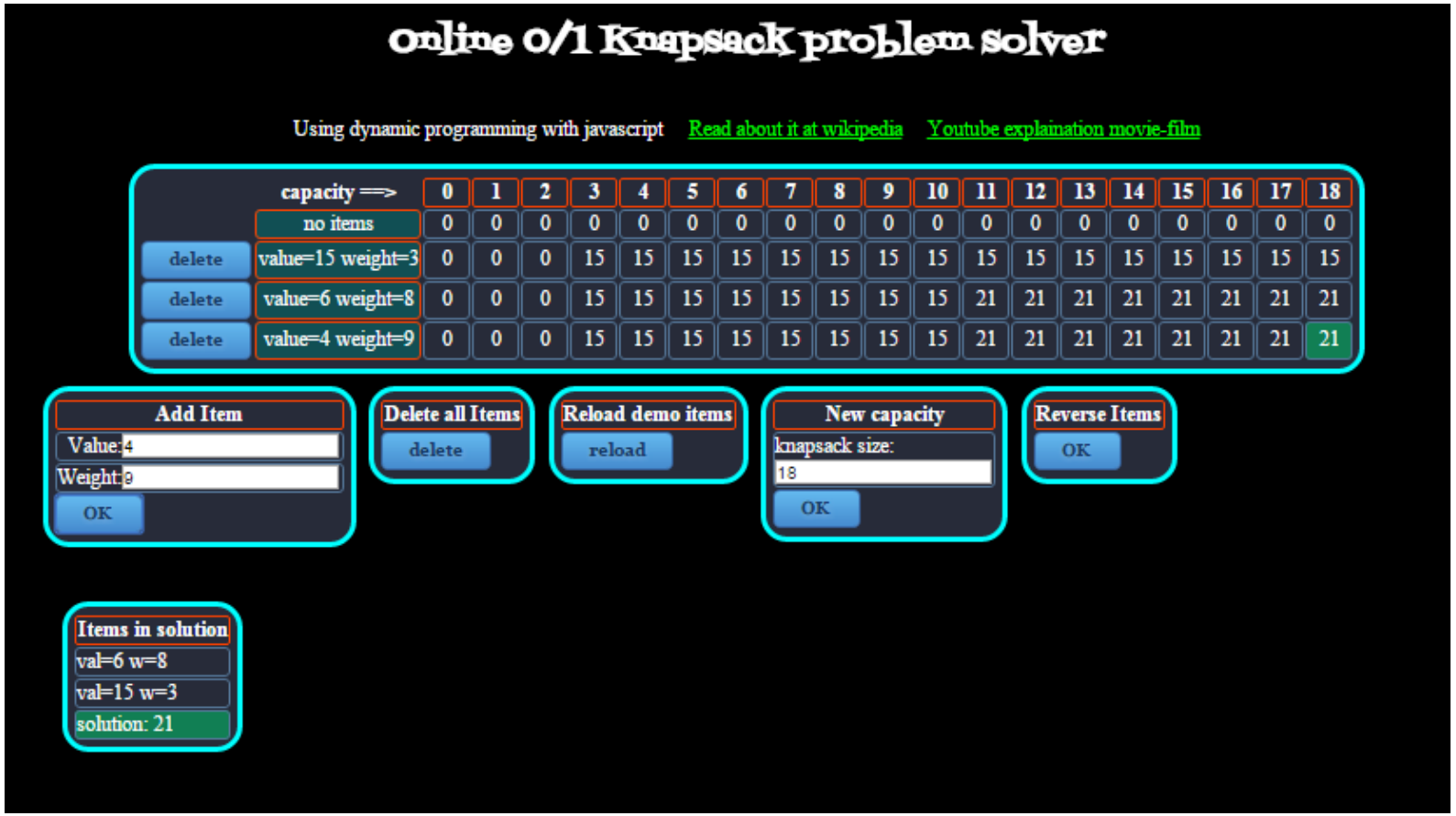
Sırt çantası sorunu

7 parçalı ve 19 kapasiteli sırt çantasıyla örnek.

capacity ==>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
no items	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
value=90 weight=9	0	0	0	0	0	0	0	0	0	90	90	90	90	90	90	90	90	90	90	90
value=85 weight=10	0	0	0	0	0	0	0	0	0	90	90	90	90	90	90	90	90	90	90	175
value=85 weight=10	0	0	0	0	0	0	0	0	0	90	90	90	90	90	90	90	90	90	90	175
value=70 weight=15	0	0	0	0	0	0	0	0	0	90	90	90	90	90	90	90	90	90	90	175
value=89 weight=9	0	0	0	0	0	0	0	0	0	90	90	90	90	90	90	90	90	90	179	179
value=89 weight=12	0	0	0	0	0	0	0	0	0	90	90	90	90	90	90	90	90	90	179	179
value=100 weight=9	0	0	0	0	0	0	0	0	0	100	100	100	100	100	100	100	100	100	190	190

Sırt çantası sorunu

http://karaffeltut.com/NEWKaraffeltutCom/Knapsack/knapsack.html



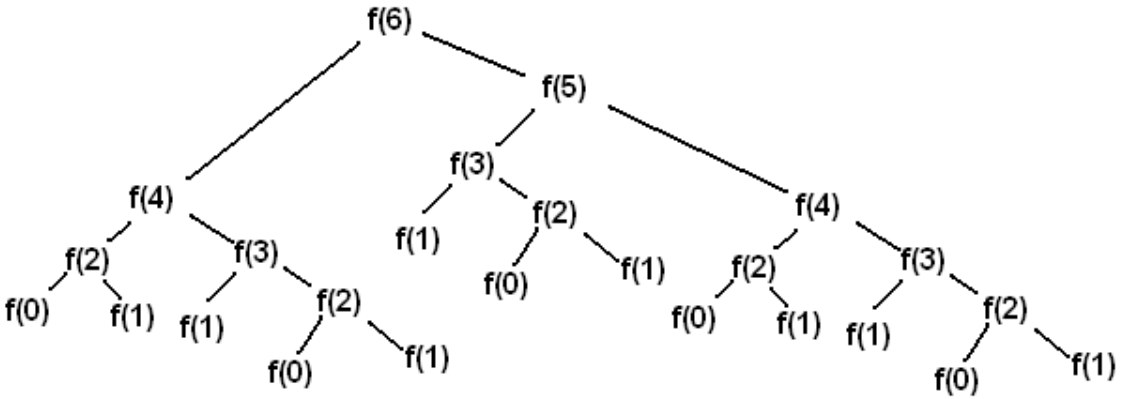
*Farklı örnekleri test edebileceğiniz çevrimiçi hesap makinesi:*https://knapsack.masao.io/

Fibonacci sayıları

- Özyinelemeli fonksiyonla hesaplama:

```
int F(int n)
 {if(n==0) return 0;
  else if(n==1) return 1;
  else return F(n-1)+F(n-2);}
```

- Çok fazla hesaplama:



Fibonacci sayıları

- Fibonacci sayılarının hesaplanması, hesaplanan tüm değerlerin bir *F[i]* dizisinde saklanması yoluyla *n* kadar doğrusal zamana ihtiyaç duyulacak şekilde düzenlenebilir.*:*

```
F[0]=0;
F[1]=1;
for(i=2;i<=n;i++) F[i]=F[i-1]+F[i-2];
```

- Bu program parçası temel fikri göstermektedir - Fibonacci sayılarını küçükten büyüğe doğru sırayla nasıl hesaplayabilir ve aynı zamanda önceki sonuçları koruyabiliriz, böylece F_n 'yi hesaplamamız gerektiğinde, zaten F_{n-i} ve F_{n-2} 'yi hesaplamış oluruz ve bunu yapabiliriz. onları kullan.

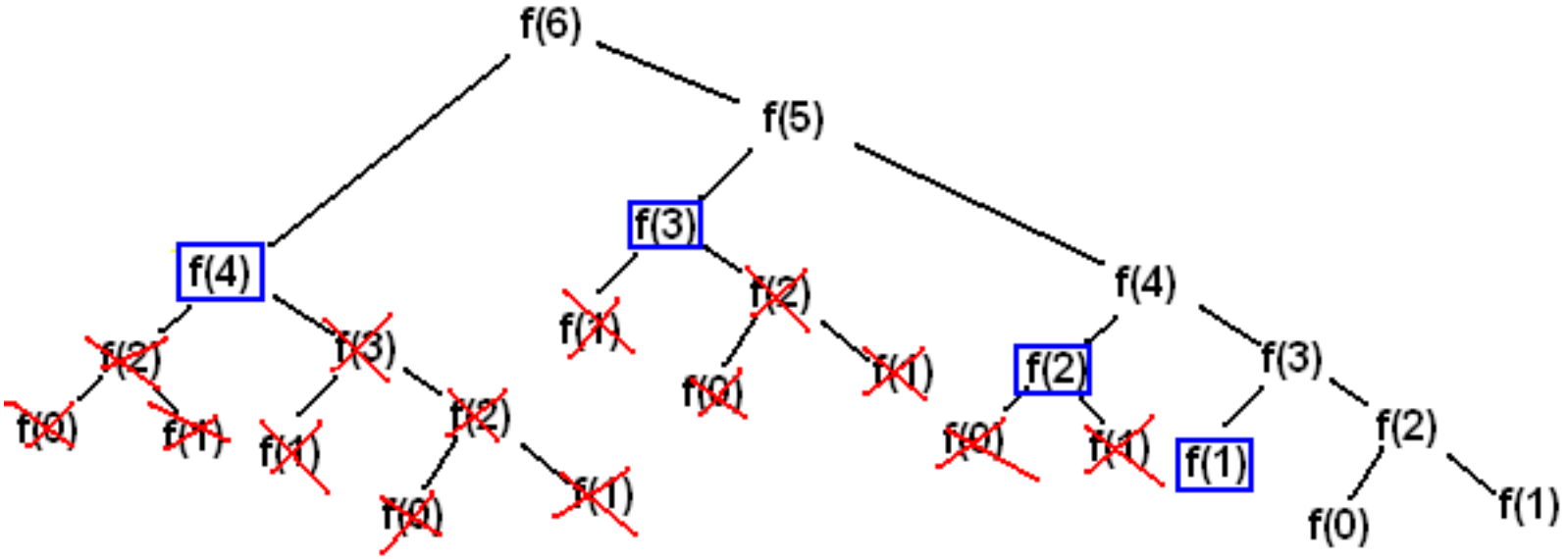
Fibonacci sayıları

- F_n 'yi hesaplamak için o ana kadar hesaplanan tüm sayıları kaydetmemize gerek olmadığını hemen fark edebiliriz.
- Yalnızca önceki iki değerin mevcut olması yeterlidir. Bu, bir dizi kullanmadan, yalnızca iki değişken kullanarak ve değerlerini uygun şekilde değiştirerek yapılabilir:

```
f0=0;
f1=1;
for(i=2;i<=n;i++) {f=f1+f0; f0=f1; f1=f;}
```

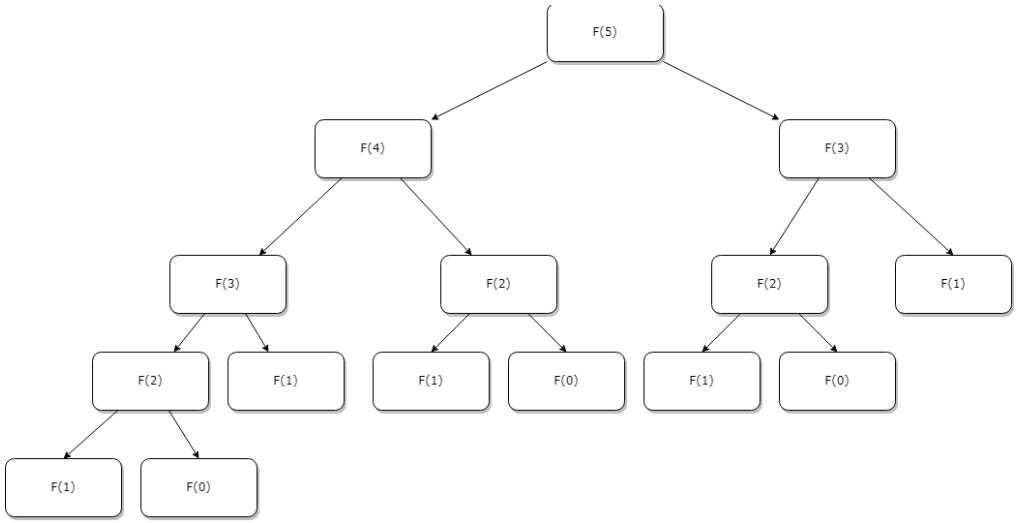
Fibonacci sayıları

- Eğer *f(n)* değeri zaten hesaplanmış ve bir çözüm tablosunda saklanmışsa, tekrar hesaplamak yerine oradan alınabilir. Bu, hesaplamaların alt ağaçlarında kaldırılan örneklerin olduğu anlamına gelir.



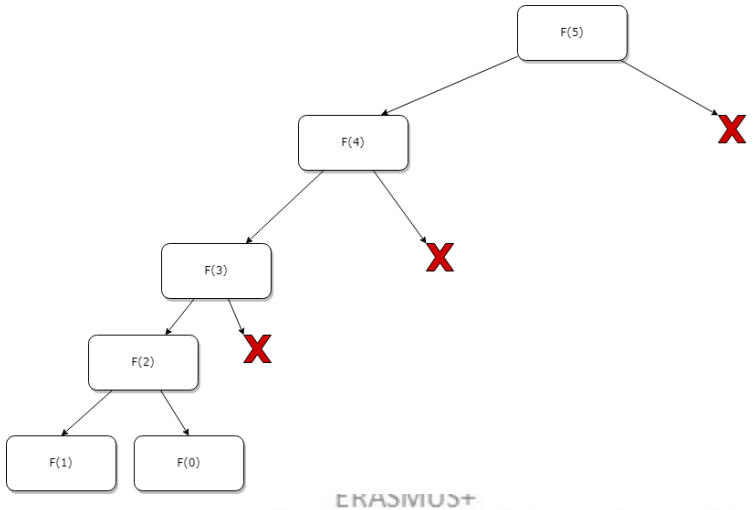
Fibonacci: Yukarıdan Aşağıya ve Aşağıdan Yukarıya Dinamik Programlama

- **Özyinelemeli Yaklaşım :**
  - Özyinelemeli yaklaşımla F(N)'yi hesaplamak için öncelikle F(N-1) ve F(N-2)'nin çözümlerini bulmaya çalışıyoruz. Ancak F(N-1)'i bulmak için F(N-2) ve F(N-3)'ü bulmamız gerekir. Bu, temel durumlara ulaşana kadar devam eder: F(1) ve F(0).



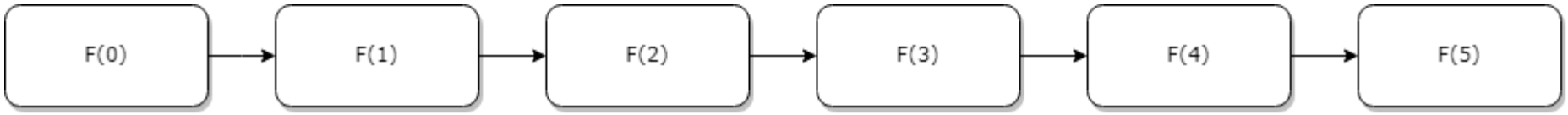
Fibonacci: Yukarıdan Aşağıya ve Aşağıdan Yukarıya Dinamik Programlama

- **Yukarıdan Aşağıya Yaklaşım :**
  - Buradaki fikir özyinelemeli yaklaşıma benzer, ancak fark şu ki, karşılaştığımız alt problemlerin çözümlerini saklayacağız. Bu şekilde, aynı alt problemle birden fazla kez karşılaştığımızda, yeniden hesaplamak zorunda kalmak yerine kayıtlı çözümümüzü kullanabiliriz. Bu, her alt problemi tam olarak bir kez hesaplamamıza olanak tanır.



Fibonacci: Yukarıdan Aşağıya ve Aşağıdan Yukarıya Dinamik Programlama

- **Aşağıdan Yukarıya Yaklaşım :**
  - Aşağıdan yukarıya dinamik programlama yaklaşımında alt problemleri çözme sıramızı yeniden düzenleyeceğiz. F(0), sonra F(1), sonra F(2)'yi vb. hesaplayacağız:



  - Bu, her sorunun çözümünü yalnızca bir kez hesaplamamıza olanak tanıyacak ve aynı anda yalnızca iki ara sonucu kaydetmemiz gerekecek.
  - Örneğin, F(2)'yi bulmaya çalışırken, yalnızca F(1) ve F(0)'ın çözümlerine sahip olmamız gerekir. Benzer şekilde F(3) için yalnızca F(2) ve F(1) çözümlerine ihtiyacımız var.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. DO stratejisi nedir?
2. DP'de ezberleme yaklaşımı nedir?
3. DO tarafından çözülen diğer bazı popüler problemleri belirtebilir misiniz?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

ON-LINE DISTANCE COURSE ON DATABASES

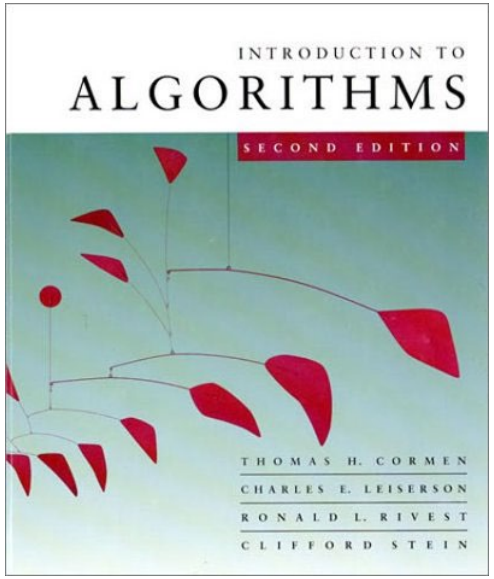
- **Modül 3. Sorgu optimizasyonu için algoritmalar ve veritabanlarındaki uygulamaları**
  - **Konu 2. Algoritma Tasarımında Stratejiler**
    - Ders 3. Sezgisel ve olasılık algoritmaları

ERASMUS+
Key Action KA2 - Cooperation for innovation and the exchange of good practices
Action Type KA226 - Partnerships for Digital Education Readiness

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

SEZGİSEL VE OLASILIKLI (RANDOMİZE) ALGORİTMALAR



Cormen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2009) [1990]. Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill.

Preslav Nakov, Panayot Dobrikov (2002). Programming = ++Algorithms). Top Team Co, Sofia.

Sezgisel ve olasılıksal algoritmalar

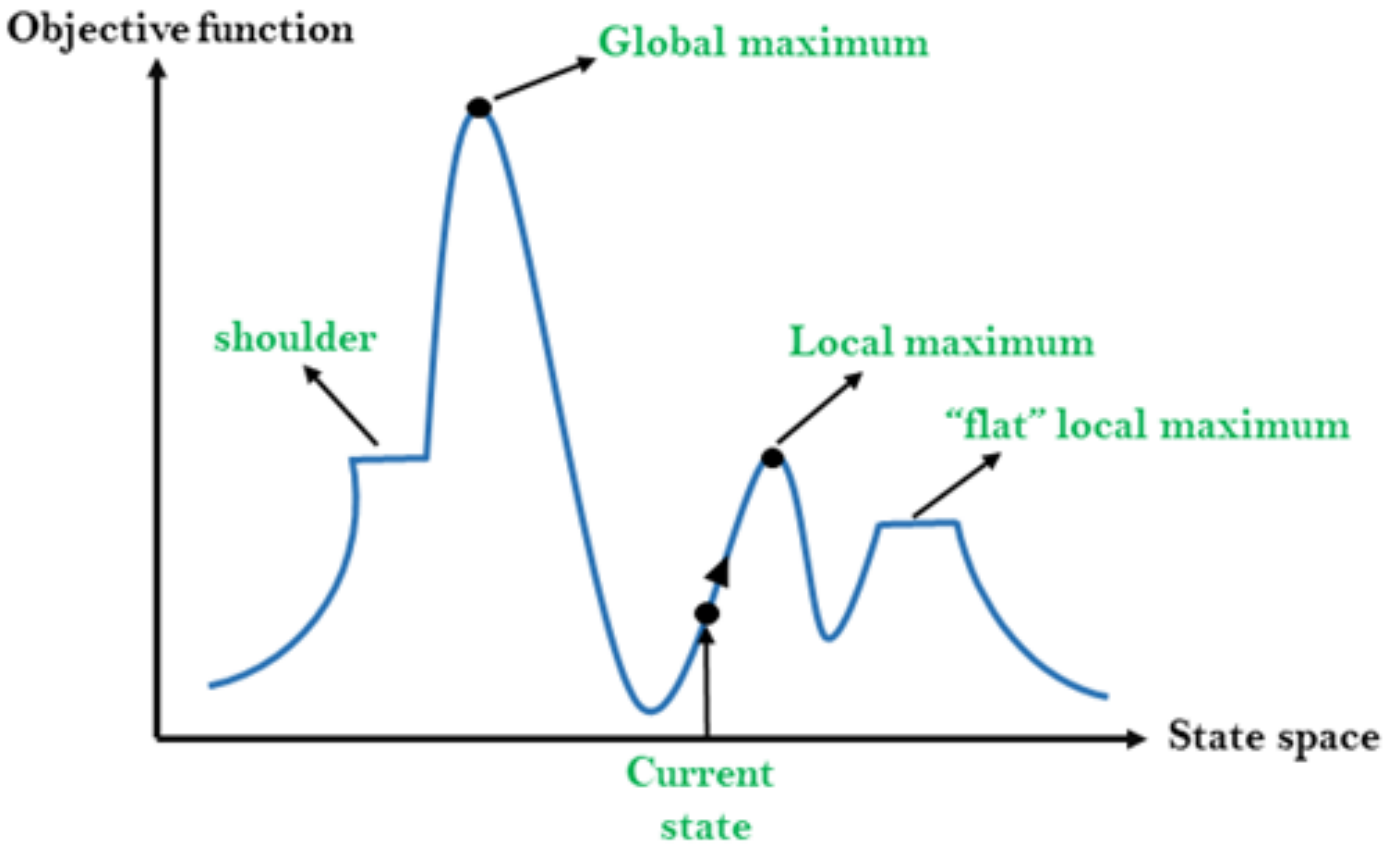
- Sezgisel (**Antik Yunanca'dan**) - bul, keşfet, sorunları çözme, öğrenme veya **deneyime dayalı** sonuçlar çıkarma yöntemlerini ifade eder.

- Bu yöntemlerin optimal olduğu garanti edilmez; bunlar yalnızca sonuca daha hızlı ulaşmanın ve karmaşıklığı azaltmanın bir yoludur.
- Ayrıntılı bir çalışmanın **pratik olmadığı** durumlarda iyi bir çözüm bulma sürecini hızlandırmak için sezgisel yöntemler kullanılır.
- Bu yöntemin örnekleri arasında bilgiye dayalı varsayımların kullanımı ve sezgisel akıl yürütme yer alır.

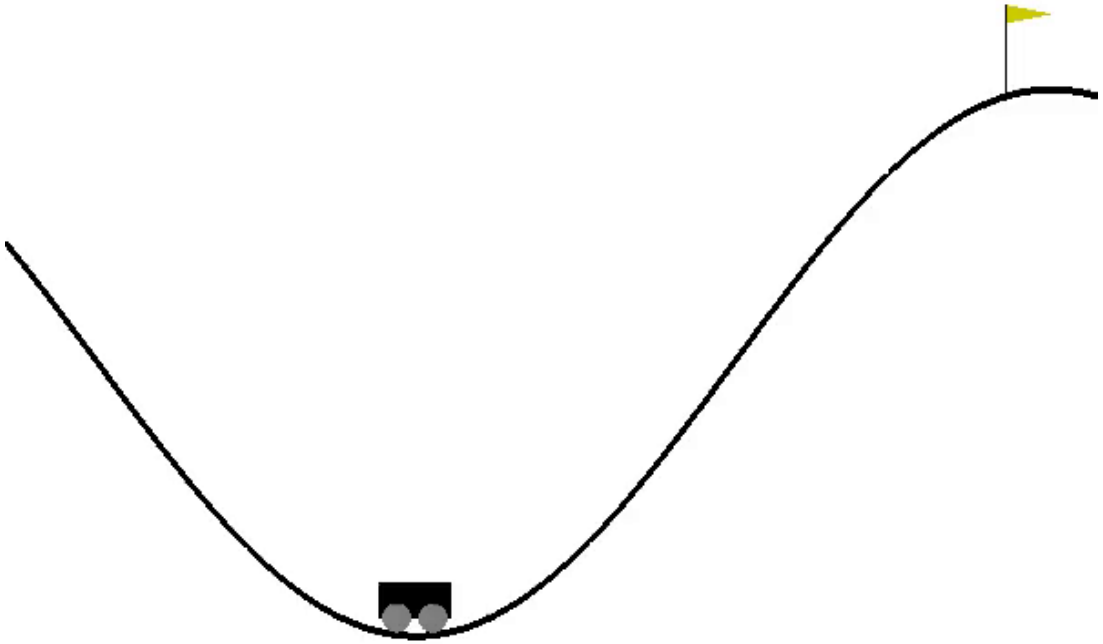
Sezgisel ve olasılıksal algoritmalar

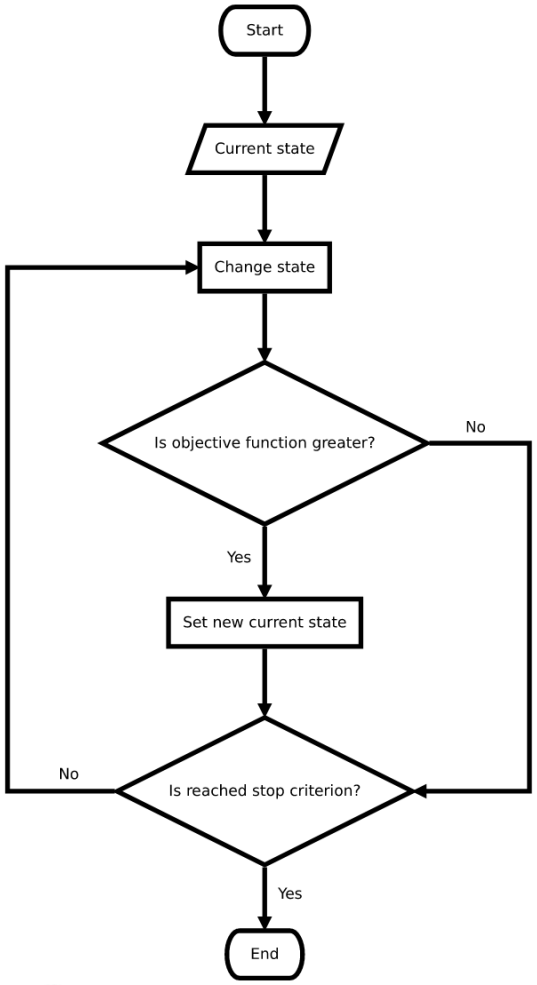
- Bilgisayar bilimlerinde buluşsal veya buluşsal yöntem, **algoritması** olmayan mantıksal veya matematiksel problemleri çözmeye yönelik bir yöntemdir. Yöntem, deneyime dayalı **tümevarımsal akıl yürütme** yoluyla çözüm bulma alanının adım adım daraltılmasını içerir.
- **Tepe tırmanma** bir tür yerel arama algoritmasıdır. Temel fikir, belirli bir uzayda, belirli bir değerlendirme fonksiyonu için diğerlerinden daha iyi bir nokta bulmaktır. Bu aynı zamanda kullanılan anahtar buluşsal yöntemdir. Başlangıç durumu arama uzayında rastgele bir nokta olarak seçilir. Tüm komşu durumları oluşturulur. Değerlendirme fonksiyonuna göre fonksiyon değerinin en büyük (veya probleme bağlı olarak en küçük) olduğu nokta seçilir.

Sezgisel ve olasılıksal algoritmalar



Sezgisel ve olasılıksal algoritmalar





Algoritma, önceden tanımlanmış koşullara ulaştığımızda veya daha fazla iyileştirme bulamadığımızda sona erer.

Sezgisel ve olasılıksal algoritmalar

Olasılıksal bir algoritma, *yürütülmesi sırasında* ***rastgele seçimler yapan*** *bir algoritmadır. Böyle bir algoritmanın çalışması, sabit girdiyle bile rastgele olabilir.*

- Yaklaşık bir çözüm bulmak için, rastgele bir başlangıç çözümü üretilir, daha sonra birbirini izleyen iterasyonlar, öncekilerden daha doğru çözümler üretir.
- Olasılıksal algoritmaların tasarımı ve analizi, girdi olarak sahip olduğumuz şeyden bağımsız olarak çalışmaya odaklanır ve algoritmanın çalışması sırasında yaptığı keyfi seçimlere bağlıdır.

Sezgisel (yaklaşık) *algoritmalar, asıl görevi hafıza veya zaman kısıtlamaları altında optimal değil yaklaşık bir çözüm bulmak olan algoritmalardır. Bu tür algoritmaların örnekleri arasında yerel arama, tabu arama,* ***açgözlü*** *algoritmalar olarak adlandırılan bir buluşsal algoritma sınıfı vb. yer alır.*

Sezgisel ve olasılıksal algoritmalar

- Diyelim ki yeni bir ofis asistanı işe almanız gerekiyor. Daha önceki işe alma girişimleriniz başarılı olmadı ve bir iş bulma kurumuna başvurmaya karar verdiniz.
- İş bulma kurumu size her gün bir aday gönderiyor.
- Bir röportaj yapabilir ve ardından o kişiyi işe alıp almayacağınıza karar verebilirsiniz. Adayla görüşmek için iş bulma kurumuna küçük bir ücret ödemeniz gerekir.
- Ancak mevcut asistanınızı işten çıkarmanız ve iş bulma kurumuna önemli bir ücret ödemeniz gerekir.
- Her zaman mümkün olan **en iyi ofis asistanına sahip olmayı** taahhüt ediyorsunuz.

Sezgisel ve olasılıksal algoritmalar

- Bu nedenle her adayla görüştükten sonra eğer o aday mevcut ofis asistanından daha nitelikli ise mevcut ofis asistanını görevden alıp yeni asistanı işe almaya karar vermelisiniz.
- Bu strateji için gerekli bedeli ödemeye hazırsınız ancak bunun size ne kadara mal olacağını değerlendirmek istiyorsunuz.
- İşe alma prosedürünün sözde kodu:

```
HIRE-ASSISTANT(n)
1  best = 0          // candidate 0 is a least-qualified dummy candidate
2  for i = 1 to n
3      interview candidate i
4      if candidate i is better than candidate best
5          best = i
6          hire candidate i
```

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Sezgisel ve olasılıksal algoritmalar

Dizideki öğelerin konumlarını rastgele değiştiren işe alma prosedürünün sözde kodu:

(Bu durumda aday listesinin rastgele bir permütasyonuna sahip olacağız)

```
RANDOMIZED-HIRE-ASSISTANT(n)
1  randomly permute the list of candidates
2  best = 0          // candidate 0 is a least-qualified dummy candidate
3  for i = 1 to n
4      interview candidate i
5      if candidate i is better than candidate best
6          best = i
7          hire candidate i
```

Açgözlü algoritmalar

- Çoğu problemde mümkün olan çözümler arasından en uygunu aranır.
- Çoğu durumda, optimal çözümü bulmak için, problemin tüm alt durumlarının çözümlerini bulmak gerekir (mutlaka optimal değildir).
- Önemli bir dezavantaj, bazılarının birden çok kez yeniden hesaplanmasıdır.
- Dinamik optimizasyon uygulanarak aynı alt vakaların tekrar tekrar hesaplanmasından kaçınılabilir, ancak ne yazık ki ikincisi, sonuçları depolamak için yeterli belleğe ihtiyaç duyulmasını gerektirir.

Açgözlü algoritmalar

- Sezgisel algoritmaların ardındaki fikir şudur: Sezgisel algoritma, sorunun tüm alt durumlarından birini hedef alır ve doğru alt durum olacağı "ümidiyle" yalnızca o alt durumu çözer.
- Bu alt durumun seçimi yerel optimallik kriterine dayanmaktadır.
- Örneğin, **açgözlü algoritmalar**, adından da anlaşılacağı gibi, yerel olarak bakarak her zaman o an için en iyi seçime odaklanır ve oldukça doğal olarak daha sonraki bir aşamada, küresel olarak bakıldığında bu seçimin en iyi seçim olmadığı ortaya çıkabilir..

Açgözlü algoritmalar

- Açgözlü algoritmaların oluşturulması kolaydır, algoritmanın karşılık gelen uygulaması karmaşık değildir ve tek dezavantajı bazen sorunun doğru çözümünü garanti etmemeleridir.
- Bununla birlikte, yukarıdakiler onların kullanışlılığını azaltmaz; buluşsal algoritmaların (ve özellikle açgözlü olanların) en iyiye yakın bir çözümü hızla bulmayı başarmaları karakteristiktir.
- Pek çok pratik görevde, tüm durumları araştırmak imkansızdır ve neredeyse sonsuz ve umutsuz bir "doğru" arayışı alternatifiyle karşılaştırıldığında, genellikle optimal çözümden %5 daha kötü bir çözüm bulan bir algoritma formüle etmek, bir başarı olarak kabul edilir. en uygun çözüm.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

Açgözlü algoritmalar

- **The first problem** we will consider is the following: find a way to obtain a given sum m (m is a natural number), using a minimum number of banknotes, with denominations from the set $C = \{a_1, a_2, ..., a_n\}$. For example, the banknotes are *1, 2, 5, 10, 20, 50 leva*.
- *Consider the following algorithm:*

1) We initialize $s=0$

2) We find the banknote *i* with a maximum value a_i $(a_i \epsilon C)$, so that $s+a_i \leq m$.

2.1) If there isn't a banknote for which $s+a_i \leq m$ it follows that the problem has no solution. End.

2.2) Otherwise we take the banknote *i* and increase *s* by a_i.

2.2.1) If $s=m$ it follows that the problem is solved. End.

2.2.2) If $s<m$ than we still haven't obtained the whole sum and repeat step 2).

- For example, for the sum of 298 leva, five 50 leva notes, two 20 leva notes, one 5 leva note and one two and one leva note will be chosen in turn (total 250 + 40 + 5 + 2 + 1 = 298).

Açgözlü algoritmalar

- Açıkçası, açıklanan algoritma açgözlü bir algoritmanın kriterlerini karşılamaktadır: her adımda maksimum yüksek değerli banknotu seçer, böylece aranan miktara mümkün olduğu kadar hızlı ulaşmayı amaçlar.
- Bu özel örnekte, soruna etkili bir çözüme yol açmaktadır.
- Örneğin notaların olası değerlerinin 2, 5, 20 ve 30 olduğu ve toplamın 40 olmasını istediğimiz durumu düşünün.
- Açgözlü algoritma önce 30'luk bir nota (mümkün olan maksimum) seçecek, ardından 5'lik iki nota, toplamda 3 nota seçecektir.

Co-funded by the Erasmus+ Programme of the European Union

Açgözlü algoritmalar

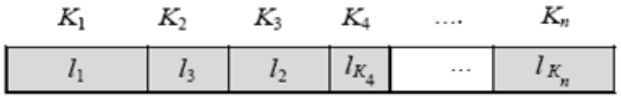
- **Ancak elbette daha iyi bir çözüm var: iki adet 20 nota.**
- Optimum çözümü bulamamanın yanı sıra, açgözlü algoritmanın hiç çözüm bulamaması da mümkündür.
- Yani örneğin 6'lık bir toplam elde etmek istiyorsak: 5 değerine sahip notadan sonra algoritmaya başka seçenek kalmaz (ve toplam yine de 2 değerine sahip 3 notadan elde edilebilir).
- Ancak işler her zaman o kadar da kötü değildir ve açgözlü algoritmanın her zaman bir çözüm bulduğunun gösterilebileceği bir takım sorunlar vardır.

Açgözlü algoritmalar

Manyetik bant görevi

l_1, l_2, ... , l_n uzunluğunda n program ve sıralı erişime sahip bir manyetik şerit vardır.

- Sıralı erişim, bant üzerinde belirli bir konumda bulunan bir kaydı okumak için, belirtilen konuma geçmeniz (kasetin kaydırılması) gerektiği anlamına gelir.



Magnetic stripe with sequential access *(K1=1, K2=3, K3=2, ...)*

- Örneğin, üçüncü programı (l_2 uzunluğunda) okumak için, ondan önceki tüm programların (l_1 ve l_3 uzunluğundakiler) "kaydırılması" gerekir.

Açgözlü algoritmalar

Manyetik bant görevi

Given the probabilities of each program $p_1, p_2, \ldots, p_n$, $0 \le 1$, $1 \le i \le n$ $\sum_{i=1}^{n} p_i = 1$. For example a probability 0.5 means that the program runs as often as all the other programs together. In fixed ordering $K_1, K_2, \ldots, K_n$ of the programs on the tape the average time needed for loading a random program is defined the following way:

$$T = \frac{\sum_{i=1}^{n}\left(p_{K_i} \sum_{j=1}^{i} l_{K_j} \right)}{n}$$

The task is to find an ordering of the programs on the tape wich has to minimalize T.

We're going to use a greedy algorithm. It is intuitively clear that as often a program is used as forward on the tape it should be written. The other criterion is the lenght of the program - as longer it is as backward on the tape it should be (to "hinder" as little as possible when scrolling is needed). Thus, we reach the following algorithm: We write the programs sequentially on the tape and in each step we choose the program K_i for which the ratio p_{Ki}/l_{Ki} is maximum.

The correctness of the last algorithm can be easily proved: it is enough to calculate how the sum will change if two items K_i and K_j change their positions in the final arrangement. We leave the immediate implementation to the reader.

Co-funded by the Erasmus+ Programme of the European Union

Genetik algoritmalar

- Genetik algoritmalar, temel fikrinin doğadaki genetik ve evrimsel süreçleri simüle etmek olduğu bir ara-bul yöntemidir.
- Böylece, belirli bir optimizasyon problemi için başlangıçta birkaç rastgele alt optimal çözüm oluşturulur.
- En başarılı olanlar tutulur ve daha da iyi olmaları umuduyla yenileri temel alınarak inşa edilir. Bu şekilde, optimal olmayan ve ümit vermeyen çözümler daha fazla değerlendirmeden izole edilir.

Co-funded by the Erasmus+ Programme of the European Union

Genetik algoritmalar

- Türlerin evrim süreciyle olan benzerlik açıktır; doğada bu ifadeye "en uygun olanın hayatta kalması" denir.
- Genetik algoritmalar, yapay zekanın hızla büyüyen alanı olan evrimsel programlamanın bir parçasıdır.
- Algoritma, **popülasyon** adı verilen bir dizi çözümle (kromozomlarla temsil edilir) çalıştırılır. Bir popülasyondan alınan kararlar yeni popülasyon tarafından alınır ve kullanılır.

Genetik algoritmalar

- Bu, yeni **nüfusun** eskisinden daha iyi olacağı umuduyla haklı çıkar.
- Yeni **popülasyonu** (nesli) oluşturmak için seçilen çözümler, yaşayabilirliklerine göre seçilir; daha uygun olanların üreme şansı daha yüksektir.
- Bu, bazı koşullar (örneğin, **popülasyon** sayısı veya en iyi çözümün kanıtlanması) sağlanana kadar tekrarlanır.

Genetik algoritmalar geniş bir problem sınıfına uygulanabilir ancak yaklaşımın en büyük dezavantajı, iyi çözümleri garanti eden teorik değerlendirmelerin eksikliğidir.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Genetik algoritmalar

Genetik Algoritmanın Taslağını Çizmek

1. **[Başla]** N kromozomdan oluşan rastgele bir popülasyon oluşturma (uygun problem çözümleri)
2. **[Canlılık]** Popülasyondaki her bir kromozom n'nin canlılığının f(x) hesaplanması
3. **[Yeni popülasyon]** Yeni popülasyon bitene kadar aşağıdaki adımları tekrarlayarak yeni bir popülasyon oluşturma
4. **[Seçim]** Yaşayabilirliklerine göre popülasyondan iki ebeveyn kromozomunun seçilmesi (daha iyi yaşayabilirlik, daha iyi seçim şansı)
5. **[Melezleme]** Melezleme yoluyla muhtemelen yeni nesli (çocukları) oluşturmak için ebeveynleri çaprazlarsınız. Melezleme olmasaydı nesil ebeveynlerinin tam bir kopyası olurdu.
6. **[Mutasyon]** Mutasyon yoluyla, muhtemelen yeni nesil bir yerde (kromozomdaki yer) mutasyona uğrar.
7. **[Kabul Ediliyor]** Yeni nesli yeni popülasyona yerleştirin
8. **[Değiştirme]** Algoritmanın daha ileri düzeyde yürütülmesi için yeni oluşturulan popülasyonun kullanılması
9. **[Kontrol Et]** Eğer son koşul karşılanıyorsa, durun ve mevcut popülasyonun en iyi çözümünü döndürün.
10. **[Döngü] 2.** adıma git

Sorgu Optimizasyonunda Buluşsal Yöntemin Kullanımı

- Sezgisel optimizasyon süreci
  1. Üst düzey bir sorgunun ayrıştırıcısı bir başlangıç dahili temsili oluşturur;
  2. Dahili gösterimi optimize etmek için buluşsal yöntem kurallarını uygulayın.
  3. Sorguda yer alan dosyalarda mevcut olan erişim yollarına dayalı olarak işlem gruplarını yürütmek için bir sorgu yürütme planı oluşturulur.

- Ana buluşsal yöntem, ilk önce ara sonuçların boyutunu azaltan işlemleri uygulamaktır.

- Örneğin, JOIN veya diğer ikili işlemleri uygulamadan önce SELECT ve PROJECT işlemlerini uygulayın.

Sorgu Optimizasyonunda Buluşsal Yöntemin Kullanımı

- **Sorgu ağacı** :
  - İlişkisel cebir ifadesine karşılık gelen bir ağaç veri yapısı. Sorgunun giriş ilişkilerini ağacın yaprak düğümleri olarak temsil eder ve ilişkisel cebir işlemlerini iç düğümler olarak temsil eder.
  - Sorgu ağacının yürütülmesi, işlenenleri mevcut olduğunda bir dahili düğüm işleminin yürütülmesinden ve ardından bu dahili düğümün, işlemin yürütülmesinden kaynaklanan ilişkiyle değiştirilmesinden oluşur.

Sorgu grafiği :

  - İlişkisel hesaplama ifadesine karşılık gelen bir grafik veri yapısı. Hangi işlemlerin ilk önce gerçekleştirileceği sırasını belirtmez. Her sorguya karşılık gelen yalnızca tek bir grafik vardır.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Sezgisel algoritmalar kesin algoritmalar mıdır?

2. Sezgisel algoritmaları kullanmak ne zaman pratiktir?

3. Rastgele algoritma nedir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

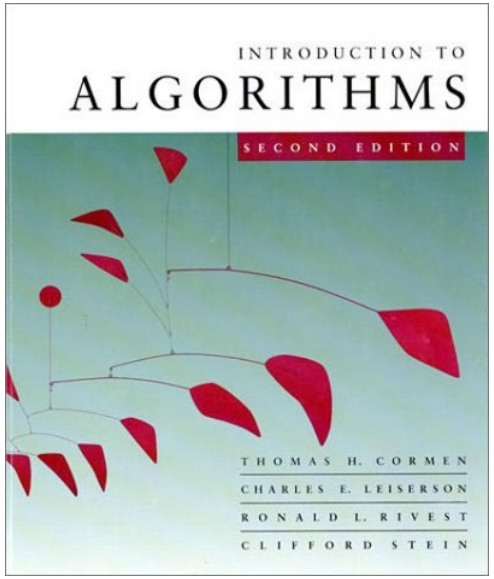
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

- **Modül 3. Sorgu optimizasyonu için algoritmalar ve veritabanlarındaki uygulamaları**
  - **Konu 2. Algoritma Tasarımında Stratejiler**
    - Ders 4. Açgözlü algoritmalar. Örnekler.

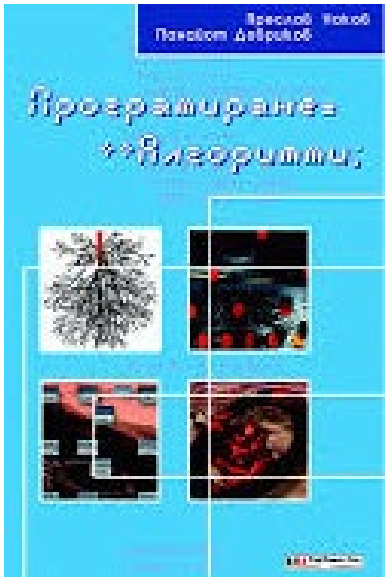
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

AÇGÖZLÜ ALGORİTMALAR. ÖRNEKLER.



Carmen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2009) [1990]. Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill.



Preslav Nakov, Panayot Dobrikov (2002). Programming= ++Algorithms). Top Team Co, София.

Açgözlü Algoritmalar

- Optimizasyon problemlerine yönelik algoritmalar genellikle her adımda bir dizi seçenek içeren bir dizi adımdan geçer.
- Pek çok optimizasyon probleminde, en iyi seçimi belirlemek için dinamik programlamayı kullanmak gereksizdir (çok güçlü bir silahtır); daha basit, daha verimli algoritmalar bunu yapabilir.
- Açgözlü bir algoritma her zaman o anda en iyi görünen seçimi yapar.
- Yani, bu seçimin küresel bir optimal çözüme yol açacağı umuduyla yerel bir optimal seçim yapar.

Açgözlü Algoritmalar

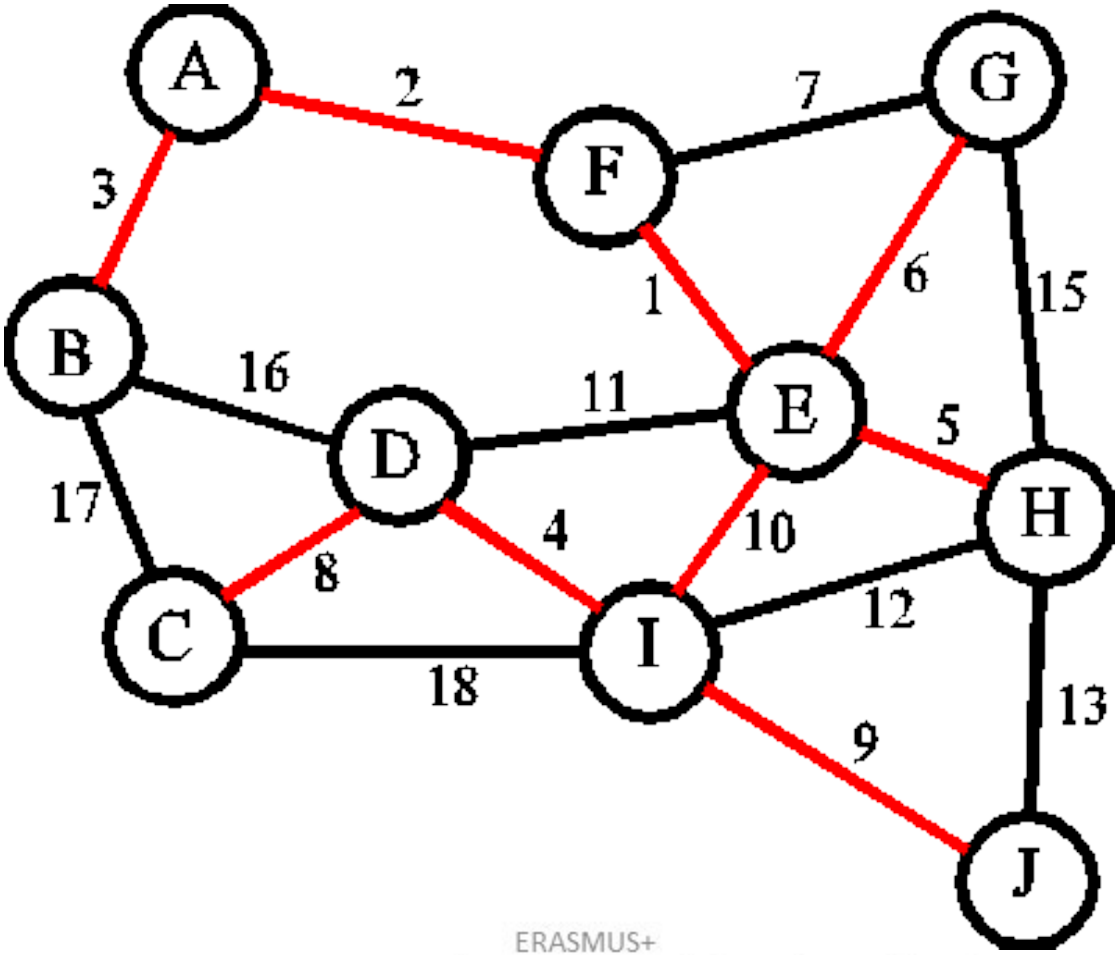
- **Açgözlü Algoritmalar** her zaman optimal çözümler sunmasa da birçok problem için böyle bir çözüm bulmaktadır.
- Genel olarak açgözlü yöntem oldukça güçlüdür ve çok çeşitli problemlerde işe yarar.
- Daha sonra (sonraki derslerde), minimum kaplama ağaçları, Djextra en kısa yol algoritması vb. dahil olmak üzere açgözlü yöntemin uygulamalarını görebileceğimiz birçok algoritma sunacağız.
- Maksimum yayılan ağacı bulmaya yönelik algoritmalar açgözlü algoritmanın klasik bir örneğini temsil eder.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Açgözlü Algoritmalar

Kapak ağacı



Açgözlü Algoritmalar

- **Açgözlü Algoritmaların** oluşturulması kolaydır, algoritmanın uygulanması karmaşık değildir ve tek dezavantajı bazen problemin doğru çözümünü garanti etmemeleridir.
- Bununla birlikte, bu onların kullanışlılığını azaltmaz; sezgisel algoritmaların (ve özellikle açgözlü olanların) en iyiye yakın bir çözümü hızla bulmayı başarmaları karakteristiktir.
- Pek çok pratik problemde, tüm durumları araştırmak imkansızdır ve neredeyse sonsuz ve ümitsiz bir "doğru" arayışı alternatifiyle karşılaştırıldığında, genellikle optimal çözümden %5 daha kötü bir çözüm bulan bir algoritma oluşturmak, bir başarı olarak kabul edilir. en uygun çözüm.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

Açgözlü Algoritmalar

- **Ele alacağımız ilk problem şudur:** C = {a1, a2, ..., BİR}. Örneğin banknotlar 1, 2, 5, 10, 20, 50 levadır.
- *Aşağıdaki algoritmayı göz önünde bulundurun :*

1) s=0'ı başlat

2) s+ai≤m olacak şekilde maksimum ai (aiϵc) değerine sahip i banknotunu bulun.

2.1) Hedefi olan bir banknot yoksa sorunun çözümü yoktur. Son.

2.2) Aksi takdirde i banknotunu alırız ve ai ile s'yi artırırız.

2.2.1) Eğer s=m ise problem çözülür. Son.

2.2.2) Eğer s<m ise hala toplam toplamı elde edemedik ve 2) adımını tekrarlayın.

Örneğin 298 leva toplamı için sırasıyla beş adet 50 leva, iki adet 20 leva, bir adet 5 leva ve bir adet 2 leva ve bir adet 1 leva banknot seçilecektir (toplam 250 + 40 + 5 + 2 + 1 = 298).

Açgözlü Algoritmalar

- Açıkçası, açıklanan algoritma açgözlü bir algoritmanın kriterlerini karşılamaktadır: her adımda maksimum değerli banknotu seçer, böylece aranan miktara mümkün olduğu kadar hızlı ulaşmayı amaçlar. Bu özel örnekte, sorunun etkili bir şekilde çözülmesine yol açmaktadır.
- Örneğin, banknotlar için olası değerlerin 2, 5, 20 ve 30 olduğu ve 40 miktarını elde etmek istediğimiz durumu düşünün.
- Açgözlü algoritma önce 30'luk bir nota (mümkün olan maksimum), ardından 5'lik iki nota seçecektir, yani toplamda 3 not.

Açgözlü Algoritmalar

- **Ancak elbette daha iyi bir çözüm var: iki adet 20 nota.**
- Optimum çözümü bulamamanın yanı sıra, açgözlü algoritmanın **hiç çözüm bulamaması** da mümkündür.
- Yani örneğin 6'lık bir toplam elde etmek istiyorsak: 5 değerli notadan sonra algoritmaya başka seçenek kalmaz (ve toplam yine de 2 değerli 3 notadan elde edilebilir).
- Ancak işler her zaman bu kadar kötü değildir ve açgözlü algoritmanın her zaman bir çözüm bulduğunun gösterilebileceği bir takım sorunlar vardır.

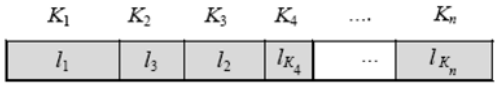
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

Açgözlü Algoritmalar

Manyetik şerit sorunu

$l_1, l_2, \ldots, l_n$ uzunluğunda n program ve sıralı erişime sahip bir manyetik şerit verildiğinde

- Sıralı erişim, şerit üzerinde belirli bir konumda bulunan bir kaydı okumak için, belirtilen konuma geçmeniz (şeridi kaydırmanız) gerektiği anlamına gelir.



Magnetic stripe with a sequential access ($K_1 = 1, K_2 = 3, K_3 = 2, \ldots$).

- Örneğin, üçüncü programı (l_2 uzunluğunda) okumak için, ondan önceki tüm programların (l_1 ve l_3 uzunluğundakiler) "kaydırılması" gerekir.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Açgözlü Algoritmalar

Magnetic stripe problem

Given the probabilities of execution of each program $p_1, p_2, \ldots, p_n$, $0 \le p_1 \le 1, \le i \le n, \sum_{i=1}^{n} p_i = 1$

For example a probability 0.5 means the program is running as often as all the other programs together. In fixed ordering $K_1, K_2, \ldots, K_n$ of the programs on the stripe the *average time T* needed for loading a random program is defined as follows:

$$T = \frac{\sum_{i=1}^{n}\left(p_{K_i} \sum_{j=1}^{i} l_{K_j} \right)}{n}$$

The task is to find an ordering of the programs on the stripe minimizing T.

We are going to use a greedy algorithm. It is clear that as often a program is used further on the stripe it should be written. The other defining criterion is the length of the stripe – as longer it is as backwards on the stripe it should be (to "hinder" less when it needs to be scrolled). Thus, we reach the following algorithm: Write sequentially the programs on the stripe and on each step we choose program K_i for which the ratio pk_i/lk_i is maximum.

The correctness of the last algorithm can be easily proved: it is sufficient to calculate how the sum will change if two members K_i and K_j change their positions in the final ordering.

Açgözlü Algoritmalar

Maksimum aktivite kombinasyonu

Problem: Öğretilecek n ders veriliyor (gerçekleştirilecek etkinlikler). Her ders i iki sayıyla tanımlanır: sabit bir başlangıç s_i ve sabit bir son f_i.

s_i ve f_i sayıları dersin başlangıç ve bitiş zamanını gösterebilir; bunları doğal sayılar olarak kabul edebiliriz.

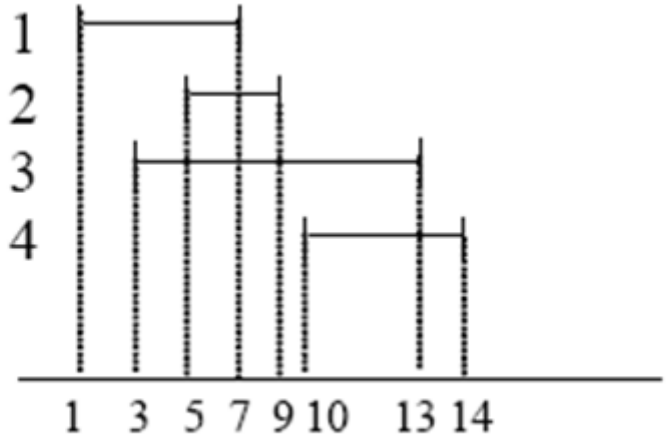
Seçilen derslerden ikisinin aynı anda verilmemesi için maksimum sayıda ders seçilmelidir, yani tek bir dershanemiz olduğunu varsayarsak, herhangi bir zamanda salonda en fazla bir ders verilebilir.

i ($s_i \leq t \leq f_i$, $1 \leq i \leq n$).

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

Açgözlü Algoritmalar

Maksimum aktivite kombinasyonu



Configuration of four lectures: beginning and end.

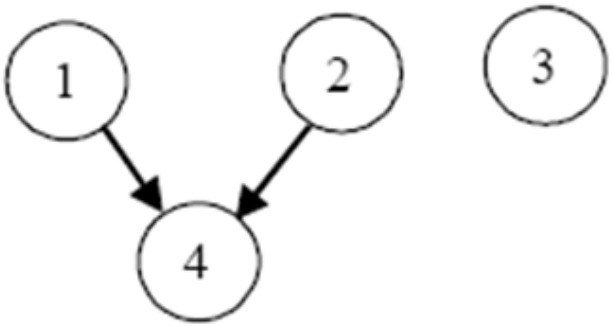
Sorun, verimli ve doğru çalışan açgözlü algoritmanın klasik bir örneğidir. 1 ve 4 numaralı dersler zaman açısından kesişmez, 1 ve 2, 2 ve 3 vb. dersler kesişir (yani aynı anda yapılamaz).

Açgözlü Algoritmalar

Maksimum aktivite kombinasyonu

Algoritma 1:

her iki "kesişmeyen" ders *i, j*'nin $(f_i < s_j)$ grafiğin bir *(i, j)* kenarına karşılık geldiği, *n* köşeli yönlendirilmiş bir *G(V, E)* grafiği oluşturuyoruz. Eğer *G*'deki en uzun yolu bulursak, karmaşıklığı $\Theta(n^2)$ olan bir algoritma elde ederiz.



Configuration graph

Açgözlü Algoritmalar

Maksimum aktivite kombinasyonu

Algoritma 2:

Dinamik programlamayı kullanarak problemi grafik oluşturmadan çözmek mümkündür. 0'dan t'ye kadar zaman için seçilen ders sayısını aşağıdaki şekilde maksimuma çıkaran bir hedef fonksiyon F tanımlarız:

F(t) = 0, eğer fi < t olan bir ders çıkmıyorsa ve

F(t) = maksimum {F(si) + 1} aksi halde $\underset{\substack{i=1,2,\ldots n \\ f_i<t}}{}$

Çözüm $F(t_0)$ fonksiyonunun değeridir. $t_0 = \max_{i=1,2,\ldots n} \{f_i\}$

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Açgözlü Algoritmalar

Maksimum aktivite kombinasyonu

Algoritma 3:

Sorun açgözlü algoritma kullanılarak basit ve etkili bir şekilde çözülebilir; burada ek bellek kullanılmadan $\Theta(n.\log_2 n)$ karmaşıklığı elde edilir. Açgözlü algoritmanın karmaşıklığı doğrusaldır; sorun, derslerin f_i'ye göre artan sırada sıralanması gerektiğidir. Ve genel durumda sıralama $\Theta(n.\log_2 n)$ karmaşıklığına sahip olduğundan, çözümün kendisinin karmaşıklığını tanımlar.

Algoritma:

Dersleri baştan sona sırayla ele alıyoruz:

1) Fi'nin minimum olduğu i dersini seçiyoruz (başlangıçta her zaman ilkini seçin çünkü dersler fi'ye göre artan sırada sıralanır)

2) sj≤fi (yani seçilen derse "engelleyen") tüm j dersleri ders listesinden hariç tutulur. 1. adım tüm dersler öğretilene kadar tekrarlanır.

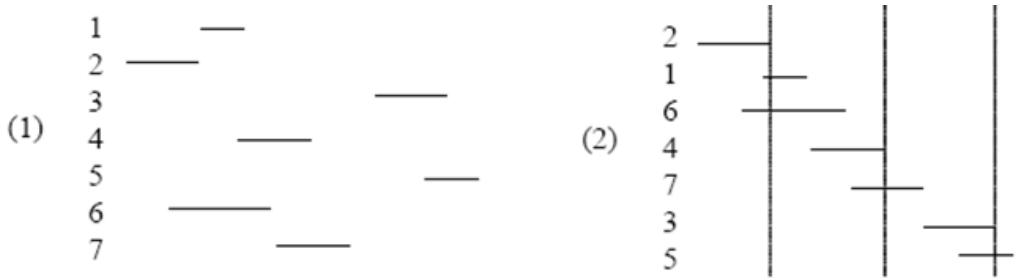
Sorting the lectures and finding the maximum combination

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Açgözlü Algoritmalar

Maksimum aktivite kombinasyonu



Choosing the lectures and find the maximum combination

Fig. (2) shows the lectures from (1) sorted by f_i. The greedy algorithm will choose lecture 2, than will skip lektures 1 and 6, and choose 4, skip 7, choose 3 and skip 5. The algorithm scheme as a pseudocode looks generally like this:

```
Choose_lecture_1
i = 1; j = 1;
while (j <= n) {
  j++;
  if (s[j] > f[i]) {
    Choose_lecture_j
    i = j;
  }
}
```

Açgözlü Algoritmalar

Maksimum aktivite kombinasyonu

follows the source code of the program (we assume that the lectures are already sorted by f_i) . The input data are set at the beginning as constants.

```
#include <stdio.h>
#define MAXN 100

const unsigned n = 7;
const unsigned s[MAXN] = { 3, 7, 5, 9, 13, 15, 17 };
const unsigned f[MAXN] = { 8, 10, 12, 14, 15, 19, 20 };

void solve(void)
{ unsigned i = 1, j = 1;
  printf(" selected lectures : %u ", 1);

  while (j++ <= n)
    if (s[j - 1] > f[i - 1]) printf("%u ", i = j);
  printf("\n");
}

int main(void) {
  solve();
  return 0;
}
```

Açgözlü Algoritmalar

Açgözlü stratejinin unsurları:

- Açgözlü bir algoritma, bir dizi seçim yaparak bir problemin optimal çözümünü elde eder. Her karar noktasında algoritma o anda en iyi görünen seçimi yapar. Bu buluşsal strateji her zaman optimal bir çözüm üretmez, ancak aktivite seçimi probleminde gördüğümüz gibi bazen üretir.
- Daha genel olarak açgözlü algoritmaları aşağıdaki adımlara göre tasarlarız:

1. Optimizasyon problemini, bir seçim yaptığımız ve çözmemiz gereken bir alt problemin kaldığı bir problem olarak ele alalım.

Co-funded by the Erasmus+ Programme of the European Union

Açgözlü Algoritmalar

Açgözlü stratejinin unsurları :

- Daha genel olarak açgözlü algoritmaları aşağıdaki adımlara göre tasarlarız :

  2. Açgözlü seçimi yapan orijinal problemin her zaman en uygun çözümünün olduğunu kanıtlayın, böylece açgözlü seçim her zaman güvenli olur.

  3. Açgözlü seçim yaptıktan sonra geriye kalanın, alt problemin optimal çözümünü yaptığımız açgözlü seçimle birleştirirsek orijinal problemin optimal çözümüne ulaşacağımız özelliğine sahip bir alt problem olduğunu göstererek optimal altyapıyı gösterin. Sorun

  Her açgözlü algoritmanın neredeyse her zaman daha hantal bir dinamik programlama çözümü vardır.

Açgözlü Algoritmalar

Açgözlü seçim özelliği :

- İlk temel bileşen açgözlü seçim özelliğidir: yerel olarak en uygun (açgözlü) seçimler yaparak küresel olarak en iyi çözümü bir araya getirebiliriz.

- Başka bir deyişle hangi seçimi yapacağımızı düşünürken alt problemlerden elde edilen sonuçları dikkate almadan mevcut problemde en iyi görünen seçimi yaparız.

Co-funded by the Erasmus+ Programme of the European Union

Açgözlü Algoritmalar

Açgözlü seçim özelliği :

- **Açgözlü algoritmaların dinamik programlamadan farklı olduğu nokta burasıdır**. Dinamik programlamada her adımda bir seçim yaparız ancak seçim genellikle alt problemlerin çözümlerine bağlıdır.
- Sonuç olarak, dinamik programlama problemlerini genellikle aşağıdan yukarıya doğru çözüyoruz, daha küçük alt problemlerden daha büyük alt problemlere doğru ilerliyoruz. (Alternatif olarak yukarıdan aşağıya ama ezberleyerek de çözebiliriz.
- Elbette kod yukarıdan aşağıya çalışsa da, bir seçim yapmadan önce yine de alt problemleri çözmemiz gerekiyor.) Açgözlü bir algoritmada, o anda en iyi görünen seçimi yaparız ve sonra kalan alt problemi çözeriz.

Açgözlü Algoritmalar

Açgözlü seçim özelliği :

- Açgözlü bir algoritmanın yaptığı seçim o ana kadar yapılan seçimlere bağlı olabilir ancak gelecekteki seçimlere veya alt problemlerin çözümlerine bağlı olamaz.
- Böylece, ilk seçimi yapmadan önce alt problemleri çözen dinamik programlamadan farklı olarak açgözlü bir algoritma, herhangi bir alt problemi çözmeden önce ilk seçimini yapar.
- Dinamik bir programlama algoritması aşağıdan yukarıya doğru ilerlerken, açgözlü bir strateji genellikle yukarıdan aşağıya doğru ilerler, açgözlü seçimleri birbiri ardına yaparak, verilen her problem örneğini daha küçük bir duruma indirir.

Açgözlü Algoritmalar

Özet :

- Her adımda belirli bir kümenin maksimum (veya bazı durumlarda, algoritmanın neyi başarmak istediğine bağlı olarak minimum) öğesinin (yerel olarak) seçilmesiyle uygulanan herhangi bir algoritma (yani bu kümedeki > ilişkisi tanıtılır), " Bunun küresel maksimuma ulaşacağını umma çabasına Açgözlü algoritma denir. Bu algoritma sınıfına adını veren de bu "o anda alabildiğini al" stratejisidir.
- Basit ve "tek yönlü"dürler; gelecekte ne olacağı konusunda endişelenmeden, şu anda sahip olduğumuz bilgilere dayanarak kararlar alırlar.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Açgözlü Algoritmalar

Özet :

- Açgözlü algoritmalar aldıkları kararları asla gözden geçirmezler. Bu, onları, verilen kararın o ana kadar alınan tüm olası kararlara dayandığı dinamik optimizasyondan ayırır; dolayısıyla algoritmamızın en uygun olanı çıkardığından eminiz.
- Algoritmanın çalıştırılması bittiğinde, eğer elde ettiğimiz maksimum değer global maksimumla eşleşiyorsa (yani en iyi çözüm), o zaman algoritmamız doğrudur, aksi takdirde açgözlü algoritma optimale çok yakın bir yanıt verir..
- Açgözlü algoritmaların uygulanması, dinamik optimizasyon algoritmalarının aksine kolaydır ve çoğu durumda oldukça etkilidir. Bu özellikler onları rekabet sorunlarının çözümünde iyi bir seçim haline getiriyor.
- **Sorgu iyileştiricileri, başvurulan tablolara katılma sırasını belirlemek için Greedy adı verilen daha hızlı bir yöntem kullanabilir.**

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Açgözlü Algoritma Nedir?
2. Açgözlü Algoritmalar Ne İçin Kullanılır?
3. Dinamik Programlama ile Açgözlü Algoritmalar arasındaki fark nedir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131
Co-funded by the Erasmus+ Programme of the European Union

VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

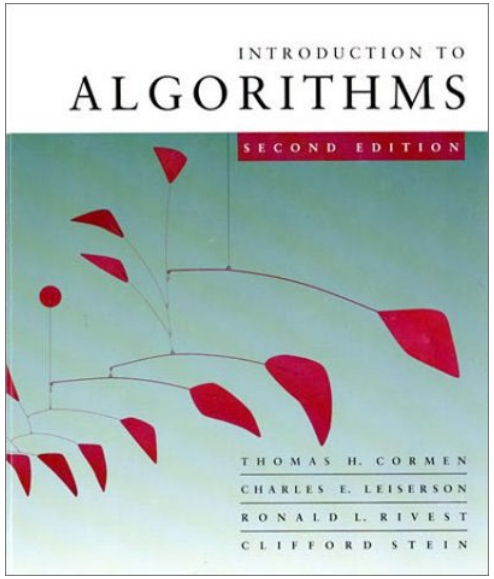
- **Modül 3. Sorgu optimizasyonu için algoritmalar ve veritabanlarındaki uygulamaları**
  - **Konu 3. Sıralama algoritmaları**
    - Ders 1. Sıralama algoritmaları - bölüm I (Eklemeye göre sıralama. Doğrudan seçime göre sıralama. Kabarcık yöntemi).

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

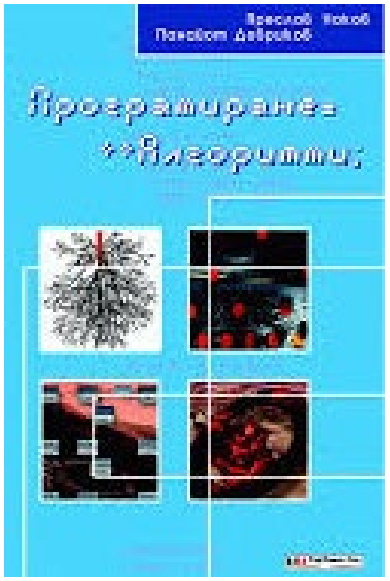
Co-funded by the Erasmus+ Programme of the European Union

Bölüm I Sıralama

Sıralama algoritmaları. Ekleme sıralaması. Seçim sıralaması. Kabarcık yöntemi.



Cormen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2009) [1990]. Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill.



Preslav Nakov, Panayot Dobrikov (2002). Programming = ++Algorithms). Top Team Co, София.

Co-funded by the Erasmus+ Programme of the European Union

Sıralama

- Çoğu zaman, aynı türden büyük verilerle çalışırken, bunların daha kolay işlenmesi için bir tür düzenlemenin getirilmesi gerekir.
- Öğelerin sıralanması, verilerin sıralanmadığı duruma kıyasla bize önemli ölçüde daha verimli bir arama algoritması sağlayabilir.
- Bazı nesne kümelerinin öğelerinin belirli bir düzende yeniden düzenlenmesi (uygun şekilde izin verilmesi) işlemine sıralama denir.

Co-funded by the Erasmus+ Programme of the European Union

Sıralama

- Sıralama, sorgu işlemede kullanılan birincil algoritmalardan biridir. Örneğin, bir **SQL sorgusu** ORDER BY yan tümcesini belirttiğinde, sorgu sonucunun sıralanması gerekir.
- Sıralama aynı zamanda JOIN ve diğer işlemler (UNION ve INTERSECTION gibi) için kullanılan sıralama-birleştirme algoritmalarında ve PROJECT işlemi için yinelenen eleme algoritmalarında (bir SQL sorgusu SELECT yan tümcesinde DISTINCT seçeneğini belirttiğinde) önemli bir bileşendir.

Sıralama

- Algoritmaların çeşitliliği o kadar fazladır ki Knuth ünlü monografisi "Bilgisayar Programlama Sanatı"nın üçüncü cildinin tamamını (800 sayfanın üzerinde) sıralamaya (ve aramaya) ayırmıştır.
- Pascal dilinin yaratıcısı Niklaus Wirth de Algoritmalar + Veri Yapıları = Programlar bölümünde bunlara oldukça fazla önem veriyor.
- Yukarıda belirtildiği gibi, bir kümenin elemanlarını sıralama süreci genellikle onları yeniden düzenlemekten ibarettir. Şartları daha sıkı tanımlayacağız.

Co-funded by the Erasmus+ Programme of the European Union

Sıralama

- M kümesi elemanlarla birlikte verilsin :

$$a_1, a_2, \ldots, a_n,$$

ve üzerlerinde tanımlanmış bir *f* fonksiyonu.

- M'nin elemanlarını sıralayarak, bunların uygun sıraya göre sıralanmasını kastediyoruz:

$$a_{i_1}, a_{i_2}, \ldots, a_{i_n}$$

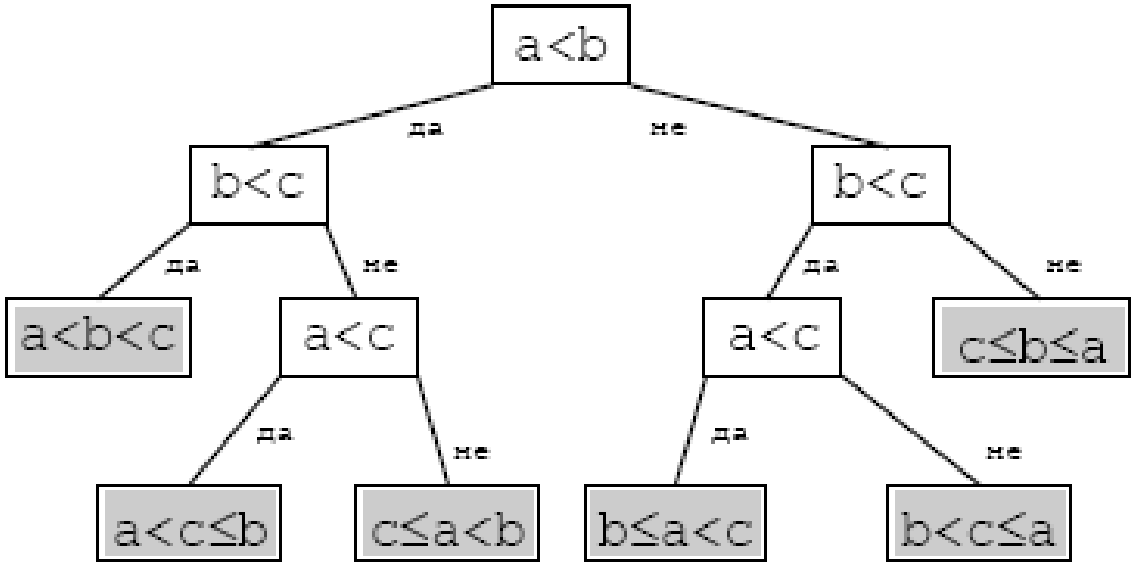
böylece $f(a_{i_1}) \leq f(a_{i_2}) \leq \ldots \leq f(a_{i_n})$. yerine getirildi.

f fonksiyonuna küme sıralama fonksiyonu denir.

Karşılaştırma sıralaması

- Bunlar, izin verilen tek işlemin, işlemleri kullanan öğe çiftleri arasında karşılaştırma olduğu klasik algoritmalardır. $< (\leq), > (\geq)$ и $= (\neq)$.

{a,b,c} kümesi için karşılaştırma ağacı



Karşılaştırma sıralaması

- Bu muhtemelen karşılaştırmaya göre sıralamanın en temel yöntemidir. Her biri bize ek bilgi sağlayan bir dizi karşılaştırmaya dayanmaktadır.
- Set tamamen sıralanıncaya kadar işlem devam eder.
- $x < y$ biçimindeki herhangi bir karşılaştırmanın iki sonucu olduğunu düşünebiliriz: *x, y*'den küçükse evet, aksi halde hayır.
- Yöntemi üç elemanlı *{a, b, c}* kümesi üzerinde göstereceğiz.

Co-funded by the Erasmus+ Programme of the European Union

Karşılaştırma sıralaması

```
if (a < b)
   if (b < c) printf("a,b,c");
   else
      if (a < c) printf("a,c,b");
      else printf("c,a,b");
else
   if (b < c)
      if (a < c) printf("b,a,c");
      else printf("b,c,a");
   else printf("c,b,a");
```

- Yukarıdaki algoritma şu açıdan güzel ve kullanışlıdır:

1) minimum sayıda karşılaştırmayı garanti eder;

2) açıkça karşılaştırma ağacını verir (minimum zaman karmaşıklığını kanıtlamak için kullanılır).

Co-funded by the Erasmus+ Programme of the European Union

Karşılaştırma sıralaması

- Avantajları maalesef burada bitiyor.
- Yeterince az sayıda öğe olsa bile, çıktı kümesinin öğelerinin olası permütasyonları olduğu kadar olası çıktıların (ağacın yaprakları) sayısının n! olduğunu fark etmek zor değil. oldukça büyük bir sayı, ilgili bir programın yazılmasına izin vermiyor.
- Daha etkili başka ayırma yöntemleri aranmalıdır.

Co-funded by the Erasmus+ Programme of the European Union

Ekleme sıralama

- Karşılaştırmaya göre sıralama için üç klasik temel evrensel yöntem vardır: ekleme, seçme ve kabarcık yöntemi.
- Temel sıralama yöntemleri nispeten az sayıda öğe (yaklaşık 20) için etkilidir ve pratikte sıklıkla kullanılır.
- Ne yazık ki, daha fazla sayıda öğe olduğunda hızları keskin bir şekilde yavaşlıyor, bu nedenle başka yöntemlerin kullanılması gerekiyor.
- Aslında, üç temel yöntemin tümü, piramidal veya hızlı sıralama gibi modern sıralama yöntemlerinin karakteristik özelliği olan $O(n.log_2\ n)$ karmaşıklığıyla karşılaştırıldığında çok daha yavaş olan bir algoritmik karmaşıklık $O(n^2)$ ile karakterize edilir..

Ekleme sıralama

- Ekleme sıralama, kartları sol elinde tutan oyuncunun, kartları birer birer sağ eline aldığı ve doğru konumlarına yerleştirdiği, kart sıralamanın iyi bilinen yöntemidir.
- Kartın doğru konuma yerleştirilmesi, doğru konum bulunana kadar önceden istiflenmiş kartlarla daha tutarlı bir şekilde (gözle) karşılaştırılmasını gerektirir.
- Ekleme nasıl yapılır? Açık bir algoritma, x'in solundaki bir öğeyle sıralı olarak karşılaştırılması ve nihai değişimidir. İşlem aşağıdaki durumlardan biri oluşana kadar devam eder:

1) x'ten küçük veya ona eşit bir anahtarla bir öğeye ulaşmak;

2) dizinin ilk elemanına ulaşmak.

Co-funded by the Erasmus+ Programme of the European Union

Ekleme sıralama

The method implements the following idea.

If the row

$a_0, a_1, \ldots, a_{i-1}$

is sorted in ascending order, the element a_i is included in the row so that it takes the "correct" position, i. e. the new row

$a_0, a_1, \ldots, a_i, \ldots, a_{i-1}$

is sorted in ascending order. These actions are repeated for $i = 1, 2, \ldots, n-1$

Co-funded by the Erasmus+ Programme of the European Union

Ekleme sıralama

Source code of insertion sorting algorithm of C++ language

// A procedure that inserts elements in a sorted in ascending order row

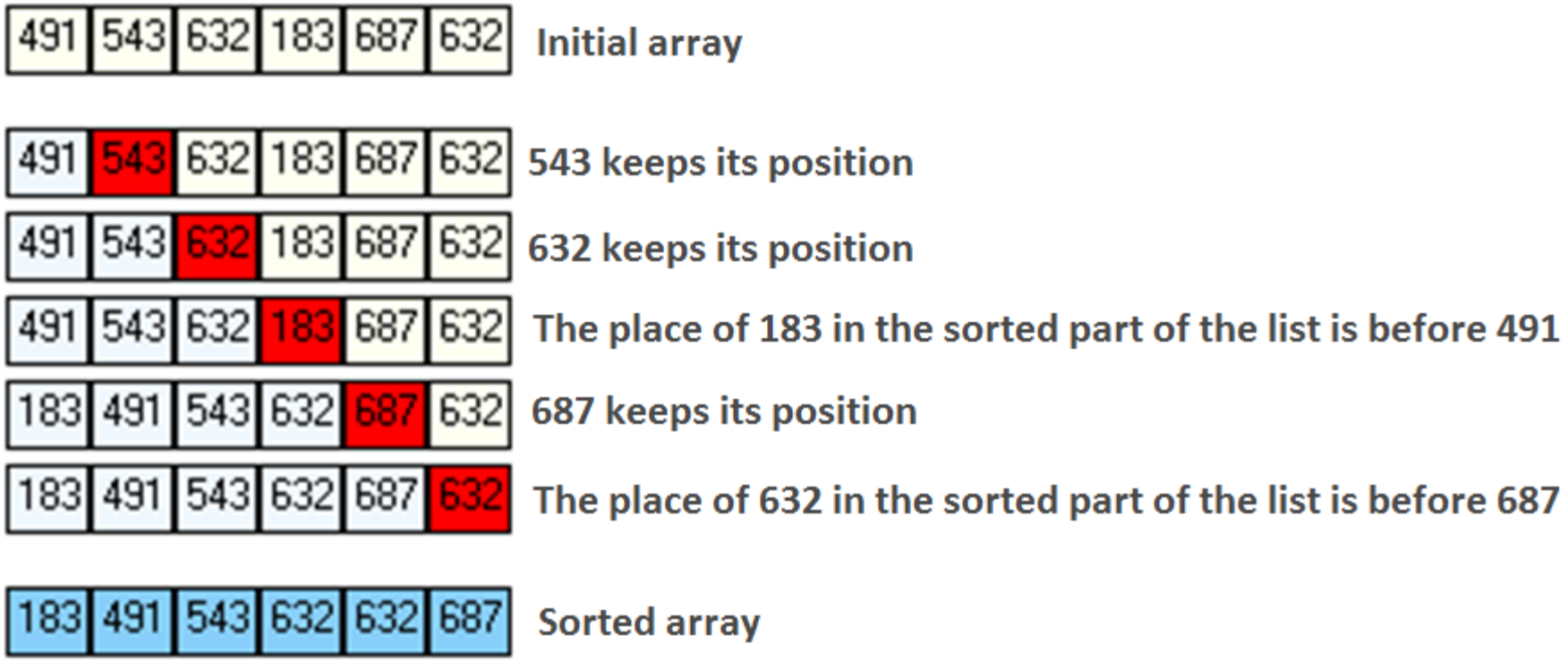
```
void Insert(int a[], int i)
{
  int j;
  int x = a[i];
  for(j = i - 1; j >= 0; j --)
    if(x < a[j])
      a[j+1] = a[j];
    else
      break;
  a[j+1] = x;
}
```

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Ekleme sıralama

ÖRNEK: Ekleme sıralaması.



Co-funded by the Erasmus+ Programme of the European Union

Doğrudan seçim yöntemine göre sıralama

- Yöntemin fikri şu şekildedir: En küçük eleman aranır, dizinin başına taşınır ve değerlendirme dışı bırakılır.
- İşlem, n elemanın tümü seçilene kadar tekrarlanır.

For $i = 0, 1, \ldots, n-2$ the sequence is considered :

$a_i, a_{i+1}, \ldots, a_{n-1}$

and the following actions are performed:

- Find k, so that $a_k = \min\{a_i, a_{i+1}, \ldots, a_{n-1}\}$.
- exchange the values of a_l and a_i.

Co-funded by the Erasmus+ Programme of the European Union

Doğrudan seçim yöntemine göre sıralama

function implementing sorting by the method of direct selection

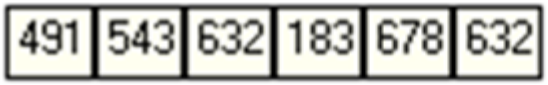
```
void SelectSort(int a[], int n)
{
  int min, k;
  for(int i = 0; i < n; i ++)
  {
    k = i;
    min = a[k];
    for(int j = i + 1; j < n; j ++)
      if(a[j] < min)
      {
        k = j;
        min = a[k];
      }
    a[k] = a[i];
    a[i] = min;
  }
}
```

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

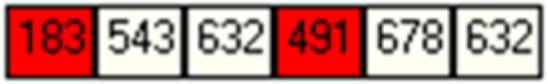
Co-funded by the Erasmus+ Programme of the European Union

Doğrudan seçim yöntemine göre sıralama

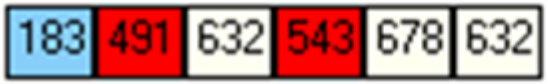
EXAMPLE: **Sorting by direct selection method**



Initial array



exchange 491 and 183 (the minimum in the unsorted part)



exchange 543 and 491 (the minimum in the unsorted part)

exchange 632 and 543 (the minimum in the unsorted part)

no exchanges are made

exchange 678 and 632 (the minimum in the unsorted part)

Sorted array

KABARCIK SIRALAMA YÖNTEMI

Nispeten küçük bir n için tavsiye edilir.

Yalnızca artan satırda sıralama yapıyoruz :

$a_0, a_1, \ldots, a_{n-1}$

Yöntem şu şekilde açıklanabilir :

1. **Sağa = n-1** olsun
2. Satır için

$a_0, a_1, \ldots, a_{right}$

herhangi iki bitişik öğeyi tutarlı bir şekilde karşılaştırın a_i ve a_{i+1}

Eğer $a_i > a_{i+1}$, her iki öğe de yerlerini değiştirir ve değişimin *i* konumu *k* değişkenine kaydedilir.

KABARCIK SIRALAMA YÖNTEMI

$a_i \leq a_{i+1}$ ise iki element yer değiştirmez.

İşlem sıra sonuna kadar devam eder. Bu taramada i ve i+1 konumundaki elemanların değişiminden sonra başka bir değişim yapılmazsa k+1, k+2, …, n-1 konumlarındaki elemanlar uygun şekilde düzenlenir.

3. **sağ = k**
4. **Sağ > 0** ise s'den itibaren eylemler. 2 ve s. 3 tekrar.
5. **Sağ = 0** ise satır sıralanır.

Kabarcık sıralama yöntemi

Function implementing sorting by the bubble method of the C++ language

```
void BubbleSort(int a[], int n)
{
  int right = n-1;
  int temp = 0;
  while(right > 0)
  {
    int k = 0;
    for(int i = 0; i < right; i ++)
      if(a[i+1] < a[i])
      {
        temp = a[i+1];
        a[i+1] = a[i];
        a[i] = temp;
        k = i;
      }
    right = k;
```

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

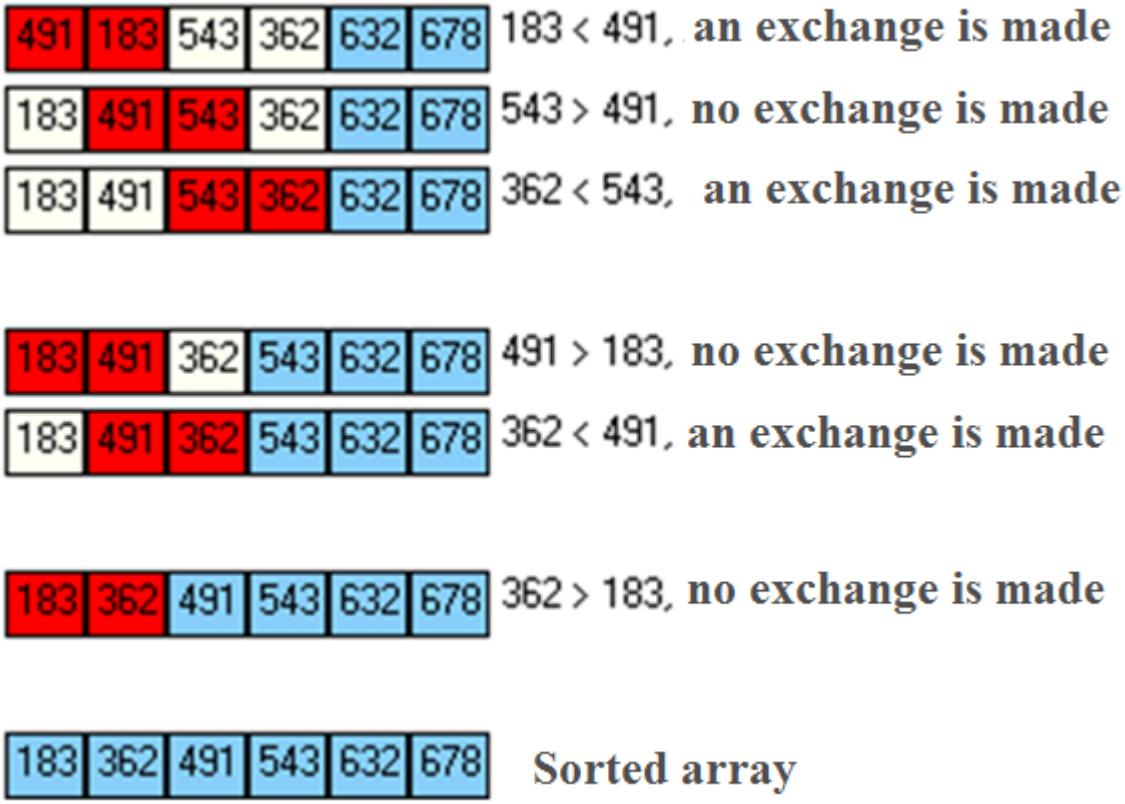
Co-funded by the Erasmus+ Programme of the European Union

Kabarcık sıralama yöntemi

EXAMPLE: Bubble sort method

491	543	632	183	678	362	**Initial array**
491	543	632	183	678	362	543 > 491, **no exchange is made**
491	543	632	183	678	362	632 > 543, **no exchange is made**
491	543	632	183	678	362	183 < 632, **an exchange is made**
491	543	183	632	678	362	678 > 632, **no exchange is made**
491	543	183	632	678	362	362 < 678, **an exchange is made**
491	543	183	632	362	678	543 > 491, **no exchange is made**
491	543	183	632	362	678	183 < 543, **an exchange is made**
491	183	543	632	362	678	632 > 543, **no exchange is made**
491	183	543	632	362	678	362 < 632, **an exchange is made**

Kabarcık sıralama yöntemi



Name of the algorithm	*Average case time complexity*	*Worst case time complexity*	*Stable?*
Bubble sort	$\Theta(n^2)$	$O(n^2)$	Yes
Selection sort	$\Theta(n^2)$	$O(n^2)$	No
Insertion sort	$\Theta(n^2)$	$O(n^2)$	Yes

Seçimli sıralama, minimum öğeyi bularak ve ardından bu minimum öğenin konumundaki öğeyle yer değiştirerek onu doğru konumuna yerleştirerek çalışır. Bu onu istikrarsız hale getirir.

Algorithm	Time Complexity (Best)	Time Complexity (Average)	Time Complexity (Worst)	Space Complexity
Bubble Sort	O(n)	O(n^2)	O(n^2)	O(1)
Insertion Sort	O(n)	O(n^2)	O(n^2)	O(1)
Selection Sort	O(n^2)	O(n^2)	O(n^2)	O(1)

Şu ana kadar tartışılan sıralama seçimleri yalnızca küçük girilen veri sonuçlarında başarılı olduğu için, büyük veri hacmiyle birleştirme sıralama, yığın sıralama veya hızlı sıralama gibi tipik sıralama yöntemlerine özgü O(nlog2 n) karmaşıklığına sahip algoritmaların kullanılması gerekir.

Co-funded by the Erasmus+ Programme of the European Union

Şu ana kadar incelenen sıralama algoritmaları yalnızca küçük boyutlu girdi verisinde, büyük veri hacminde etkili olduğundan, birleştirme sıralama gibi sıralama yöntemlerinin karmaşıklığı $O(n.log_2 n)$ karakteristiğine sahip algoritmaların kullanılması gerekir, yığın veya hızlı sıralama.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Sıralama algoritmaları neden önemlidir?
2. Tartışılan üç sıralama yönteminden hangisi kararsızdır ve neden?
3. Kabarcık sıralamasının nasıl çalıştığını açıklayın?
4. Ekleme sıralamasının nasıl çalıştığını açıklayın?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

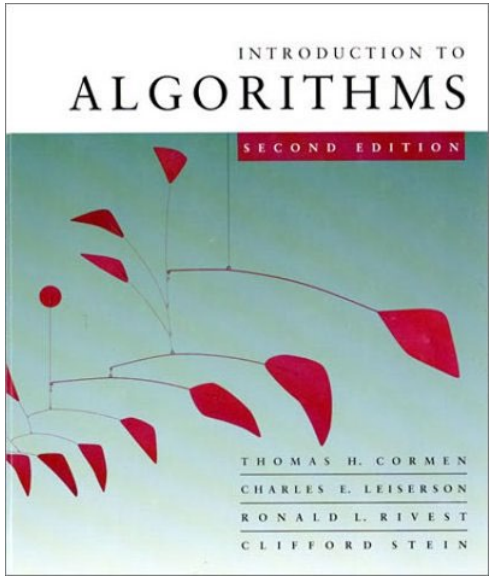
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

- **Modül 3. Sorgu optimizasyonu için algoritmalar ve veritabanlarındaki uygulamaları**
  - **Konu 3. Sıralama algoritmaları**
    - Ders 2. Sıralama algoritmaları - bölüm II (Doğrusal zamanlı sıralama - Hızlı sıralama. Birleştirme sıralama. Yığın sıralama.)

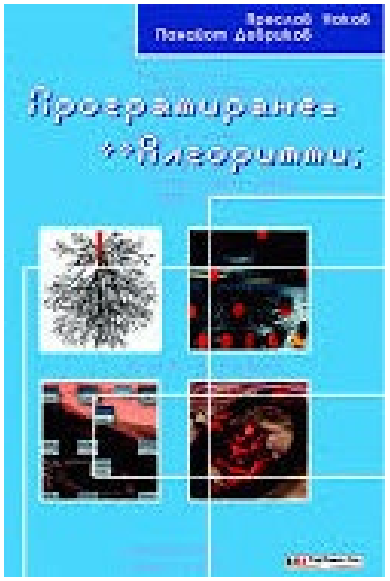
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

SIRALAMA BÖLÜM II Doğrusal zaman sıralaması. Hızlı sıralama. Sıralamayı birleştir. Yığın sıralaması.



Cormen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2009) [1990]. Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill.



Preslav Nakov, Panayot Dobrikov (2002). Programming= ++Algorithms). Top Team Co, Sofia.

Sıralama

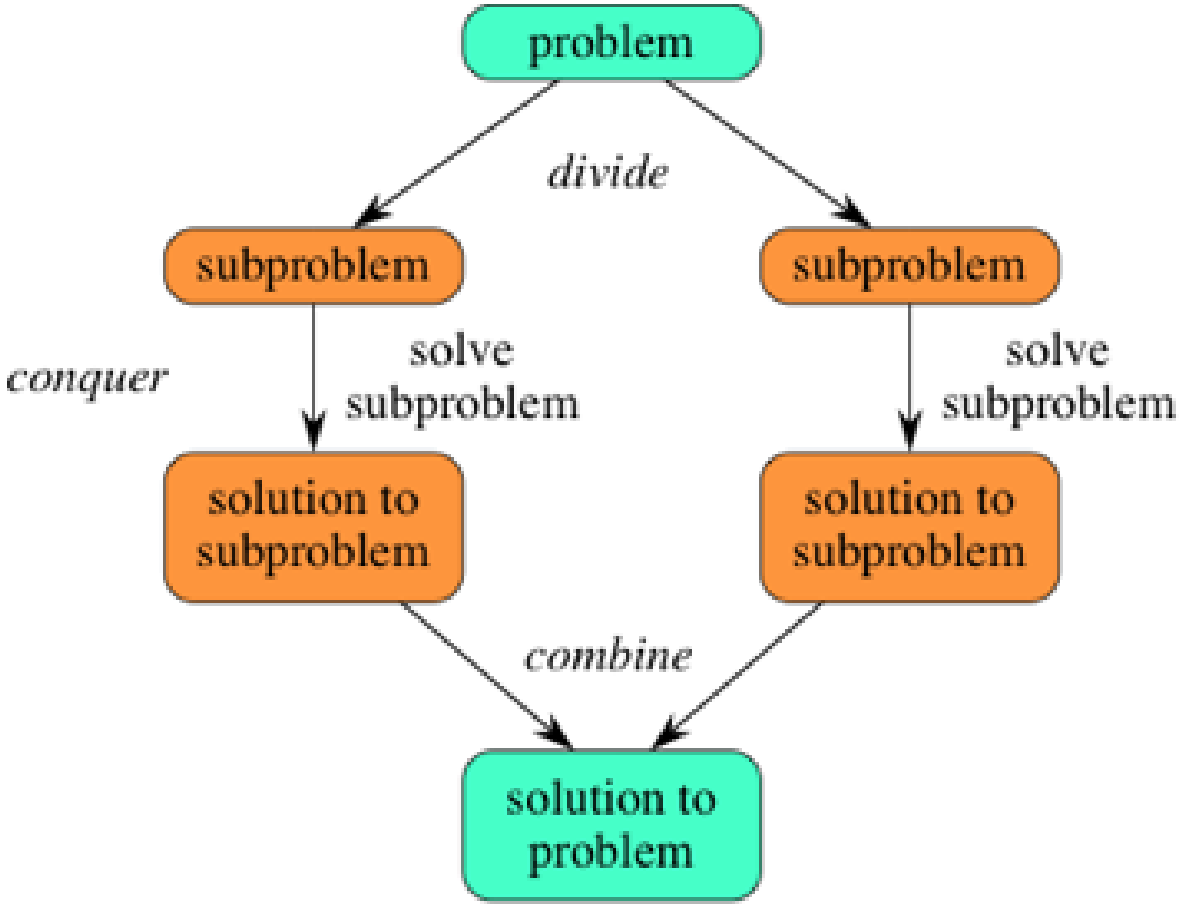
- Önceki derste karşılaştırmaya göre sıralamaya dayalı üç algoritmayı tanıtmıştık: eklemeyle, seçmeyle ve kabarcık yöntemiyle.
- Bu algoritmalar en kötü durumda $O(n^2)$ karmaşıklığına sahiptir, bazıları da ortalama durumda.
- Ancak bu algoritmalar **küçük girdi boyutları için hızlıdır.**

Hızlı sıralama

- C.A.R. tarafından önerildi. 1962 yılında Hoare
- **Böl ve yönet algoritması**.
- "Yerinde" sıralama (ekleme sıralaması gibi, ancak birleştirme sıralamasından farklıdır).
- Çok sayıda pratik uygulamaya sahiptir (çeşitli modifikasyonlarla).

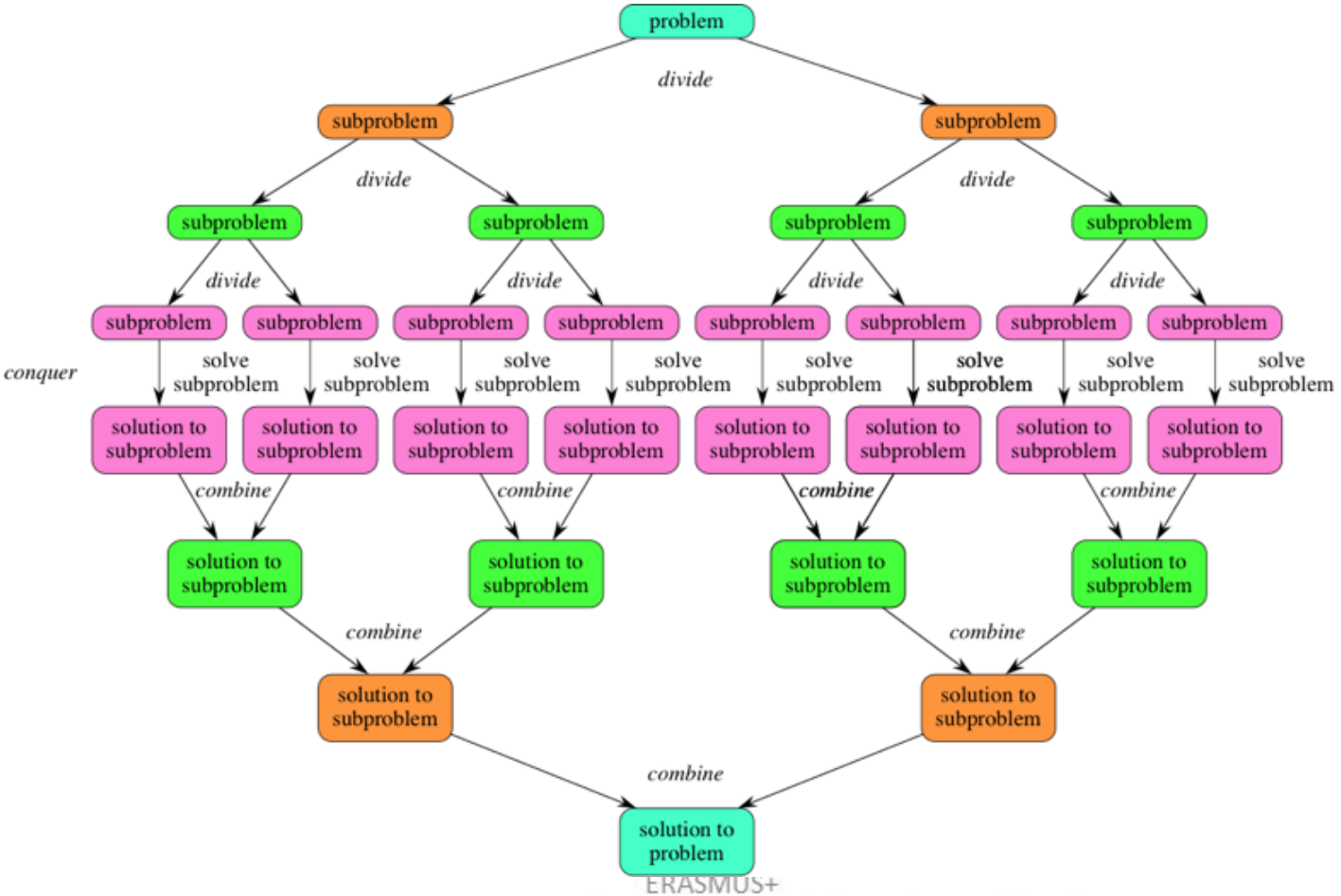
Hızlı sıralama

- **Böl ve fethet algoritması (revizyon).**



Hızlı sıralama

- **Böl ve fethet algoritması (revizyon).**



Hızlı sıralama

- *n* öğeli bir dizinin hızlı sıralaması :

1. **Bölmek :** diziyi x merkezi etrafında iki alt diziye bölün, alt alt dizideki öğeler üst alt dizideki öğelere göre olsun. $\leq x \leq$

2. **Kural:** İki alt dizinin yinelemeli sıralaması.
3. **Birleştir:** önemsiz.
   Anahtar kelime: alt programların bölünmesi için doğrusal zaman

Hızlı sıralama

- Aynı algoritmayı sol ve sağ parçalara da uyguluyoruz, tek bir öğe içeren aralıklara ulaşana kadar dikkate alınan alt dizilerin sol ve sağ sınırlarını kademeli olarak azaltıyoruz.
- Algoritma işini bitirdikten sonra dizi sıralanacaktır.
- *En kötü durumda, algoritma $O(n^2)$ karmaşıklığına sahiptir.*
- *Ortalama durumda, algoritmanın karmaşıklığı* ***O(nlog n)'dir.***

Örneğin **n=100**, n^2=10 000, log n=6.64, n*log n=664

Co-funded by the Erasmus+ Programme of the European Union

Hızlı sıralama

Eylem sorumlusu :

1. Sıralanacak öğeler listesinden bir "**pivot**" öğesi seçin.
2. Liste, "**pivot**" öğesinden daha küçük olan tüm öğeler onun soluna, daha büyük olan tüm öğeler ise sağına yerleştirilecek şekilde yeniden düzenlenir.
3. Daha küçük öğelerin bulunduğu listede ve daha büyük öğelerin bulunduğu listede yukarıdaki adımları yinelemeli olarak tekrarlayın.
4. Ortaya çıkan listeler, sıralanmış listeyi oluşturmak için birleştirilir (birleştirme).

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Hızlı sıralama

Örnek 1:

[5 3 2 8 7 6 1 9 4]

For a "pivot"element choose 5 **. Divide the list into three parts:**

elements smaller than "pivot", "pivot", bigger than "pivot"

[3 2 1 4] | [5] | [8 7 6 9]

Recursively perform the actions on the newly obtained list

[2 1] | [3] | [4] | [5] | [7 6] | [8] | [9]

[1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9]

Link the lists and obtain the sorted list

[1 2 3 4 5 6 7 8 9]

Hızlı sıralama

Basitleştirilmiş bir sözde kod kullanarak algoritmanın uygulanmasını görebiliriz:

```
1   function quicksort('array')
2       if length('array') ≤ 1
3           return 'array'  // an array of zero or one elements is already
sorted
4       select and remove a pivot element 'pivot' from 'array'  //
see selection of "pivot"element below
5       create empty lists 'less' and 'greater'
6       for each 'x' in 'array'
7           if 'x' ≤ 'pivot' then append 'x' to 'less'
8           else append 'x' to 'greater'
9       return concatenate(quicksort('less'), list('pivot'),
quicksort('greater'))  // two recursive requirements
```

Hızlı sıralama

- Elementlerin ancak diğer elementlerle karşılaştırılarak kontrol edildiğini görebiliriz. Bu, hızlı sıralamayı karşılaştırmaya göre sıralama olarak tanımlar.
- Algoritmanın doğruluğu aşağıdaki iki argümana dayanmaktadır:

- her yinelemede, önceden işlenmiş öğeler istenen konuma yerleştirilir: kendisinden küçüklerse "pivot" öğesinden önce ve ondan büyükse "pivot" öğesinden sonra;

- Her yineleme, henüz birer birer sıralanmayan öğelerin sayısını azaltır..

Tüm algoritmanın doğruluğu tümevarımla kanıtlanır: sıfır veya bir öğe için algoritma verileri değiştirmeden bırakır; Daha büyük bir veri kümesi için algoritma iki parçayı birleştirir: "pivot"tan daha küçük öğeler ve ondan daha büyük öğeler.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

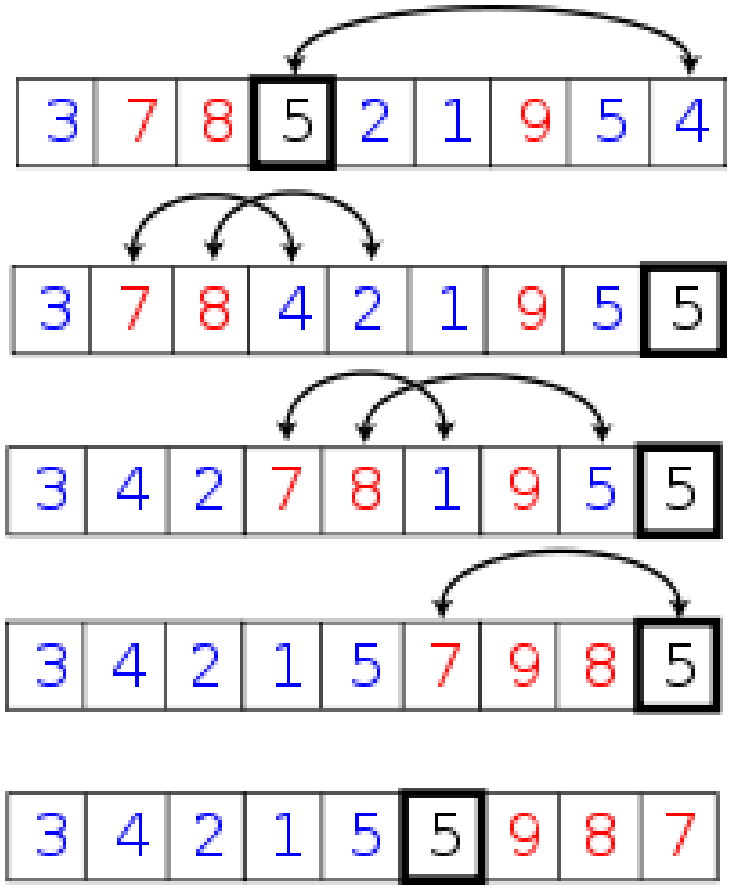
Hızlı sıralama

Hızlı Sıralama için etkileşimli örnek (Örnek 2):

6 5 3 1 8 7 2 4

Hızlı sıralama

Hızlı Sıralama için Örnek 3:



İşaretli öğe pivot öğedir, mavi öğeler daha küçük veya eşittir ve kırmızı öğeler daha büyüktür.

Hızlı sıralama

Pivot elemanının seçimi :

- Hızlı sıralamanın ilk sürümlerinde, en soldaki öğe genellikle **pivot** olarak seçiliyordu.
- Ne yazık ki bu, oldukça yaygın bir durum olan zaten sıralanmış diziler için işe yaramaz.
- Sorun, pivot elemanı için keyfi bir indeks veya bölümün orta elemanı veya (özellikle daha uzun seriler için) bölümün ilk, orta ve son elemanlarının medyanı (Robert Sedgewick tarafından önerildiği gibi) seçilerek kolayca çözüldü.).

Co-funded by the Erasmus+ Programme of the European Union

Hızlı sıralama

- Hızlı sıralamada, öğeler belirli bir ağaç yapısına sırayla eklenmez, ancak aynı anda özyinelemeli çağrıda elde edilen bir ağaç oluşturur.
- Hızlı sıralamanın en yakın rakibi yığın sıralamadır.
- Burada en kötü zaman her zaman O(log n)'dir, ancak yığın sıralamanın genellikle standart hızlı sıralamadan daha yavaş olduğu kabul edilir.
- Birleştirme sıralaması *O(log n).* en kötü durum süresi *O(nlog n)* olan, hızlı sıralamayla karşılaştırılabilecek başka bir özyinelemeli algoritmadır.

Birleştirme sıralaması

- Birleştirmenin her adımında a'nın ilk yarısındaki bir elemanı a'nın ikinci yarısındaki bir elemanla karşılaştırıp b'ye bir eleman yazıyoruz, dizilerin elemanlarına 3 çağrı yapıyoruz.
- Hızlı yerinde sıralama ve yığın sıralamanın aksine kararlı bir algoritmadır ve bağlantılı listelerde ve diskler veya ağ depolama gibi yavaş kartuşlarda depolanan çok büyük listelerde kullanılmak üzere kolayca uyarlanabilir.
- Birleştirme sıralamasının ana dezavantajı, ek *O(n)* dizi işlem alanına ihtiyaç duymasıdır; hızlı yerinde sıralama ve kuyruk özyinelemesinin temel değişkeni ise yalnızca *O(log n)* kullanır.

Birleştirme sıralaması

- Algoritma “böl ve yönet” ilkesine dayanmaktadır.
- Eylem ilkesi :

1. Sıralanmamış liste, yaklaşık toplam uzunlukta (doğrusal zaman için) rastgele iki alt listeye bölünür;
2. Tek uzunluktaki listelere ulaşılana kadar alt listeleri tekrar tekrar bölün;
3. İki alt listeyi yeni bir sıralı listede birleştirin (doğrusal zaman için).
4. Birleştirme sıralamasının avantajı her zaman *n*log(n)* karmaşıklığıyla çalışmasıdır.

Co-funded by the Erasmus+ Programme of the European Union

Birleştirme sıralaması

- Bu sıralama algoritmasının temeli "**böl ve yönet**" yöntemidir: sıralanan dizi, her biri sıralanan iki parçaya bölünür ve ardından sıralanan alt diziler birleştirilir.
- Birleştirme sıralama algoritması bilinen en *etkili* sıralama algoritmalarından biridir.
- Algoritma paralel olarak kullanılabilir, yani. e. işlemcinin her çekirdeğinde diziyi 2'ye bölen yöntem beslenir ve ardından sayılar tek seferde birleştirilir.
- Eğer 8 adet mikroişlemcimiz varsa, her işlemciye Split yöntemi uygulanıyor ve ardından 8 parçanın tamamı aynı anda birleşiyor.
- *n* işlemcimiz olduğunda algoritma *n* kat daha hızlı hale gelebilir

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Birleştirme sıralaması

Source code of the merge sort algorithm of C++ language

```
// A procedure that implements merging of two arrays into a third one
void merge(const int a[], int n, const int b[], int m, int* c)
{
  int i = 0, j = 0, k = -1;

  while(i < n && j < m)
    if(a[i] < b[j])
    {
      k ++;
      c[k] = a[i];
      i ++;
    }
    else
    {
      k ++;
      c[k] = b[j];
      j ++;
    }

  if(i == n)
    while(j < m)
    {
      k ++;
      c[k] = b[j];
      j ++;
    }
  else
    while(i < n)
    {
      k ++;
      c[k] = a[i];
      i ++;
    }
}
```

Co-funded by the Erasmus+ Programme of the European Union

Birleştirme sıralaması

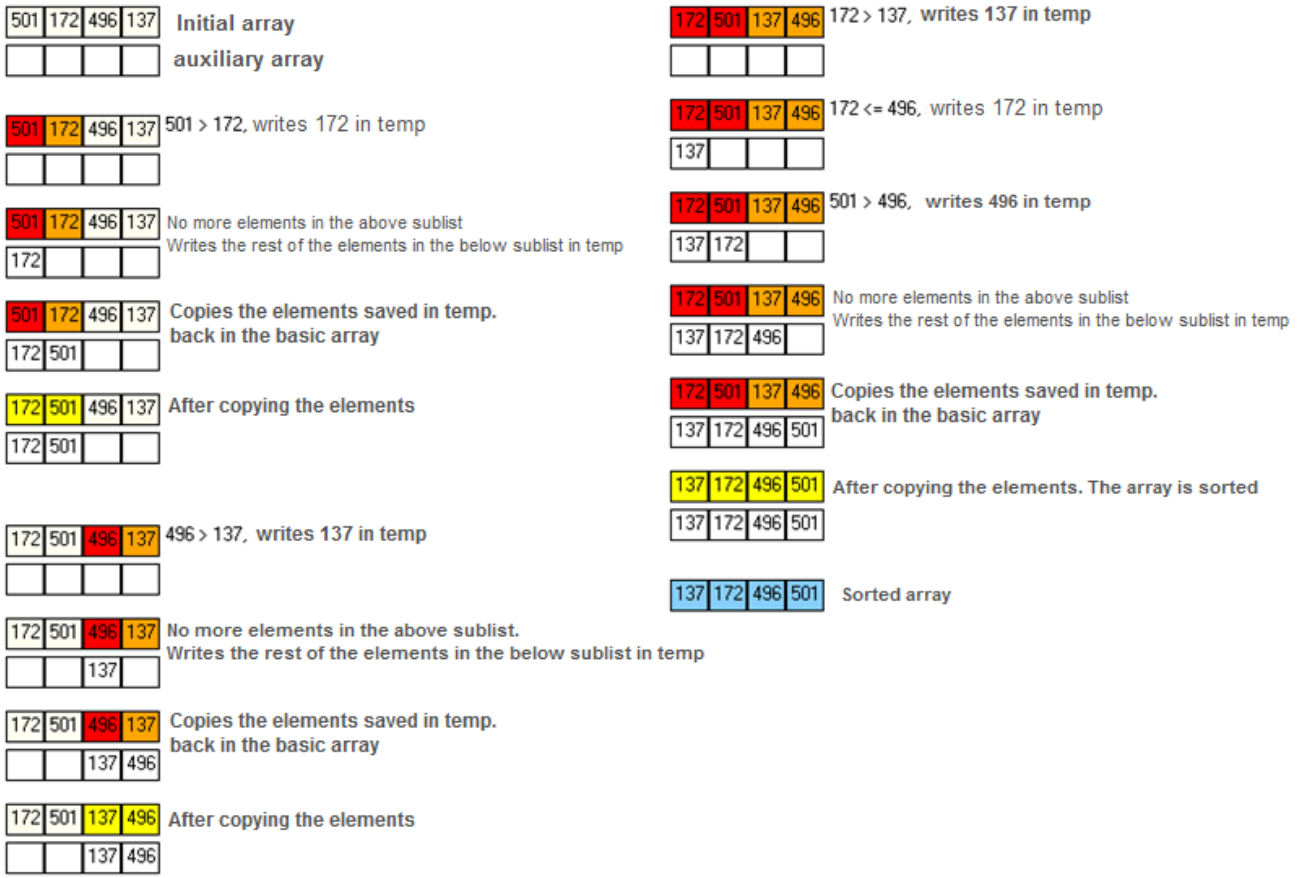
```
// The procedure that solves the merge sort task
void MergeSort(int a[], int n)
{
  if(n < 2)
    return;

  int nLeft = n / 2;
  int nRight = n - nLeft;

  MergeSort(a, nLeft);
  MergeSort(a + nLeft, nRight);

  int* p = new int[n];
  merge(a, nLeft, a + nLeft, nRight, p);

  for(int i = 0; i < n; i ++)
    a[i] = p[i];

  delete [] p;
}
```

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Birleştirme sıralaması

Örnek: Birleştirme sıralaması.



Co-funded by the Erasmus+ Programme of the European Union

Birleştirme sıralaması

Birleştirme sıralamasının etkileşimli bir örneği.

6 5 3 1 8 7 2 4

Yığın sıralaması

- Yığın sıralaması, doğrudan seçim yoluyla bir tür sıralama algoritmasıdır.
- Hızlı sıralama algoritmasının iyi bir uygulamasına göre O(n log n) karmaşıklığıyla daha yavaş olmasına rağmen, çok fazla dizi veya özyineleme kullanmadığından daha uygun bir en kötü durum avantajına sahiptir.
- Yığın sıralaması 1964 yılında J. W. J. Williams tarafından icat edildi.
- Yığın sıralaması iki adıma ayrılabilir :
  - İlk adım, kümenin elemanlarından bir "yığın" oluşturmak;
  - İkinci adım, piramidal veri yapısından (yığın) en büyük öğenin alınıp çıkarılması ve başka bir diziye yerleştirilmesiyle sıralanmış bir dizi oluşturulur.

Yığın sıralaması

- "Yığın" her çıkarma işleminde yeniden oluşturulur ve tüm nesneler yığın belleğinden kaldırıldıktan sonra tam sıralanmış dizi oluşturulur.
- İlk adımda minimum yığın veya maksimum yığın seçilerek sıralanan öğelerin yönü değiştirilebilir.
- Bu algoritmayı yürütmek için iki dizi gereklidir; biri yığındaki öğeler için, ikincisi sıralanan dizi için.
- Yığın sıralaması tek bir diziyle de yapılabilir - bir öğe "yığından" çıkarıldığında yer açar ve öğe aynı dizinin sonuna eklenebilir.
- Yığın sıralama esas olarak hızlı sıralama ile rekabet eder. Hızlı sıralamanın en kötü durumu $O(n^2)$ 'dir.

Co-funded by the Erasmus+ Programme of the European Union

Yığın sıralaması

Yığın sıralaması için etkileşimli örnek

6 5 3 1 8 7 2 4

Co-funded by the Erasmus+ Programme of the European Union

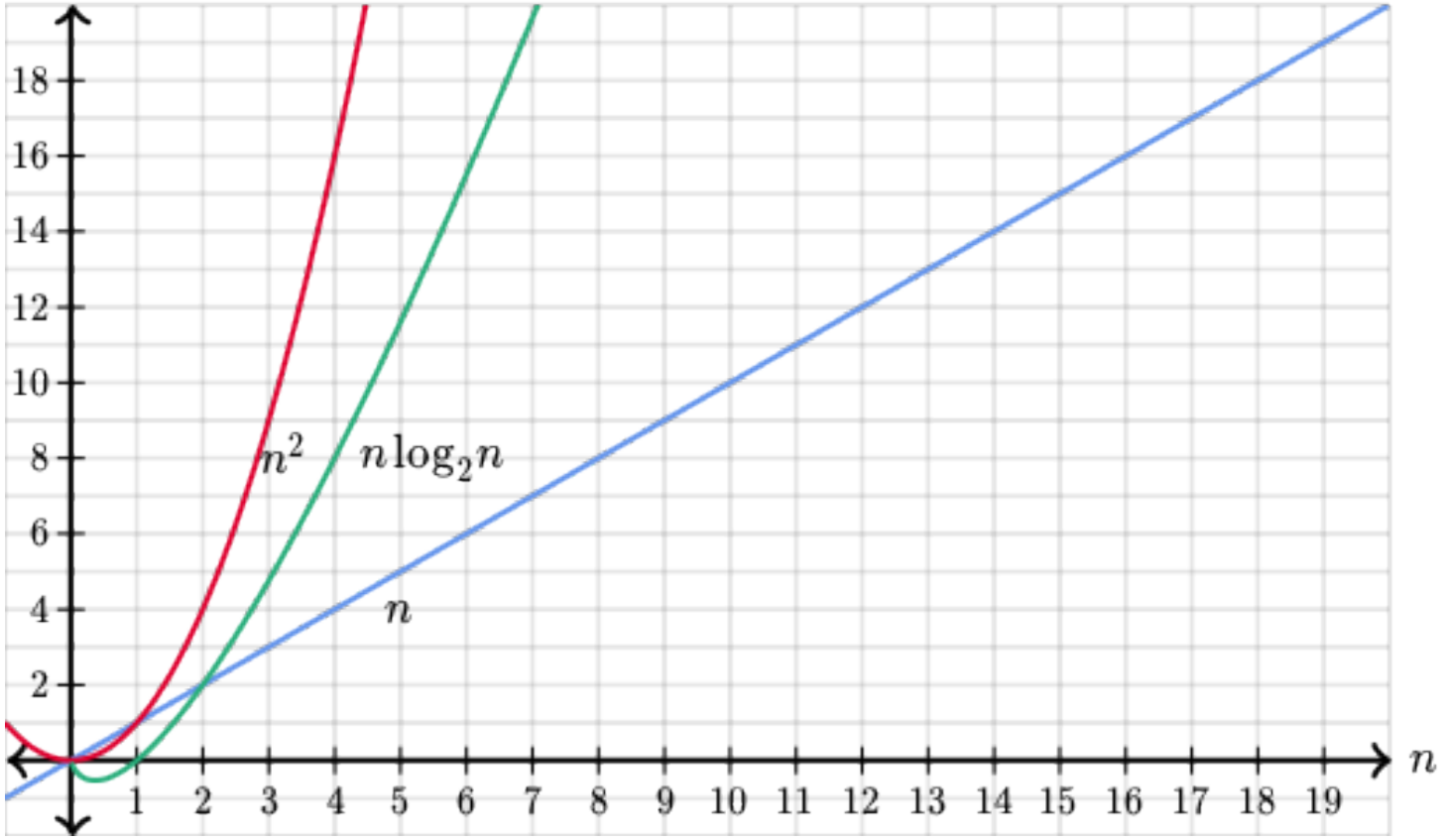
Sıralama

Sıralama ve istatistik.

Algorithm	Worst-case running time	Average-case/expected running time
Insertion sort	$\Theta(n^2)$	$\Theta(n^2)$
Merge sort	$\Theta(n \lg n)$	$\Theta(n \lg n)$
Heapsort	$O(n \lg n)$	—
Quicksort	$\Theta(n^2)$	$\Theta(n \lg n)$ (expected)

Sıralama

Sıralama ve istatistik.



Sıralama - sonuçlar

- Sıralamayı O(nlgn) zamanında gerçekleştiren çeşitli algoritmalar tanıttık.
- Birleştirme sıralaması ve yığın sıralaması en kötü durumda bu üst sınıra ulaşır.
- Hızlı sıralama ortalama durumda bu sınıra ulaşır.
- Ayrıca, bu algoritmaların her biri için, algoritmanın O(nlgn) sürede çalışmasını sağlayan n giriş verisinden oluşan bir dizi elde edilebilir.
- Bu algoritmaların ilginç bir özelliği vardır: Sıralamalarındaki öğelerin sırasının belirlenmesi yalnızca girdi öğeleri arasındaki karşılaştırmalara dayanır.

Sıralama - sonuçlar

- Veritabanındaki harici sıralama, çoğu veri tabanı dosyası gibi, diskte depolanan ve ana belleğe tam olarak sığmayan büyük kayıt dosyaları için uygun olan sıralama algoritmalarını ifade eder.
- Tipik harici sıralama algoritması, ana dosyanın küçük alt dosyalarını (çalışmalar adı verilen) sıralayarak başlayan ve ardından sıralanan işlemleri birleştirerek sırayla birleştirilen daha büyük sıralanmış alt dosyalar oluşturan bir sıralama-birleştirme stratejisi kullanır.
- Sıralama-birleştirme algoritması, diğer veritabanı algoritmaları gibi, ana bellekte, çalıştırmaların gerçek sıralama ve birleştirme işlemlerinin gerçekleştirildiği arabellek alanı gerektirir.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Doğrusal zamanla sıralama nedir?
2. Birleştirme Sıralamasının nasıl çalıştığını açıklayın?
3. Yığın Sıralamanın nasıl çalıştığını açıklayın?
4. Quicksort ne zaman Mergesort'tan daha iyidir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

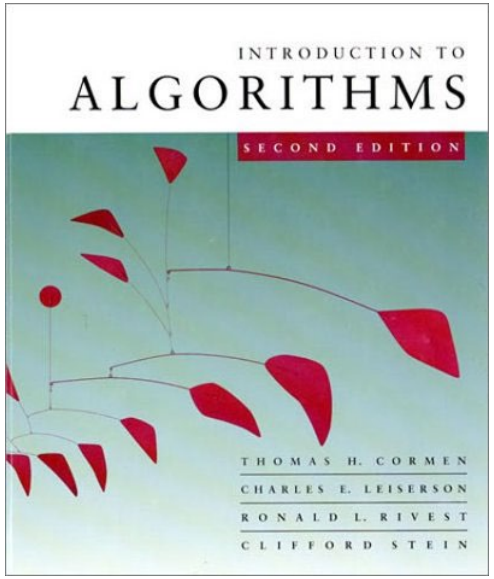
ON-LINE DISTANCE COURSE ON DATABASES

- **Modül 3. Sorgu optimizasyonu için algoritmalar ve veritabanlarındaki uygulamaları**
  - **Konu 4. Grafik algoritmaları**
    - Ders 1. Grafik algoritmalarına giriş.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

GRAFİK ALGORİTMALARINA GİRİŞ



Carmen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2009) [1990]. Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill.

Preslav Nakov, Panayot Dobrikov (2002). Programming= ++Algorithms). Top Team Co, София.

Co-funded by the Erasmus+ Programme of the European Union

Giriş

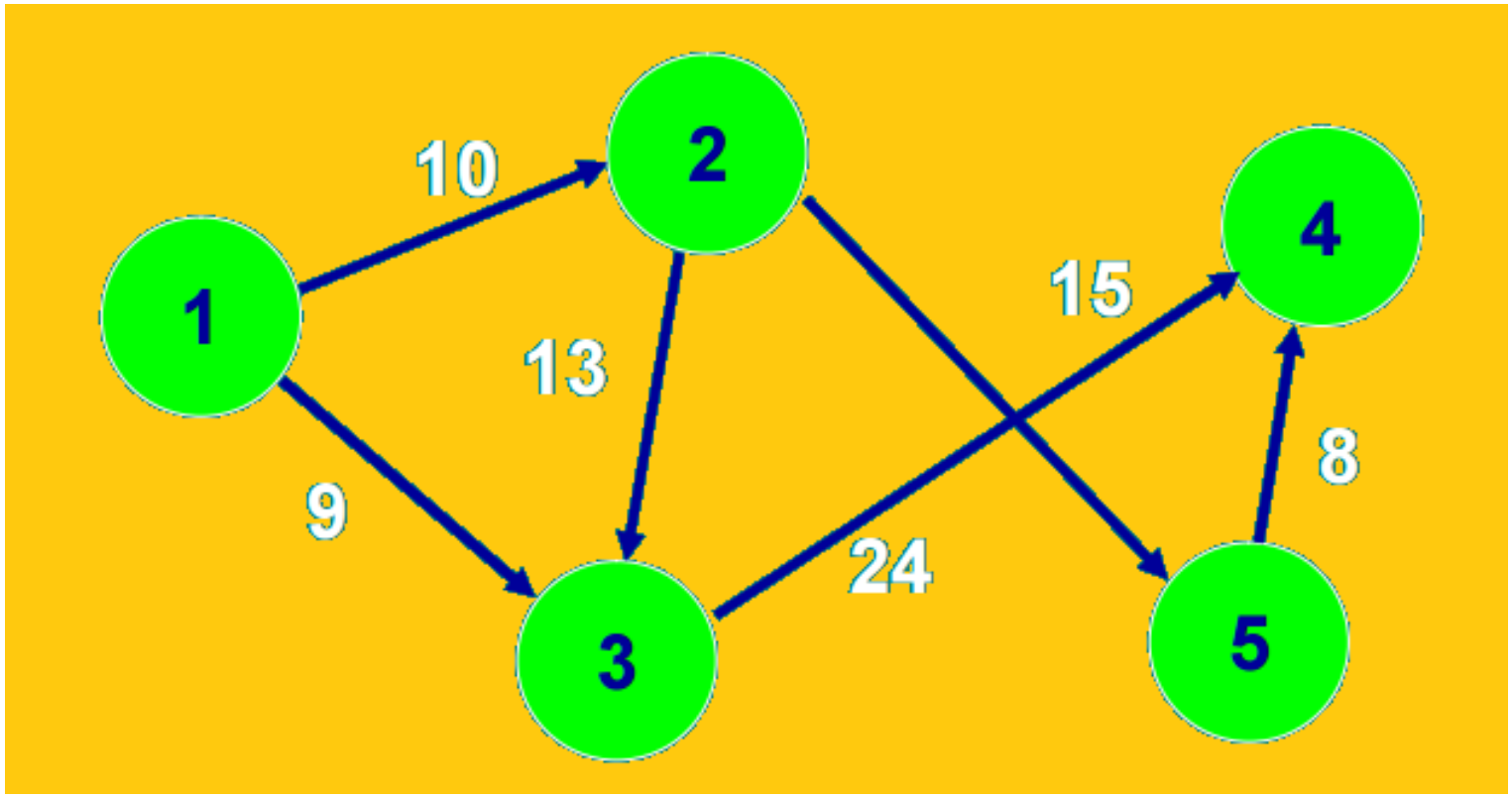
- Bu *derste* grafikleri temsil etme ve grafikleri arama yöntemleri sunulmaktadır.
- Grafik arama teknikleri, grafik algoritmaları alanının temelini oluşturur.
- Bir grafik, bir dizi köşe (düğüm) ve yay (kenar) olarak görüntülenir.
- Bir dizi nesnenin ve bunların ilişkilerinin sunumu kullanılır.
- Genellikle düğümler nesnelere, kenarlar ise aralarındaki bağlantılara karşılık gelir.

Giriş

- Genel olarak konuşursak, bir grafik, kenarlar olarak adlandırılan köşelerden (düğümlerden) ve bunlar arasındaki bağlantılardan oluşan bir veri yapısıdır.
- Kenarların bir yönü varsa grafik yönlüdür, aksi takdirde yönsüzdür.
- Kenarların ayrıca bir ağırlığı (veya uzunluğu) olabilir. Daha sonra grafiğin **ağırlıklı grafik** olduğu söylenir.
- Bağlantılı listeler ve ağaçlar grafiklerin özel durumlarıdır.
- Aşağıda yönlendirilmiş, ağırlıklı bir grafik örneği verilmiştir:

Giriş

- Yönlendirilmiş, ağırlıklı bir grafik:



Co-funded by the Erasmus+ Programme of the European Union

Ve bu yüzden! Grafik Nedir?

- Her birinin sıfır veya daha fazla ardılı ve sıfır veya daha fazla öncülü olabilen öğeler topluluğu

- Ağaçlar ve listeler grafiklerin yalnızca özel durumlarıdır

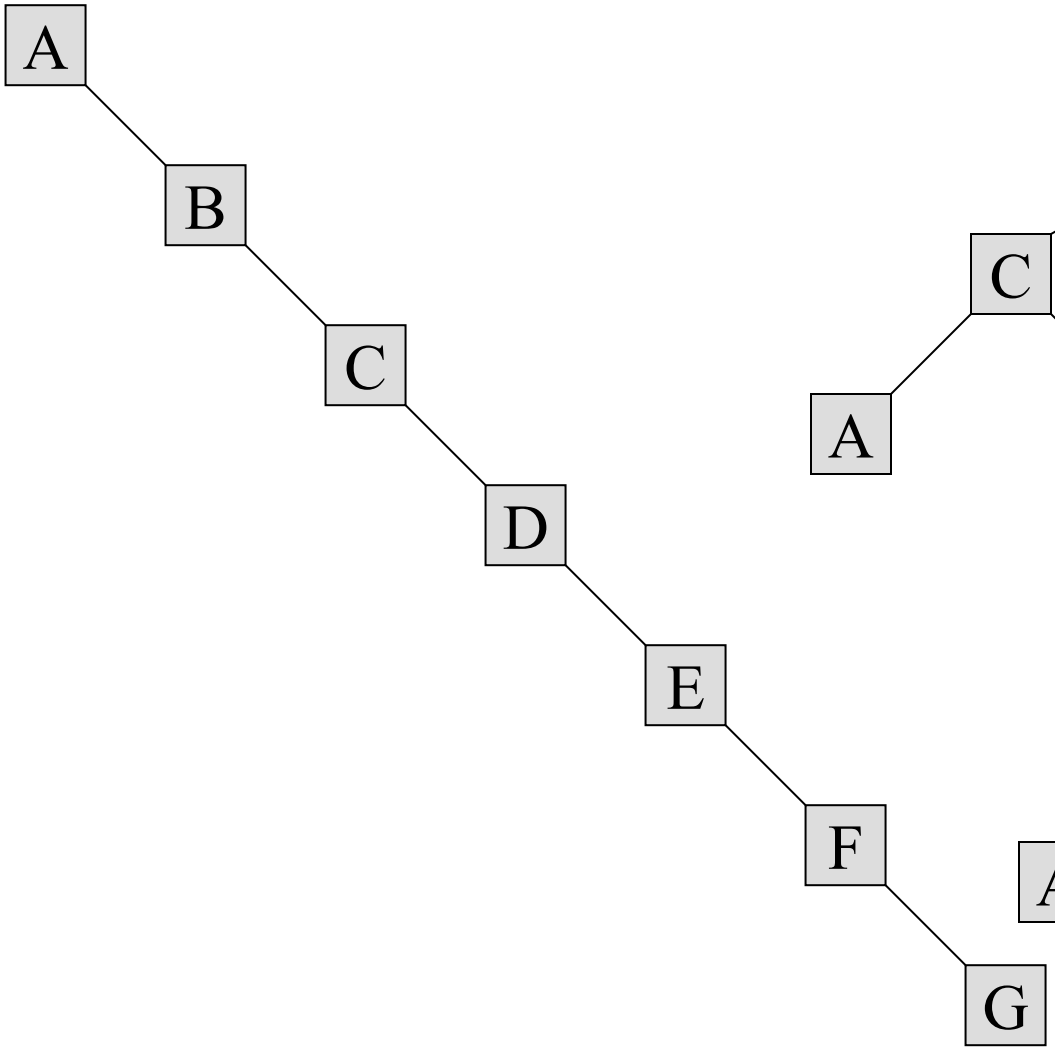
Günlük Yaşamda Grafikler

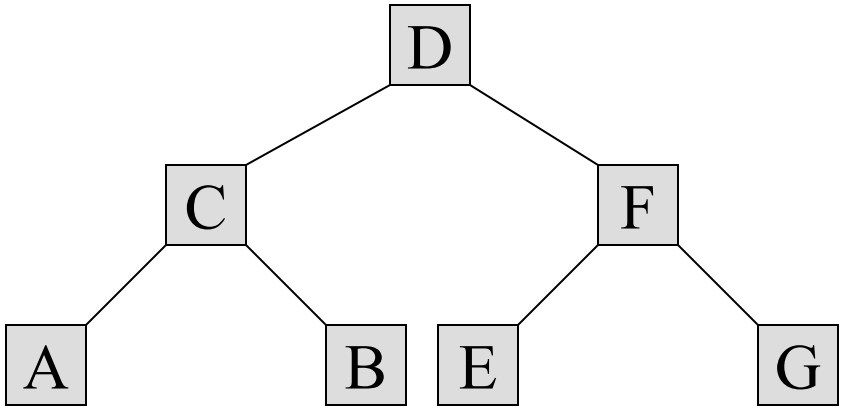
- Bir yol haritası
- Havayolu güzergahlarının haritası
- Web sayfaları arasındaki bağlantılar
- Sosyal ağdaki ilişkiler
- Bir iletişim veya ulaşım ağındaki akış kapasitelerinin şeması

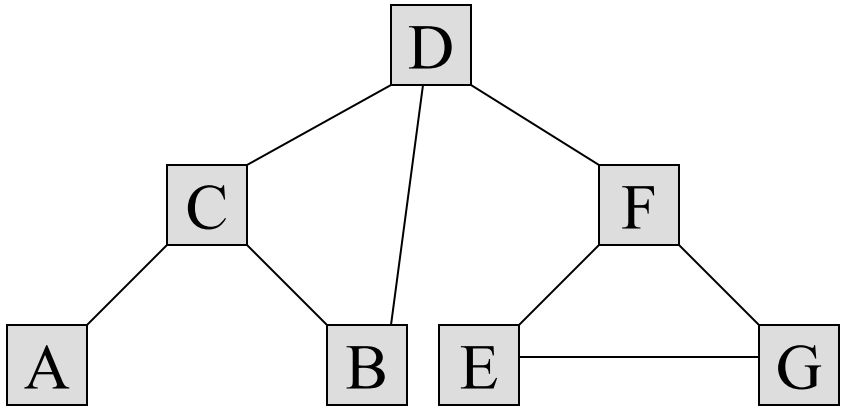
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

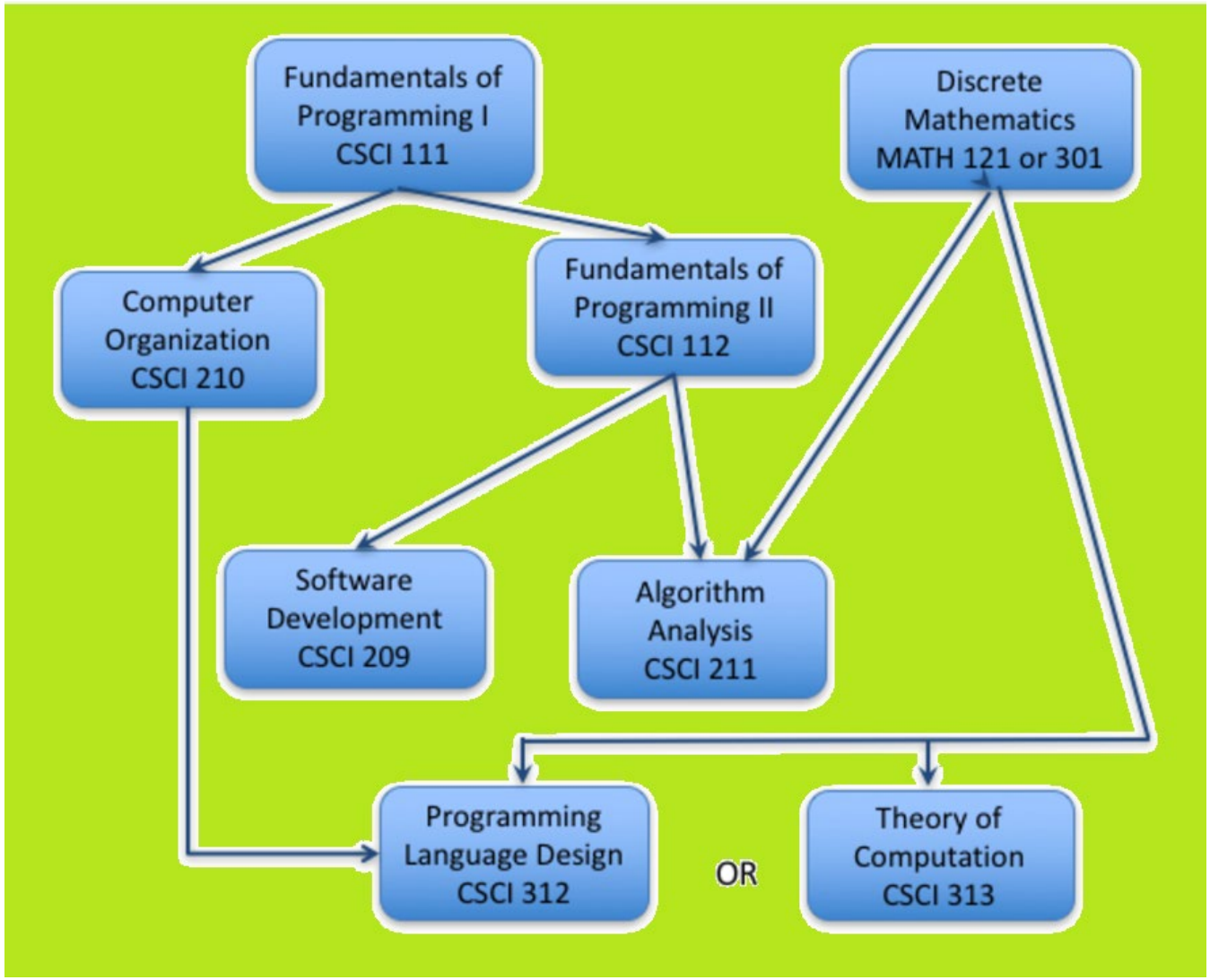
Örnekler







Co-funded by the Erasmus+ Programme of the European Union

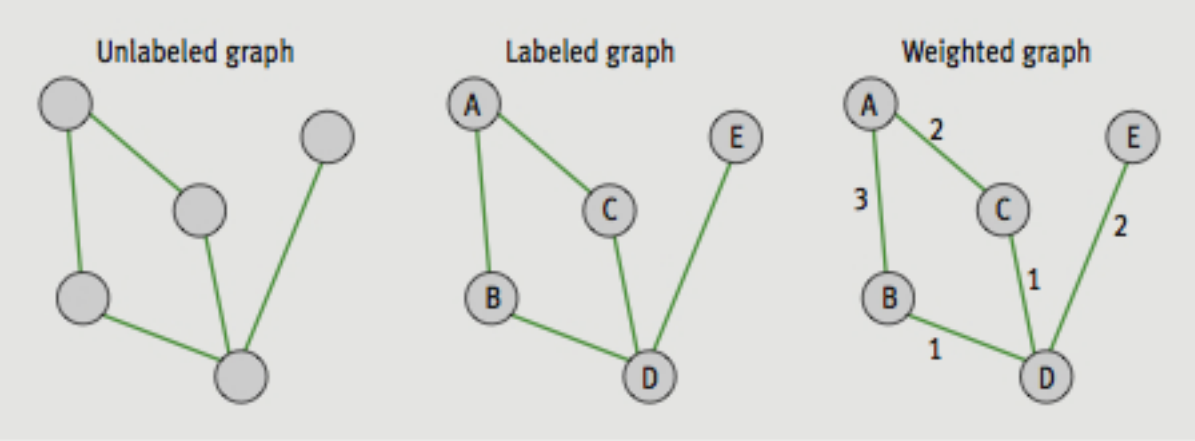


Tepe Noktası, Kenar, Etiket ve Ağırlık

- Bir grafikteki düğümlere köşeler de denir
- Köşeler arasındaki bağlantılara kenarlar denir
- Köşeler ve kenarlar etiketlenebilir veya etiketlenmemiş olabilir
- Sayısal kenar etiketine ağırlık da denir

Co-funded by the Erasmus+ Programme of the European Union

Örnekler



Co-funded by the Erasmus+ Programme of the European Union

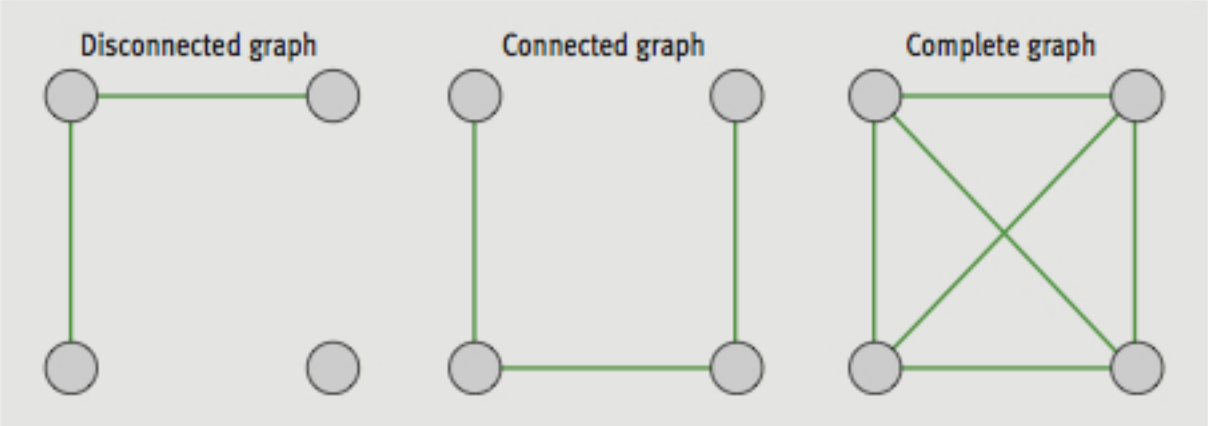
Bağlantılar

- Her köşeden başka bir köşeye en az bir kenar varsa bir grafik bağlantılıdır

- Her köşeden diğer köşelere bir kenar varsa grafik tamamlanmış demektir

Co-funded by the Erasmus+ Programme of the European Union

Örnekler

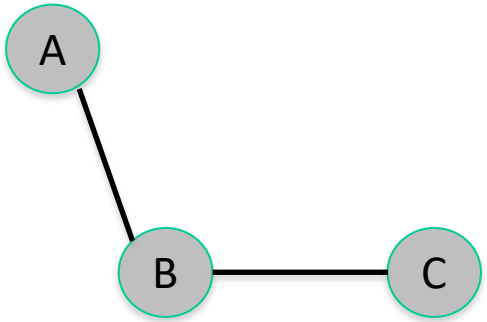


Yollar ve Döngüler

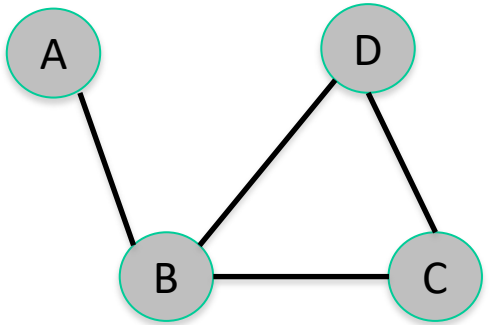
- Yol, bir tepe noktasına başka bir tepe noktasından ulaşılmasına izin veren bir kenar dizisidir
- Basit yol, bir köşenin birden fazla ziyaret edilmediği yoldur.
- Döngü, bir tepe noktasının birden fazla ziyaret edildiği bir yoldur

Örnekler

Basit yol: ABC



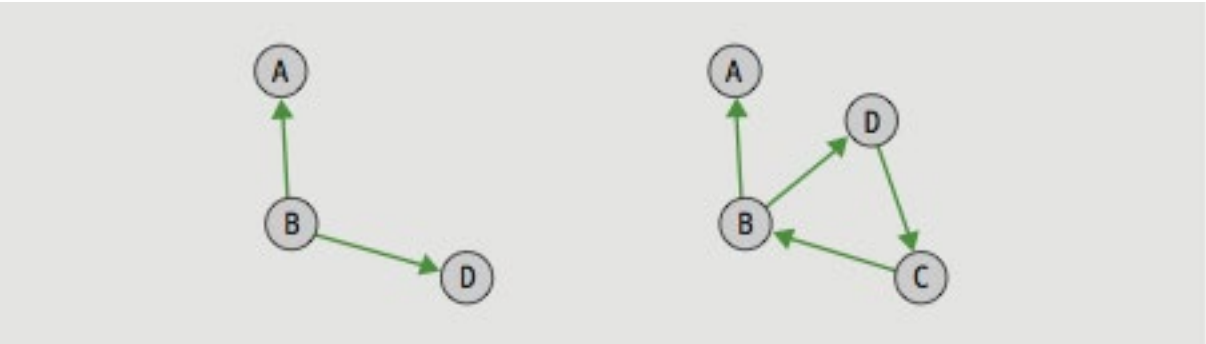
Döngüsü: BCD



Yönlendirilmiş Grafikler (Digraflar)

- Yönlendirilmiş bir grafikteki her kenar bir yönü işaret eder ve kaynak tepe noktasından hedef tepe noktasına doğru harekete izin verir

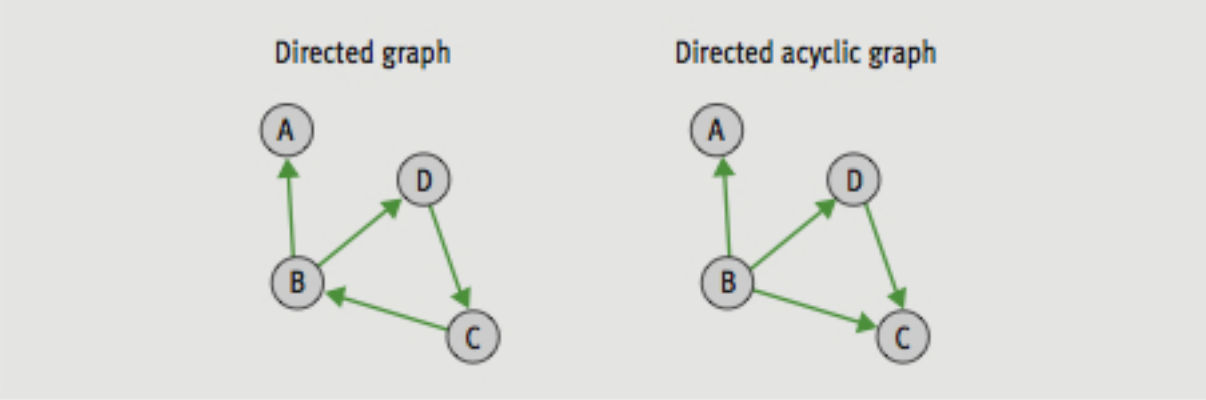
- Bu kenarlara yönlendirilmiş kenarlar denir



Co-funded by the Erasmus+ Programme of the European Union

Yönlendirilmiş Döngüsel Olmayan Grafikler (DAG'ler)

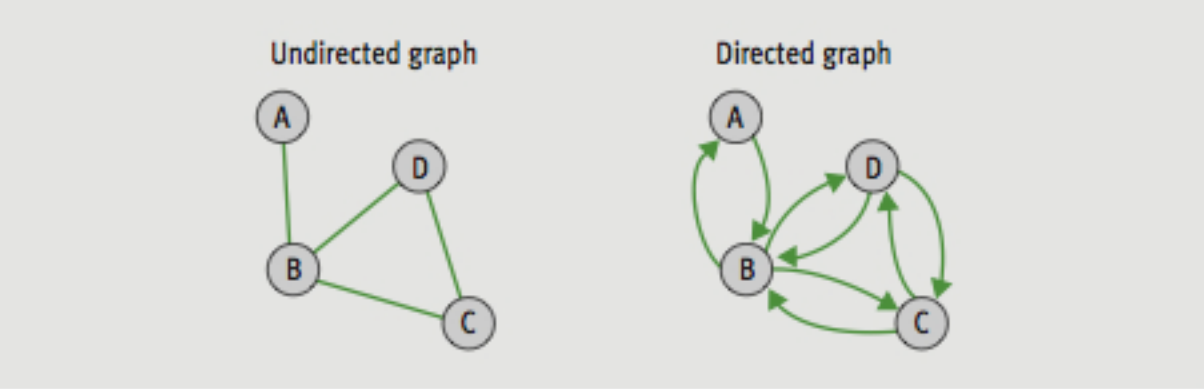
- Yönlendirilmiş bir asiklik grafik, döngüsü olmayan yönlendirilmiş bir grafiktir



Co-funded by the Erasmus+ Programme of the European Union

Yönlendirilmiş ve Yönsüz Grafikler

- Yönlendirilmemiş bir grafikteki kenarlar her iki yönde de hareketi destekler



Co-funded by the Erasmus+ Programme of the European Union

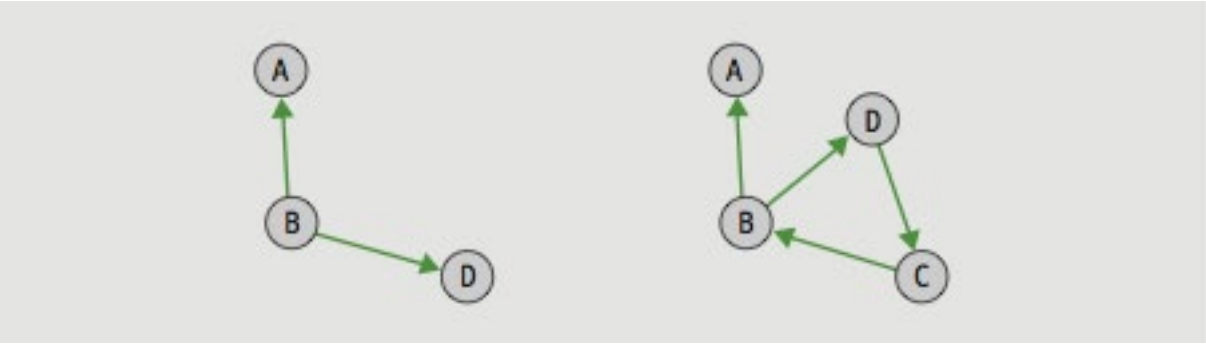
Seyrek ve Seyrek Olmayan Grafikler

- Seyrek grafikler, minimum sayıda kenara (kabaca *N* kenar) sahip bağlantılı grafiklerdir.

- Seyrek olmayan grafikler maksimum kenar sayısına yakındır (kabaca N^2 kenar)

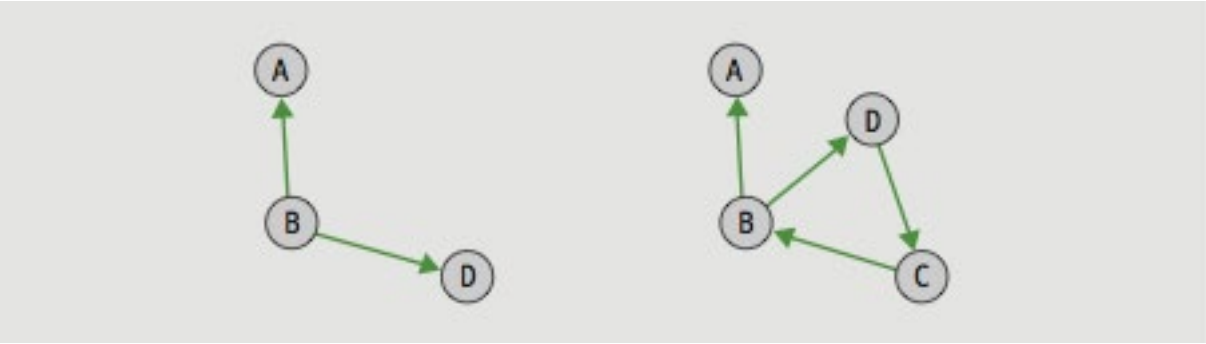
Bitişik Tepe Noktaları

- B'den A'ya yönlendirilmiş bir kenar varsa, A köşesi B köşesine bitişiktir



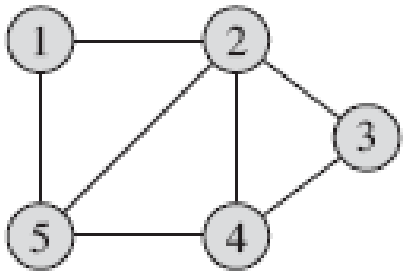
Olay Kenarları

- BA kenarı, B'den A'ya yönlendirilmiş bir kenar ise, B köşesine tesadüf eder.



Grafiklerin sunumu

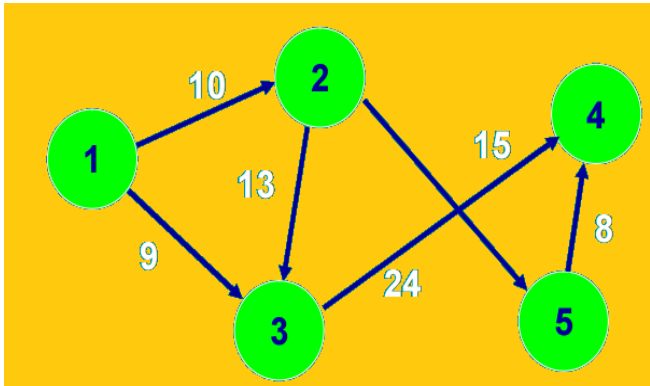
- **Grafiklerin sunumu** : Bir grafiği temsil etmenin olası bir yolu komşuluk matrisidir. N satır ve n düğümden oluşur; burada n, grafikteki düğüm sayısıdır. Her satır ve sütun belirli bir düğüme karşılık gelir. Düğüm 1 ile düğüm 2 arasında bir kenar varsa, o zaman [1] [2] konumunun elemanı 1'dir ve kenar yoksa - 0'dır.



	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Grafiklerin sunumu

- **Grafiklerin sunumu** : Bir grafiği sunmanın başka bir olası yolu da insidans matrisidir. Burada satırlar düğümleri, sütunlar ise kenarları temsil ediyor. *[i] [j]* konumunun elemanı, *j* kenarı *i* düğümünden çıkarsa -1, *j* kenarı *i* düğümünden gelirse 1, aksi takdirde 0'dır.



-1	-1	0	0	0	0
1	0	-1	-1	0	0
0	1	1	0	-1	0
0	0	0	0	1	1
0	0	0	1	0	-1

0	10	9	0	0
0	0	13	0	15
0	0	0	24	0
0	0	0	0	0
0	0	0	8	0

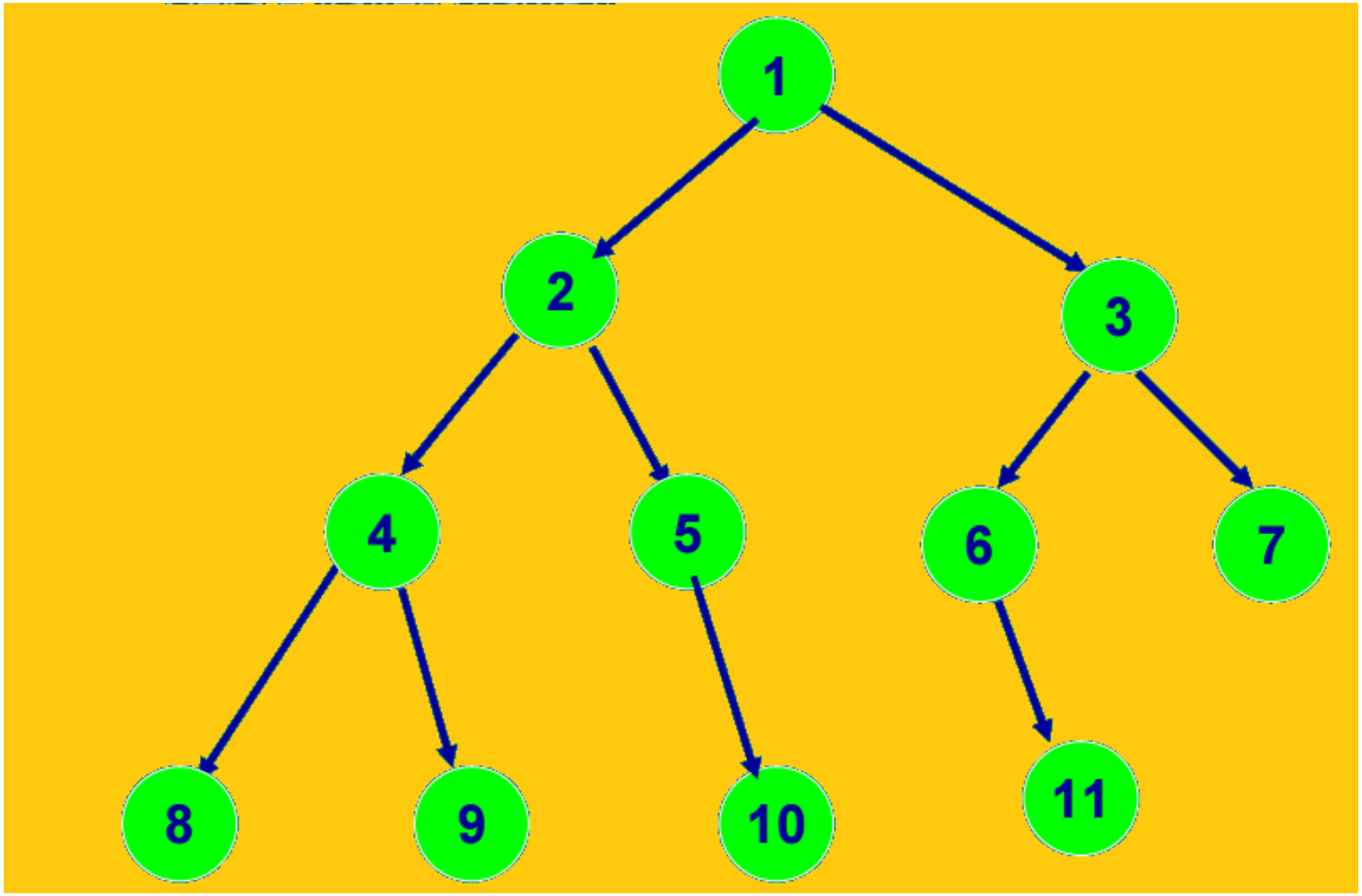
Grafik ağırlıklandırıldığında komşuluk matrisinde birimler yerine kenarların ağırlıklarını kaydedebiliriz.

Grafik geçişi

- İki grafik arama algoritması da uygulanabilir: derinlemesine ve genişlikte.
- **Derinlik öncelikli aramada**, kişi bir başlangıç düğümünden başlar ve ardılına, ardından ardılın ardılına doğru hareket eder ve bu, ardılı olmayan (hiçbir kenarın dışarı çıkmadığı) bir düğüme ulaşana kadar devam eder. Daha sonra önceki düğüme dönüş yapılır.
- **Genişlik öncelikli aramada**, kişi bir başlangıç düğümünden başlar ve onun tüm ardıllarını arar, ancak daha sonra ardılların ardıllarını arar.

Grafik geçişi

- Aşağıdaki grafik göz önüne alındığında:



Grafik geçişi

- Bir düğümün ardıllarını sayılarına göre artan sırada geçersek, o zaman:
- **Derinlik öncelikli aramada**, düğümler aşağıdaki sırayla geçilecektir:

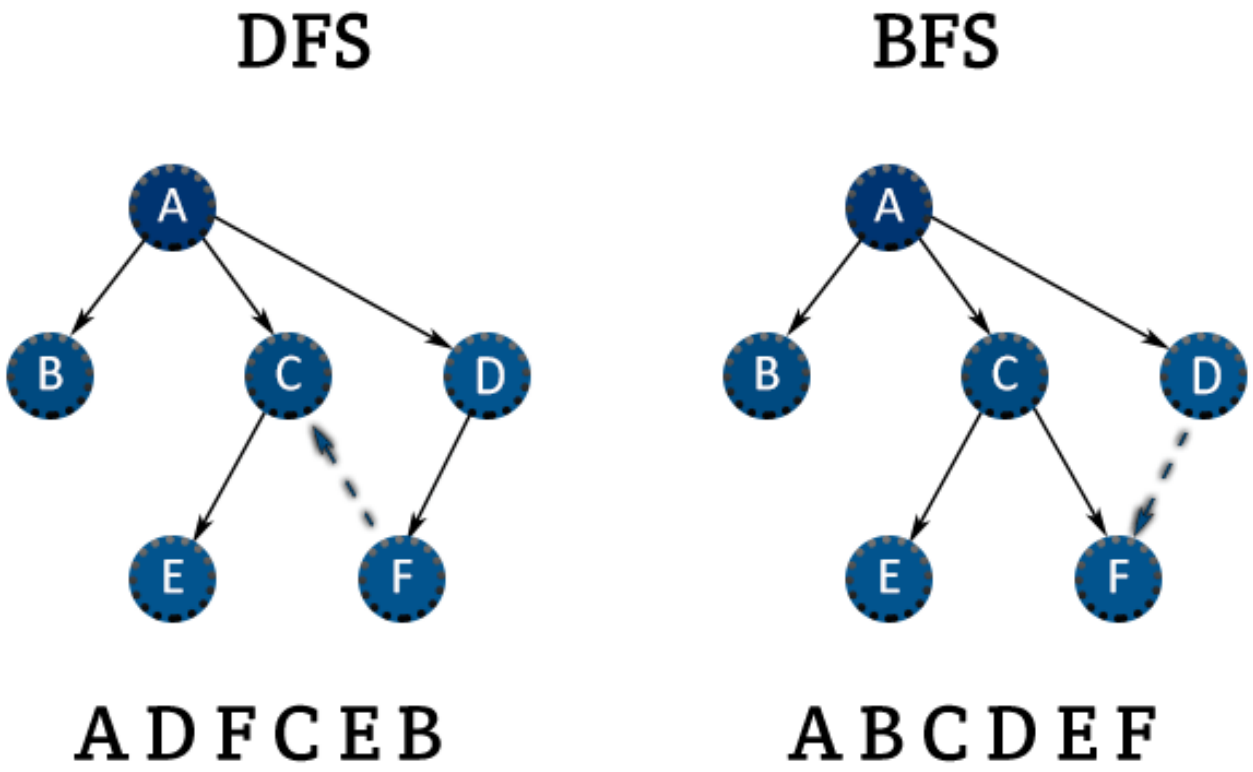
  1 2 4 8 9 5 10 3 6 11 7

➢ **Genişlik öncelikli aramada** düğümler aşağıdaki sırayla geçilecektir :

  1 2 3 4 5 6 7 8 9 10 11

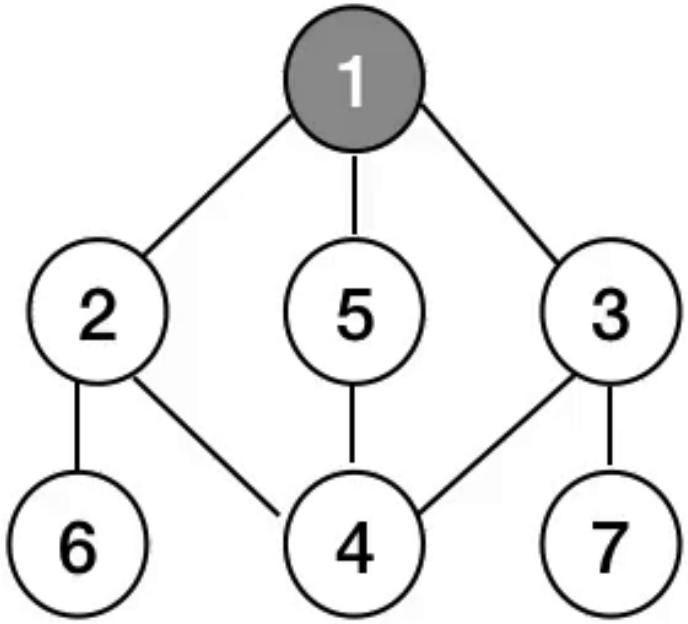
Grafik geçişi

- Aşağıdaki grafik göz önüne alındığında:



Grafik geçişi

- Aşağıdaki grafik göz önüne alındığında:



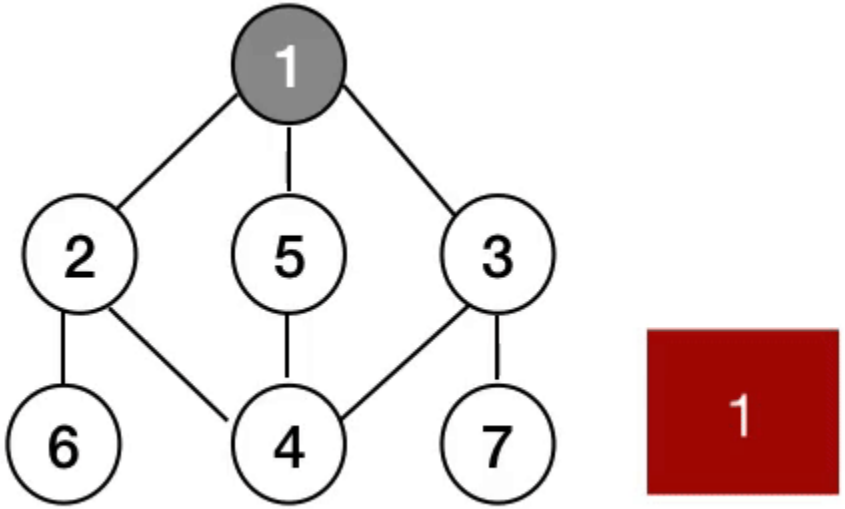
DFS

1

Grafik geçişi

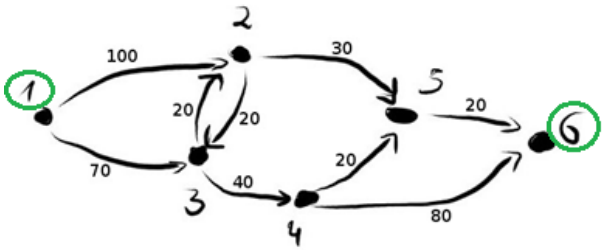
- Aşağıdaki grafik göz önüne alındığında:

BFS



Veritabanında grafik geçişi uygulaması

- Gerçek hayattan bir örnek alalım. Düğümlerin bir listesini ve aralarındaki bağlantıların bir listesini içeren bir **veritabanımız** olduğunu varsayalım (bunları şehirler ve yollar olarak düşünebilirsiniz). Görevimiz düğüm 1'den düğüm 6'ya en kısa yolu bulmaktır.



- Aslında bu grafik geçişinden başka bir şey değil. Aklımıza gelebilecek ilk fikir, her iki tablodaki tüm satırları almak ve tek bir **SQL sorgusu** ile bir DFS (Derinlik-Önce Arama) veya BFS (Genişlik-Önce Arama) algoritması uygulamak olabilir!

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Grafikleri temsil etmenin farklı yöntemleri nelerdir?

2. Grafiklerde arama yapmanın farklı yöntemleri nelerdir?

3. Grafik nedir ve ne için kullanılır?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

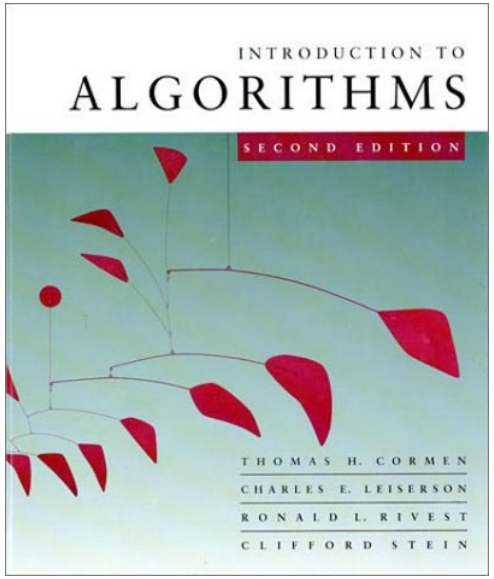
Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

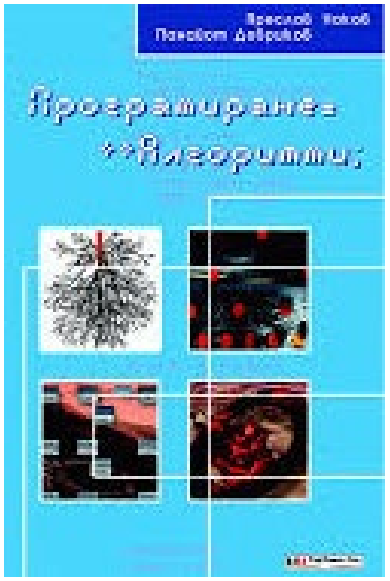
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

- **Modül 3. Sorgu optimizasyonu için algoritmalar ve veritabanlarındaki uygulamaları**
  - **Konu 4. Grafik algoritmaları**
    - Ders 2. Grafiklerin Ağaç Örtüsü.

GRAFİKLERİN AĞAÇ KAPAĞI

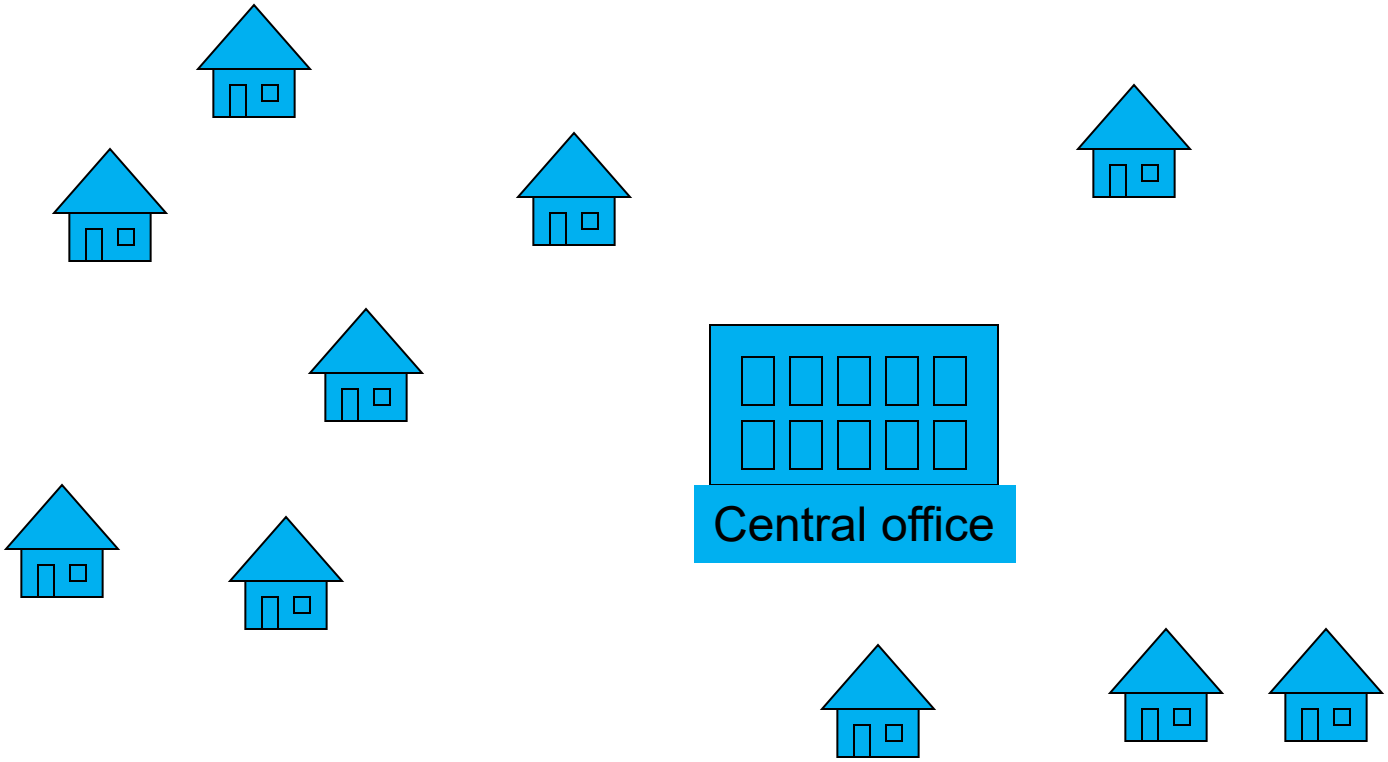


Carmen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2009) [1990]. Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill.

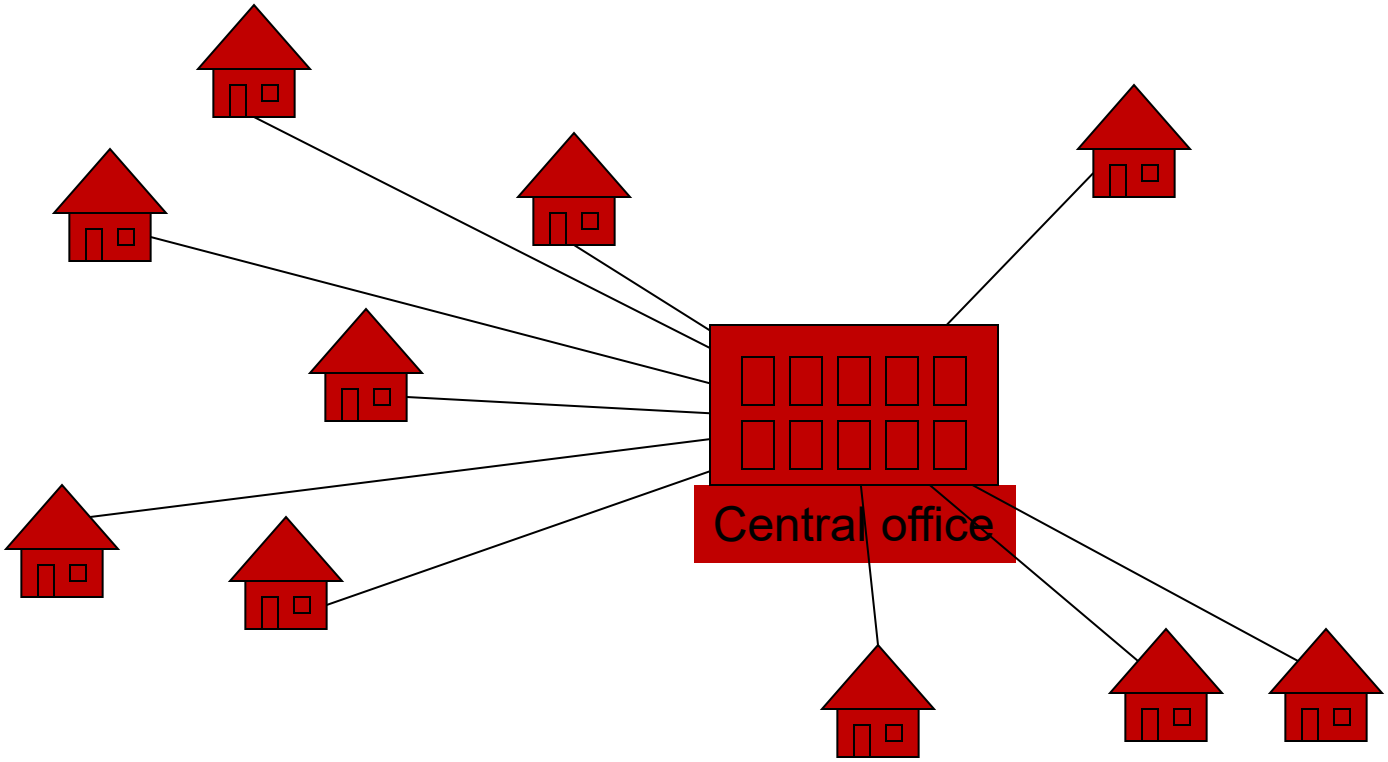


Preslav Nakov, Panayot Dobrikov (2002). Programming= ++Algorithms). Top Team Co, София.

Sorun: Telefon Kablosunu Döşemek

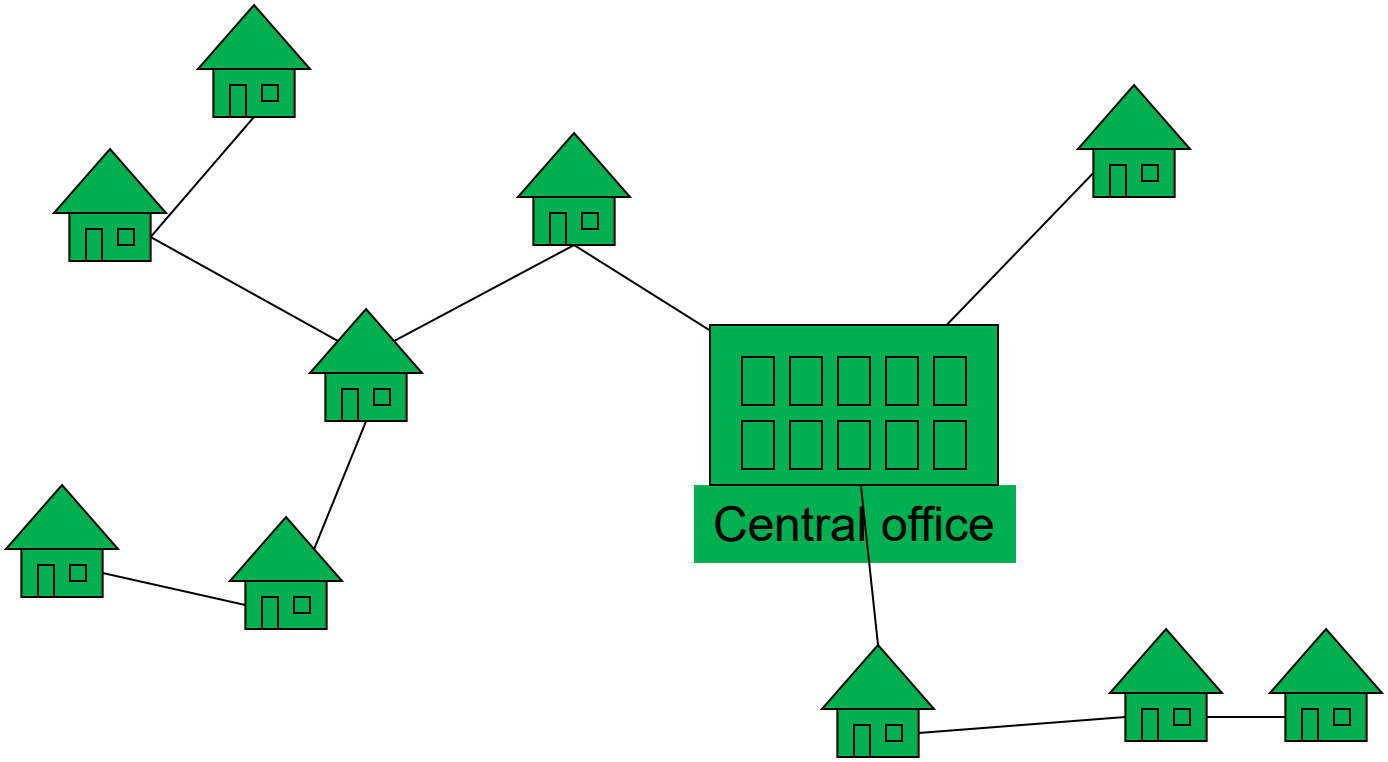


Kablolama: Naif Yaklaşım



PAHALI!

Kablolama: Daha İyi Yaklaşım



Müşterileri birbirine bağlayan toplam kablo uzunluğunu en aza indirin.

Örtü (Yayılan) Ağaç

- *G = (V, A)* rastgele bir grafik olsun. Grafiğin tüm düğümlerini içeren G'nin kenarları tarafından oluşturulan herhangi bir ağaca **örtü (yayılan) ağaç denir**.
- Ekonomi, yönetim vb. alanlardaki birçok gerçek görev, bazı optimallik koşullarını karşılayan örtü ağaçlarını bulma (arama, inşa etme) sorunları olarak yorumlanabilir.
- Pek çok problemde sadece bir örtü ağacı değil, aynı zamanda en uygun özelliklere sahip bir ağaç da bulmak gerekir. Yönsüz *G = (V, A)* grafiğinde, her bir kenarın *(u, v)* ağırlığı *c(u, v)* olsun; burada ağacın ağırlığı, katılan yayların ağırlıklarının toplamıdır.

Örtü (Yayılan) Ağaç

- ***Örnek:*** *Bir inşaat şirketi, altı yerleşim birimi arasında, bu yerleşim birimleri arasındaki iletişimi sağlayacak yolların yapılması projesini araştırıyor. Bu yerleşim yerleri arasındaki olası yolların her birinin döşenmesi için gereken maliyetler (fiyatlar) bilinmektedir. Şirket karayolu iletişimini minimum maliyetle sağlamak istiyor.*
- *Düğümleri şehirlere, kenarları bu şehirler arasında yönlendirilebilecek yollara karşılık gelen ve kenarların ağırlıkları da yolun güzergahını belirlemenin karşılık gelen maliyetleri olan yönlendirilmemiş G = (V, A) grafiğini düşünün. Açıkçası, bu durumda şirketin stratejisi, bu grafik için minimum ağırlığa (maliyete) sahip bir kapak ağacı bulmaktan ibarettir.*

Örtü (Yayılan) Ağaç

- Çıkış grafiğinin kenarları rastgele sırada dikkate alınır ve algoritmanın her adımında karşılık gelen kenarın kapak ağacına dahil edilip edilmeyeceğine karar verilir.
- Bu durumda ağaca dahil olan kenar **yeşil**, dahil olmayan kenar ise siyah renktedir. e. Algoritma kenarları renklendirme işlemidir. Renkli kenar daha fazla dikkate alınmaz.
- Algoritmanın her adımında, **yeşil** (yani zaten ağaçta yer alan) kenarlara sahip kümedeki söz konusu kenarın bir döngü oluşturup oluşturmadığı kontrol edilir. Eğer öyleyse - kenar siyah renklidir (yani kapak ağacı dahil değildir).

Örtü (Yayılan) Ağaç

- **Yeşil (**ağaca dahil olan) kenarlar bir veya daha fazla bileşen oluşturur.
- Her bileşenin düğümleri, demet olarak adlandıracağımız bir dizi düğüm oluşturur.
- Bu nedenle, söz konusu kenar, her iki düğümünün de o ana kadar oluşturulan demetlerden birine ait olması durumunda, ağaca dahil edilen kenarlarla bir döngü oluşturacaktır.
- Başka bir deyişle, algoritma, kenarları renklendirmenin yanı sıra, bağlı bileşenlerin düğümlerini de korur (depolar) ve günceller.

Örtü (Yayılan) Ağaç

- Kapak ağacı bulma algoritmasının açıklaması:
- ADIM 1. Rastgele bir kenar seçiyoruz. Bu kenarı **yeşile** boyadık. Bu kenarın düğümlerinden bir demet oluşturuyoruz.
- ADIM 2. Rastgele renklendirilmemiş bir kenar seçiyoruz. Böyle bir kenar yoksa 4. adıma geçin.
- ADIM 3. Aşağıdaki dört durum mümkündür:

Örtü (Yayılan) Ağaç

- Kapak ağacı bulma algoritmasının açıklaması :

a) kenarın düğümlerinden hiçbiri önceden oluşturulmuş bir gruba ait değildir - kenarı **yeşile** boyayın ve bu kenarın düğümlerinden yeni bir demet oluşturun. 4. adıma geçin.

b) her iki kenar da şu ana kadar oluşturulmuş aynı köşe grubuna aittir - kenarı siyaha boyarız, yani onu inşa ettiğimiz ağaca dahil etmiyoruz. 4. adıma geçin.

Co-funded by the Erasmus+ Programme of the European Union

Örtü (Yayılan) Ağaç

- Kapak ağacı bulma algoritmasının açıklaması:
- в) kenarın bir tepe noktası belirli bir kümeye aittir ve diğeri o ana kadar oluşan kümelerden herhangi birine ait değildir - kenarı **yeşil** renklendirin ve diğer tepe noktasını içeren kümeye dahil olmayan tepe noktasını dahil edin. 4. adıma geçin.
- г) kenar düğümleri farklı gruplara aittir - kenarı **yeşile** boyayın ve iki grubu yeni bir grup halinde birleştirin. 4. adıma geçin.

Örtü (Yayılan) Ağaç

- Kapak ağacı bulma algoritmasının açıklaması:
- ADIM 4. Grafiğin tüm köşeleri bir demet halinde ise (renkli kenarların sayısı grafiğin köşe sayısından bir eksikse), yeşil renkli kenarlar grafiğin kapak ağacını oluşturur. Son.
- ADIM 5. Adım 2'ye geçin.
- Bu tür formüle edilmiş algoritma, bir kapak ağacı mevcut olduğunda iyi "çalışır" ve eğer mevcut değilse "döngü" oluşturur.

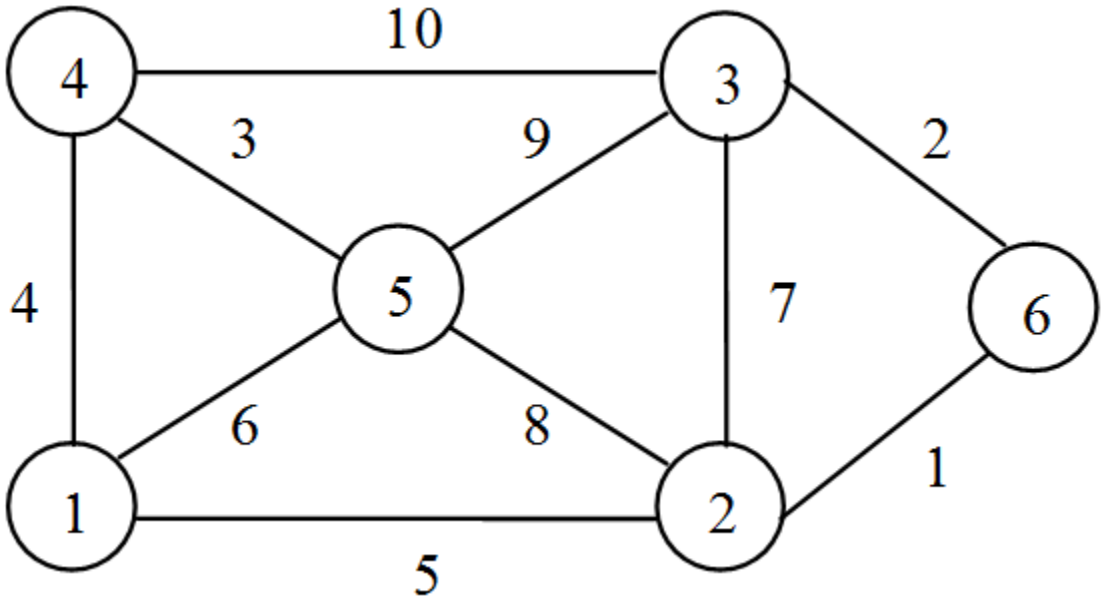
Örtü (Yayılan) Ağaç

- Kapak ağacı bulma algoritmasının açıklaması :

"Döngü"yü kaldırarak çözüm önerisi: 4. adımda grafiğin tüm kenarlarının renkli olup olmadığını kontrol ediyoruz. Durum böyleyse ve yeşil renkli kenar sayısı, grafiğin köşe sayısından bir eksiği kadar azsa, grafiğin kapsayan ağacı yoktur.

Örtü (Yayılan) Ağaç

- Algoritmayı aşağıdaki örnekle açıklayacağım:
- Örnek 2: G = (V, A) grafiği verildiğinde



- Bu grafik için bir kapak ağacı oluşturun.

Örtü (Yayılan) Ağaç

- Çözüm:

14	Edge	Color	Bunch 1 (of nodes)	Bunch 2 (of nodes)	Bunch 3 (of nodes)	…
1	(1, 5)	Green	1, 5	Ø	Ø	
2	(2, 6)	Green	1, 5	2, 6	Ø	
3	(4, 3)	Green	1, 5	2, 6	4, 3	
4	(5, 3)	Green	1, 5, 4, 3	2, 6	Ø	
5	(1, 4)	Black	1, 5, 4, 3	2, 6	Ø	
6	(5, 2)	Green	1, 5, 4, 3, 2, 6	Ø	Ø	

Altıncı kenarı dikkate alıp renklendirdikten sonra, grafiğin tüm köşeleri bir demet halinde ortaya çıktığı için (veya bu aynıdır - yeşil renkli kenarların sayısı, grafiğin köşe sayısından bir eksiktir), ağaç (1, 5), (2, 6), (4, 3), (5, 3), (5, 2) kenarlarından oluşan başlangıç grafiğinin kapsayan ağacıdır.

Örtü (Yayılan) Ağaç

- Çözüm:
- Bulunan örtü ağacının ağırlığı :

c(1, 5) + c(2, 6) + c(4, 3) + c(5, 3) +

c(5, 2) = 6 + 1 + 10 + 9 + 8 = **34**.

- Şimdi aynı grafiğe, kenarlarını artan ağırlık sırasına göre dikkate alarak algoritmayı uygulayalım.

Örtü (Yayılan) Ağaç

- Çözüm:

№	Edge	Color	Bunch 1 (of nodes)	Bunch 2 (of nodes)	….
1	(6, 2)	green	6, 2	Ø	
2	(6, 3)	green	6, 2, 3	Ø	
3	(4, 5)	green	6, 2, 3	4, 5	
4	(4, 1)	green	6, 2, 3	4, 5, 1	
5	(1, 2)	green	6, 2, 3, 4, 5, 1	Ø	

Bu durumda elde edilen örtü ağacının ağırlığı:

c(6, 2) + c(6, 3) + c(4, 5) + c(4, 1) + c(1, 2) = 1 + 2 + 3 + 4 + 5 = 15 ve maksimumdur kapak ağacı.

Örtü (Yayılan) Ağaç

- Aşağıdaki iki ifadeyi formüle edebiliriz :
- St. 1. **Minimum örtü ağacı** arama algoritması.

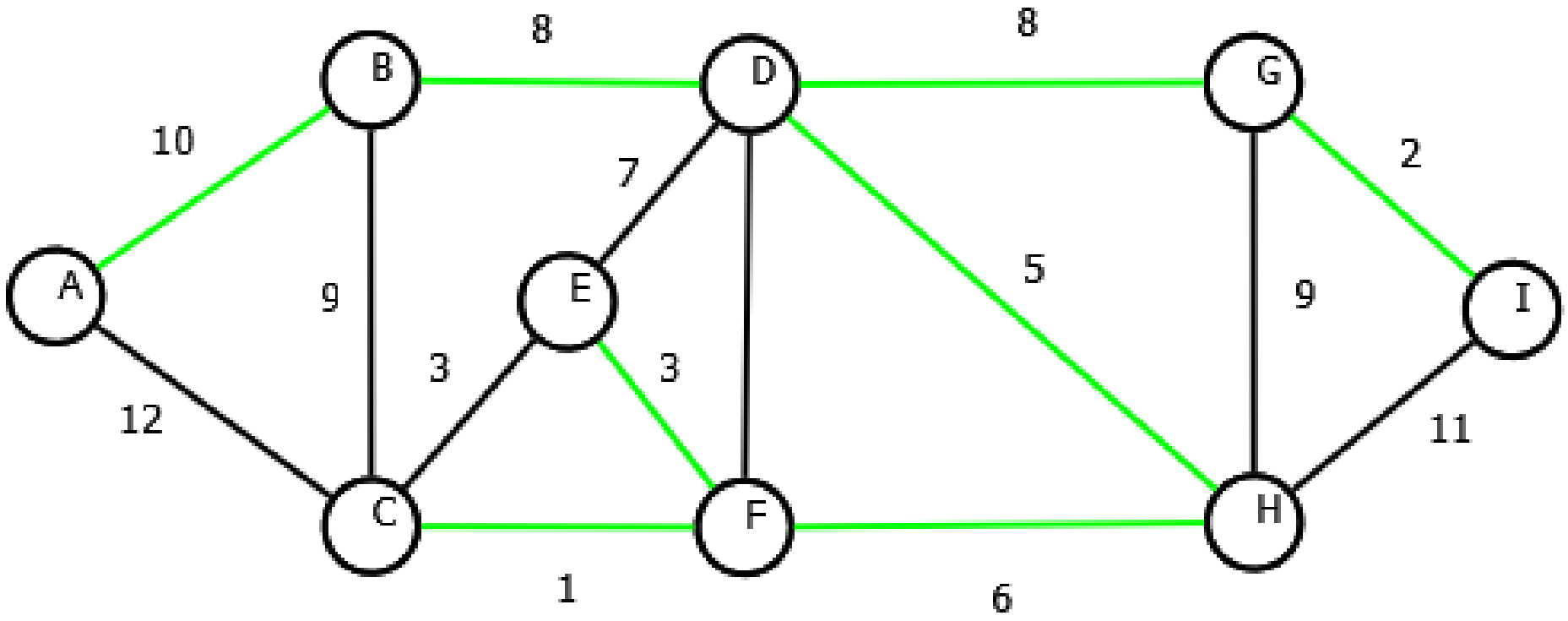
Kenarların ağırlıkları artacak şekilde dikkate alınarak kapak ağacı arama algoritması uygulanır. Eşit ağırlıklara sahip kenarlar varsa bunlar rastgele sırada dikkate alınır.

- St. 2. **Maksimum örtü ağacı** arama algoritması.

Kenarların ağırlıkları azalacak şekilde dikkate alınarak kaplama ağacı arama algoritması uygulanır. Eşit ağırlıklara sahip kenarlar varsa bunlar rastgele sırada dikkate alınır.

Örtü (Yayılan) Ağaç

- **Minimum Yayılan Ağaçlar örneği**



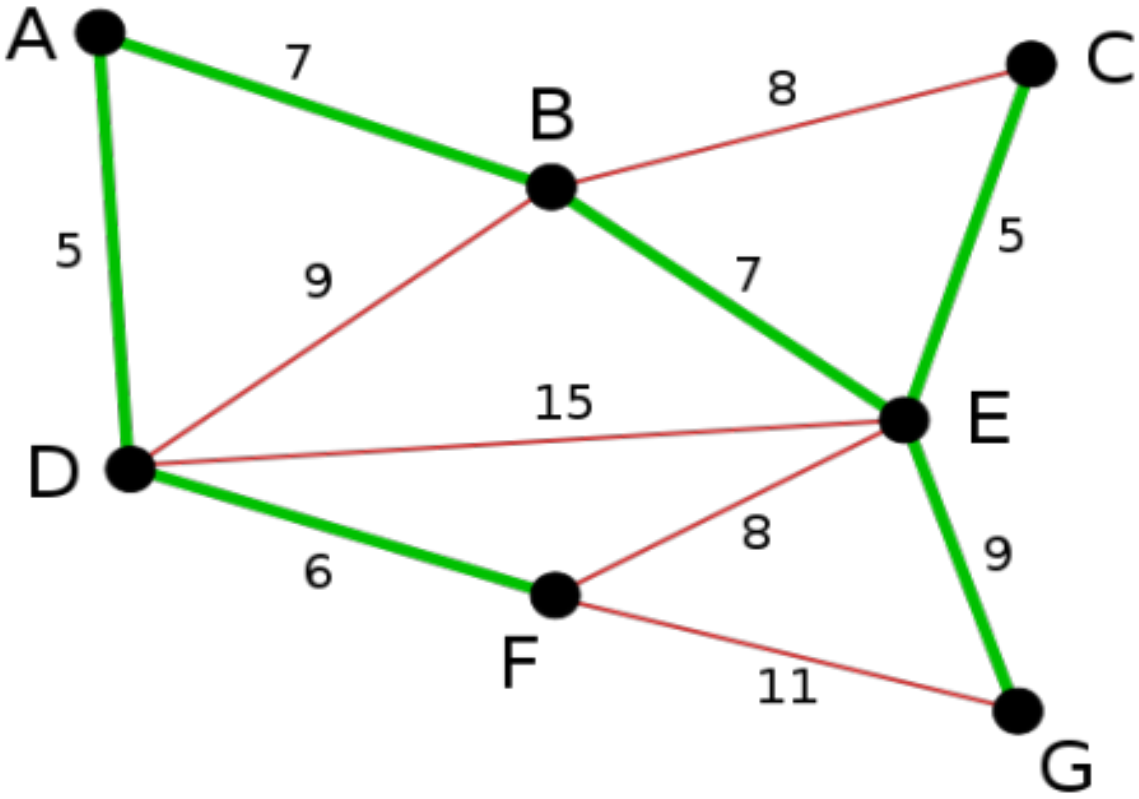
Örtü (Yayılan) Ağaç

- **Kruskal'ın Minimum Yayılan Ağaç Algoritması**
- Kruskal **açgözlü bir algoritma** önerdi. *G = (V, E), grafiğinde ağacımız G^T = (V, T) olsun:*

1. Grafiğin *N* köşesini *N* ayrı *Si, i = 1* kümesine bölüyoruz. . *N*;
2. Kenarları ağırlıklarının değerini azaltarak sıralıyoruz;
3. *N - 1* kez, $u \notin T$ olan ve iki ayrı S_i ve S_j kümesini yeni bir $S_k = S_i \cup S_j$ kümesine bağlayan en az ağırlığa sahip u kenarını seçiyoruz;

Örtü (Yayılan) Ağaç

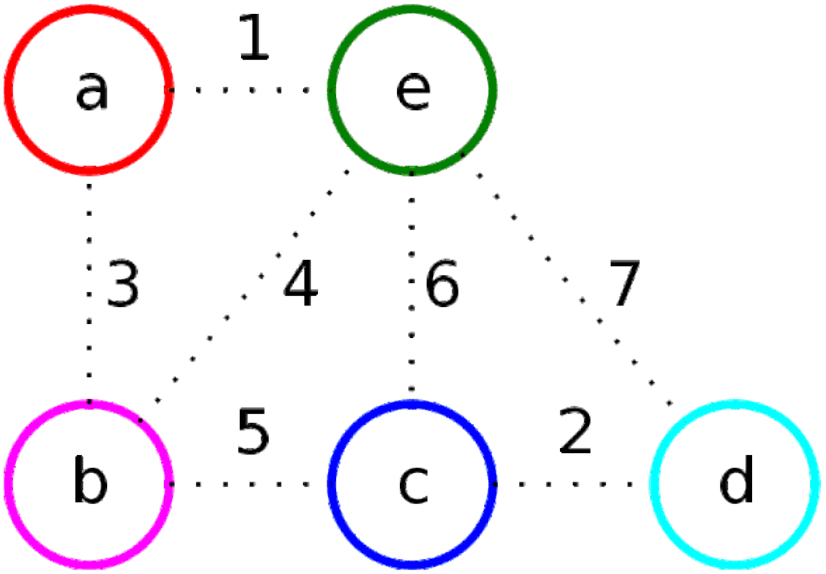
- **Minimum Yayılan Ağaç: Kruskal Algoritması**



Örtü (Yayılan) Ağaç

- **Minimum Yayılan Ağaç: Kruskal Algoritmasının etkileşimli görselleştirilmesi**

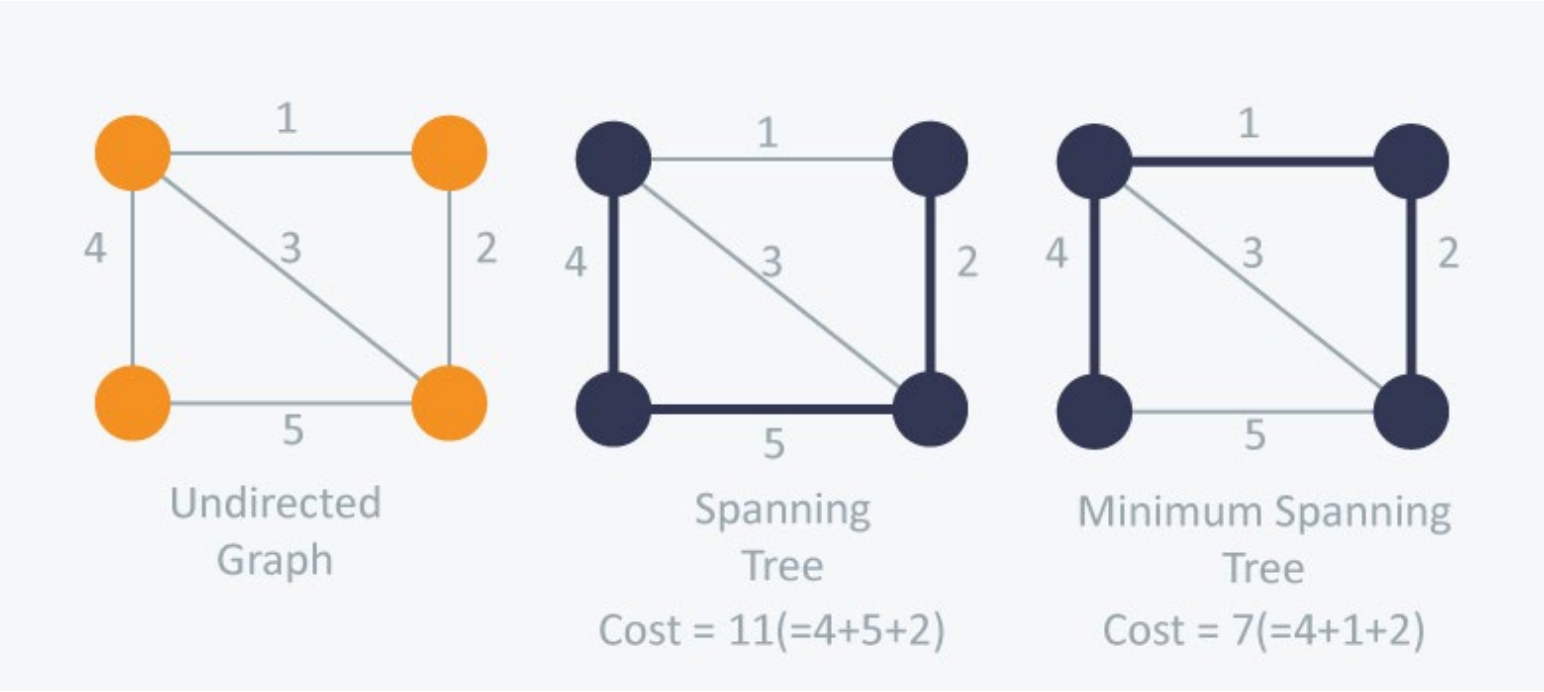
Edge	ab	ae	bc	be	cd	ed	ec
Weight	3	1	5	4	2	7	6



Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

Örtü (Yayılan) Ağaç



Bir veritabanında, yürütme sorgu ağacının oluşturulması için sorgu optimizasyon sürecinde Kruskal algoritması kullanılabilir.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Grafiklerin kapsamı problemi neleri içerir?
2. Minimum yayılan ağaç nedir?
3. Minimum yayılan ağaçta bir kenarın ağırlığını nasıl bulursunuz?
4. Minimum kapsayan ağaç oluşturmak için kullanılan bazı algoritmalar nelerdir?
5. Çok kenarı olan büyük grafiklerde Kruskal algoritmasını kullanmanın dezavantajları nelerdir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

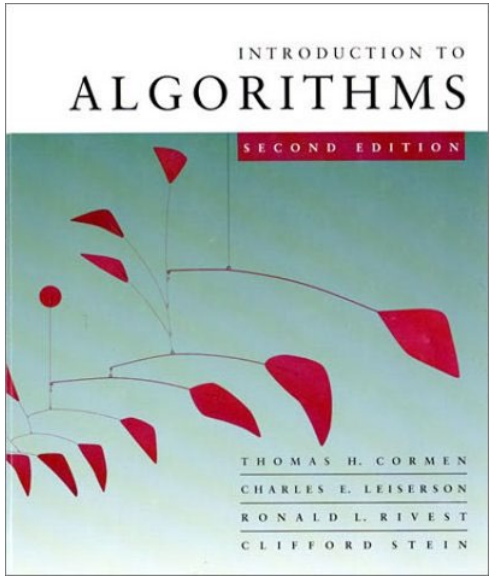
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

- **Modül 3. Sorgu optimizasyonu için algoritmalar ve veritabanlarındaki uygulamaları**
  - **Konu 4. Grafik algoritmaları**
    - Ders 3. En Kısa Yol Algoritmaları.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

EN KISA YOL ALGORİTMALARI



Carmen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2009) [1990]. Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill.

Preslav Nakov, Panayot Dobrikov (2002). Програмиране = ++Алгоритми). Top Team Co, София.

Giriş

- Bir grafikte iki köşe arasındaki en kısa yola, bir köşeden (düğümden) başlayıp diğerinde biten yol denir ve ilgili kenarların ağırlıklarının toplamı minimumdur.
- ***Tanım.*** Kenar ağırlıkları gerçek sayılarla verilen ağırlıklı yönlendirilmiş bir *G(V, E)* grafiği verilsin. *G* cinsinden bir yolun uzunluğuna, içindeki kenarların ağırlıklarının toplamı denir.
- Bazı problemlerde yol uzunluğunu bir toplam olarak değil, yola dahil olan kenarların (ve hatta köşelerin) ağırlıklarının başka bir fonksiyonu olarak tanımlamak mümkündür.

Grafiklerdeki en kısa yollar

- Bu algoritmaların bu durumlarda geçerli kalabilmesi için bazı spesifik optimallik kriterlerinin karşılanması gerekir (bunlar, söz konusu algoritmaya bağlıdır).
- Yolun basit olması gibi bir kısıtlama olamayacağından grafiğin döngü içerdiği durumlarda dikkatli olmalıyız.
- Yani, örneğin, minimum bir yol arıyorsak ve negatif bir döngü (negatif uzunlukta bir döngü) varsa, bu döngü üzerinde rastgele sayıda kez "dönebileceğiz"; yolun uzunluğu burada olacaktır. (içindeki kenarların toplamına eşit) eksi sonsuza doğru keyfi olarak azalacaktır.

Grafiklerdeki en kısa yollar

- Benzer şekilde, eğer maksimum bir yol arıyorsak ve pozitif bir döngü varsa, o zaman onu içeren her yol için döngü üzerinde "dönebilir" ve keyfi olarak büyük bir uzunluk elde edebiliriz.
- En kısa yol bir döngü içerebilir mi? Az önce gördüğümüz gibi tek bir negatif ağırlık döngüsü içeremez.
- Aynı zamanda pozitif ağırlıklı bir döngü içeremez, çünkü bu döngüyü yoldan kaldırmak aynı kaynak ve köşe hedefine sahip ve daha düşük yol ağırlığına sahip bir yol oluşturur.

Grafiklerdeki en kısa yollar

- Yani eğer $p = \langle v_0, v_1, \ldots, v_k \rangle$ yol ve $c = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ bu yol üzerinde pozitif ağırlığa sahip bir döngü ise (böylece. $v_i = v_j$ ve $w(c) > 0$), yolun $p' = \langle v_0, v_1, \ldots, v_i, v_{j+1}, v_{j+2}, \ldots, v_k \rangle$

ağırlık ve $w(p') = w(p) - w(c) < w(p)$, dolayısıyla p, v_0 'dan v_k 'ye giden en kısa yol olamaz.

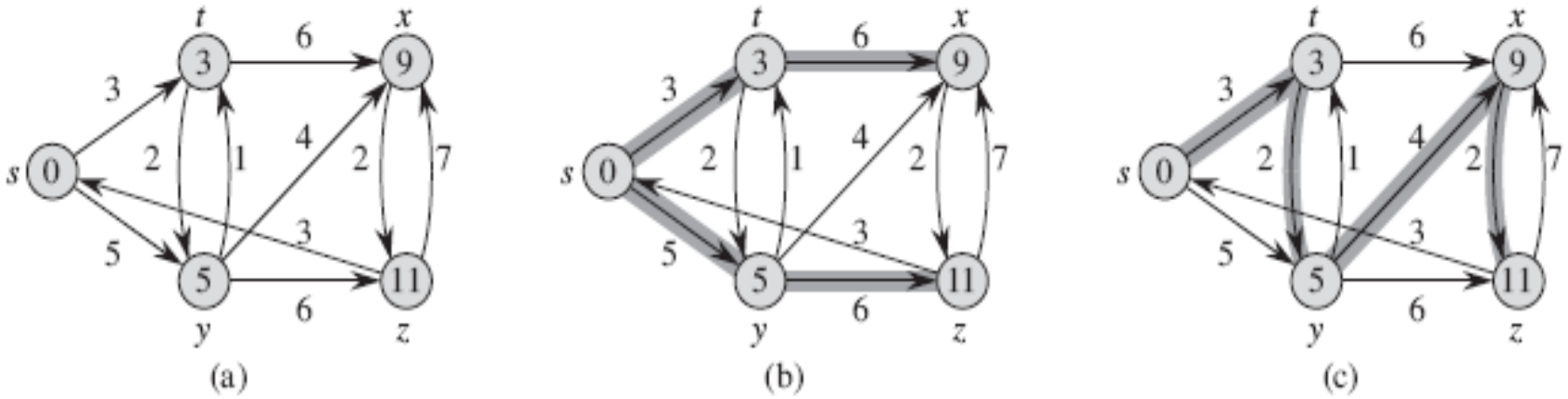
- Böylece geriye yalnızca ağırlığı 0 olan çevrimler kalır. Her yoldan ağırlığı 0 olan döngüleri çıkarıp aynı ağırlıkta bir yol üretebiliriz.
- Dolayısıyla, eğer bir s (kaynak) noktasından, ağırlığı 0 olan bir döngü içeren hedef v noktasına kadar en kısa yol varsa, o zaman bu grafikte bu döngü olmadan s'den v'ye başka bir en kısa yol vardır.

Co-funded by the Erasmus+ Programme of the European Union

Grafiklerdeki en kısa yollar

- Ağırlığı 0 olan çoklu döngülere sahip bir grafikte en kısa yolu ararken, bu döngüler olmadan grafikte en kısa yolu bulana kadar tespit edilen bu döngüleri tekrar tekrar kaldırabiliriz.
- **Bu nedenle**, en kısa yolları bulduğumuzda bunların döngüye sahip olmadığını varsayabiliriz, yani. örneğin, bunlar basit yollardır.
- Bir grafikteki her döngüsel olmayan yol $G=(V, E)$ en fazla $|V|$ içerdiğinden farklı köşeler, en fazla $|V|-1$ yayını (kenar) içerir.
- En fazla $|V|-1$ yayına sahip en kısa yolu bulmaya odaklanabiliriz.

Grafiklerdeki en kısa yollar



a) Kaynaklardan gelen en kısa yol ağırlıklarına sahip ağırlıklı yönlendirilmiş grafik

b) Renkli yaylar kaynaktan en kısa yolların bir ağacını oluşturur.

c) Kısa yol üzerinde aynı köke sahip başka bir ağaç.

Dijkstra'nın algoritması

- Belirli bir köşeden diğerlerine giden minimum yolları bulmanın en etkili yöntemi Dijkstra algoritmasıdır. **Bir tür açgözlü algoritmadır.** Yalnızca pozitif ağırlıklara sahip ağırlıklı grafiklerde çalışır.
- *n* köşeli ağırlıklı yönlendirilmiş bir *G(V, E)* grafiği verilsin. Algoritmanın uygulanabilmesi için *f(i,j)* kenarlarının ağırlıklarının pozitif sayılar olması gerekmektedir.
- Algoritmanın birçok versiyonu mevcuttur; daha yaygın olanı, bir tepe noktasını "başlangıç" tepe noktası olarak tanımlar ve ilk tepe noktasından grafikteki diğer tüm köşe noktalarına en kısa mesafeyi bulur.

Dijkstra'nın algoritması

- Algoritma ayrıca, hedefin tepe noktasına giden en kısa yol belirlendikten sonra algoritmayı durdurarak, bir tepe noktasından bir hedefin tepe noktasına en kısa mesafeyi bulmak için de kullanılabilir.
- Örneğin, bir grafiğin köşeleri şehirleri temsil ediyorsa ve uç noktaları doğrudan bir yolla bağlanan iki şehir arasındaki mesafeyi temsil ediyorsa, Dijkstra'nın algoritması bir şehir ile diğer tüm şehirler arasındaki en kısa yolu bulmak için kullanılabilir.

Dijkstra'nın algoritması

Olası herhangi bir optimizasyon uygulanmadan, bu algoritmanın karmaşıklığı $O(N^2)$ 'dir; burada N, verilen grafikteki köşelerin sayısıdır.

- Algoritma olası bir optimizasyonla uygulanırsa, $O(log(N)*M)$, karmaşıklığını elde eder; burada M, grafikteki kenar sayısıdır; bu genellikle daha iyi bir karmaşıklıktır, ancak N^2 'yi de aşabilir.
- En kısa yol bulma algoritmalarının en verimlisi olmasına rağmen bu algoritma her graf için geçerli değildir. Kullanıldığı grafikler negatif döngülere (kenar ağırlıklarının toplamının 0'dan küçük olduğu döngüler) sahip olmamalıdır.

Dijkstra'nın algoritması

- **Örnek 1:** Köşeleri dünyadaki havalimanlarına ve yayları da aralarındaki hava hatlarına karşılık gelen bir *G = (V, E)* grafiğini düşünün. Bu rotanın (yolun) bir anlamda optimal olması için iki nokta arasında seyahat etmek üzere bir rota nasıl seçilir?
- Optimallik uzunluk, hız, güvenlik vb. ile ilgili olabilir.
- Grafiğin her yayının kilometre, zaman vb. cinsinden yorumlanan bir *c(x, y)* ağırlığına eşlenmesine izin verin. Bu ağırlıklar aynı zamanda bir $C = (c_{ij})$ matrisinde de belirtilebilir. Ağırlık matrisinin elemanları pozitif, negatif veya sıfır olabilir.

Dijkstra'nın algoritması

- **Örnek 2:** Ticari bir gezgin, yolda diğer şehirleri de ziyaret etmek amacıyla Sofya'dan Varna'ya seyahat etmeyi planlıyor. Tüccar, ilgili şehirdeki müşterilerin olası ziyaretinin kendisine ne gibi kazançlar getireceğini büyük bir kesinlikle bilir. Bu tüccar hangi yolu seçmeli?
- Bir tüccarın ziyaret etmesi muhtemel şehirlerin köşelerini içeren bir G grafiği düşünün.
- Her yayın ağırlığı "seyahat maliyeti" - "beklenen gelir" anlamına geliyorsa, negatif ağırlığa sahip yayların (nakliye maliyetlerinin beklenen kârdan daha yüksek olduğu bölümler) olması muhtemel olduğu açıktır.

Dijkstra'nın algoritması

- Bu durumda ticari seyahat eden kişinin, "Sofia" ve "Varna" köşeleri arasındaki grafikteki en kısa rotaya karşılık gelen bir rota seçmesi gerekir.
- En kısa yol arama algoritmalarının "tüm ağırlıkların negatif olmadığı" ve ağırlıkların keyfi olduğu durumlarda farklı olduğunu not ediyoruz.
- Ayrıca grafikte (x, y) yayının olmaması durumunda ağırlığının = ∞ olduğunu kabul edeceğiz. Yayların ve yolların ağırlıklarını durum uzunlukları için en doğal terim olarak adlandıracağız.
- En kısa yolları bulma probleminde, G grafiğinde negatif ağırlıklı döngülerin bulunmaması kısıtı getirilmektedir.

Dijkstra'nın algoritması

Dijkstra'nın algoritması fikri

Bu algoritmada, tüm yayların c_{ij} uzunluklarının negatif olmadığı varsayılır.

- Algoritma, bir grafiğin köşelerini karşılık gelen sayılarla sıralı olarak işaretleme işlemi olarak düşünülebilir.
- Genel durumda, x köşesindeki d(x) işaretleme numarası geçicidir ve s'den x'e kadar olan yol uzunluğu için bir üst sınır verir.
- Algoritmanın yürütülmesi sırasında, her adımda geçici işaretleme numaralarından tam olarak biri sabit tutulacak (renkli) olacak şekilde işaretleme numaralarının değerleri azaltılır.

Dijkstra'nın algoritması

Dijkstra'nın algoritması fikri

- Bu durumda, sabit (renkli) işaretleme sayısı d(x) artık bir üst sınır değil, s'den x'e en kısa yolun tam uzunluğudur.
- Ayrıca karşılık gelen x köşesinin renkli olduğunu düşünüyoruz.
- En kısa yol uzunluğu dışında yolun kendisi de aranır ve grafiğin yaylarından biri de renklendirilir, böylece onu aranan yola dahil eder.

Dijkstra'nın algoritması

Dijkstra algoritmasının adımları

- ***ADIM 1.*** İlk köşe noktalarını renklendirin**,**

  d(s) = 0 (sabit işaretleme numarası),

  d(x) = ∞ (geçici işaretleme numaraları), x ≠ s,

  p = s (son renkli köşe).

- ***ADIM 2.*** *(Geçici işaretleme numaralarını değiştirme.) Renklendirilmemiş tüm x köşeleri için, aşağıdaki formülü kullanarak d(x)'i yeniden hesaplayın* :

  $$d(x) = \min \{d(x), d(p) + c(p, x)\} \quad (1)$$

Dijkstra'nın algoritması

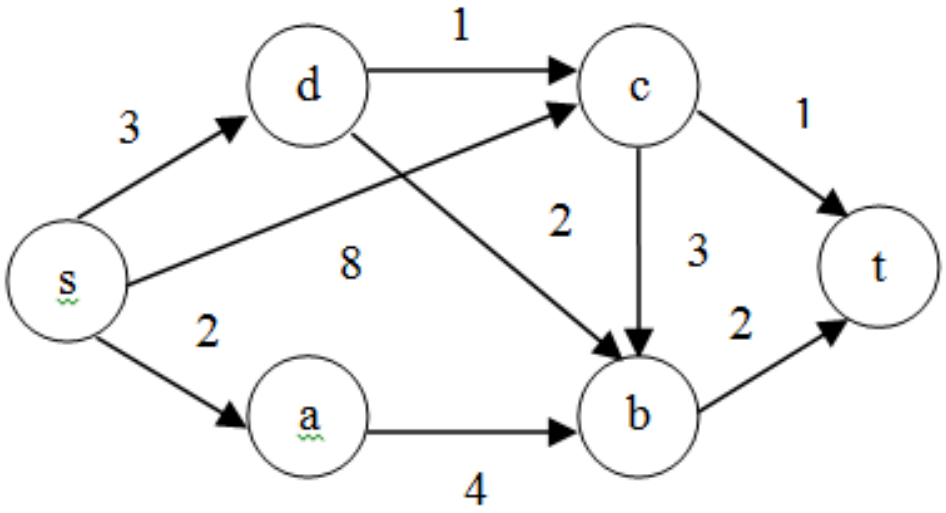
Dijkstra algoritmasının adımları

Renklendirilmemiş herhangi bir x köşesi için d(x) = ∞ ise, prosedürü sonlandırın; grafikte s'den renklendirilmemiş köşelere giden yol yoktur. Aksi takdirde, d(x) ağırlığının minimum olduğu x köşesini renklendirin. Ayrıca minimum değere ulaşılan x değerini girerek yayı da renklendirin. p = x'i koyun.

ADIM 3 . Eğer p = t ise işlemin sonu, s'den t'ye renkli yaylardan oluşan tek yol s ve t arasındaki en kısa yoldur. Aksi halde 2. adıma geçin.

Dijkstra'nın algoritması

Sorun : *Dijkstra algoritmasını kullanarak aşağıdaki grafikte s ve t arasındaki en kısa yolu bulun :*



ADIM 1. Diğer tüm köşeler için d(s) = 0 ve d(x) = ∞ olduğunu varsayıyoruz. Renklendiriyoruz. p = s olduğunu varsayıyoruz.

Dijkstra'nın algoritması

ADIM 2. Formül (1)'i kullanarak grafiğin renklendirilmemiş köşeleri için yeni işaretleme numaralarını hesaplıyoruz.

$d(a) = \min \{d(a), d(s) + c(s, a)\} = \min \{\infty, 0 + 2\} = 2,$

$d(d) = \min \{d(d), d(s) + c(s, d)\} = \min \{\infty, 0 + 3\} = 3,$

$d(c) = \min \{d(c), d(s) + c(s, c)\} = \min \{\infty, 0 + 8\} = 8.$

Yeniden hesaplanan sayılar arasında d(a) minimum işaretleme sayısı olduğundan, a tepe noktasını ve (s, a) yayını renklendiriyoruz. p = a olduğunu varsayıyoruz.

ADIM 3. t köşesi renkli olmadığından bizi 2. adıma yönlendirin.

Dijkstra'nın algoritması

ADIM 2. p = a

d(b) = min {∞, d(a) + c(a, b)} = min {∞, 2 + 4} = 6,

d(d) = 3, d(c) =8, d(t) = ∞.

Minimum sayı d(d) = 3'tür, dolayısıyla d köşesini ve yayı (s, d) renklendiriyoruz. p = d olduğunu varsayıyoruz.

ADIM 3. Bizi 2. adıma iletin.

ADIM 2. p = d

d(c) = min {8, d(d) + c(d, c)} = min {8, 3 + 1} = 4,

d(b) = min {6, d(d) + c(d, b)} = min {6, 3 + 2} = 5,

d(t) = ∞.

c köşesini ve yayı (d, c) renklendirin. p = c olduğunu varsayıyoruz.

ADIM 3. Bizi 2. adıma iletin.

Dijkstra'nın algoritması

ADIM 2. p = c

d(b) = min {5, 4 + 3} = 5,

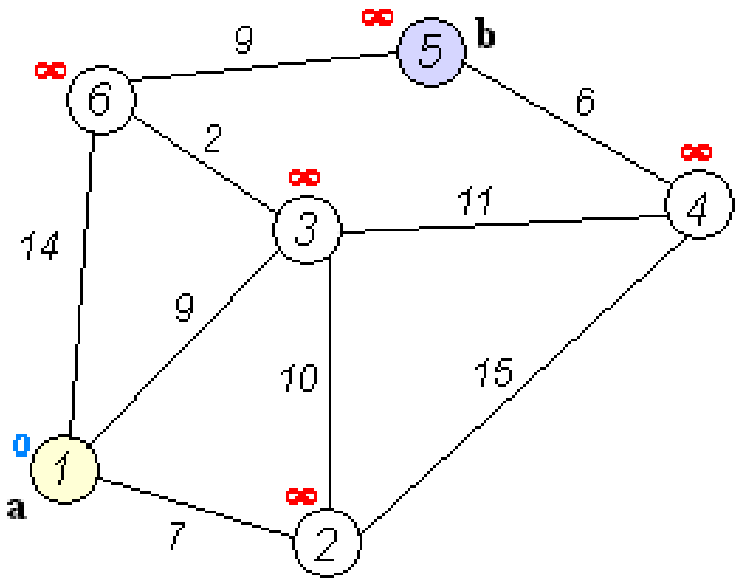
d(t) = min {∞, d(c) + c(c, t)} = min {∞, 4 + 1} = 5.

Minimum d(b) = d(t) = 5 sayılarından herhangi biridir. d(t)'yi seçip t tepe noktasını ve (c, t) yayını renklendiriyoruz. p = t olduğunu varsayıyoruz.

ADIM 3. Algoritmanın sonu. En kısa yolların (s, a), (s, d), (d, c), (c, t) ağacı aynı zamanda en kısa (s - t) yolunu da verir (s, d), (d, c) , (c, t) uzunluğu 3 + 1 + 1 = 5 olan.

Dijkstra'nın algoritması

Etkileşimli örnek:



Bellman-Ford algoritması

- **Bellman-Ford algoritması,** yönlendirilmiş, ağırlıklı bir grafikte bir tepe noktasından diğer tüm köşelere giden en kısa yolları hesaplayan bir algoritmadır. Bellman-Ford algoritması **Dinamik Programlama**ya bir örnektir ve aşağıdan yukarıya yaklaşımı izler.
- Dijkstra'nın algoritmasından daha yavaştır ancak kenar ağırlıkları negatif sayılar olan grafikler arasında geçiş yaparken çok daha esnektir.
- Algoritma, adını algoritmayı sırasıyla 1958 ve 1956'da yayınlayan iki yaratıcısı Richard Bellman ve Lester Ford Jr'dan almıştır. Edward F. Moore da aynı algoritmayı 1957'de yayınladı ve bu nedenle bazen **Bellman-Ford-Moore algoritması** olarak da anılıyor.

Bellman-Ford algoritması

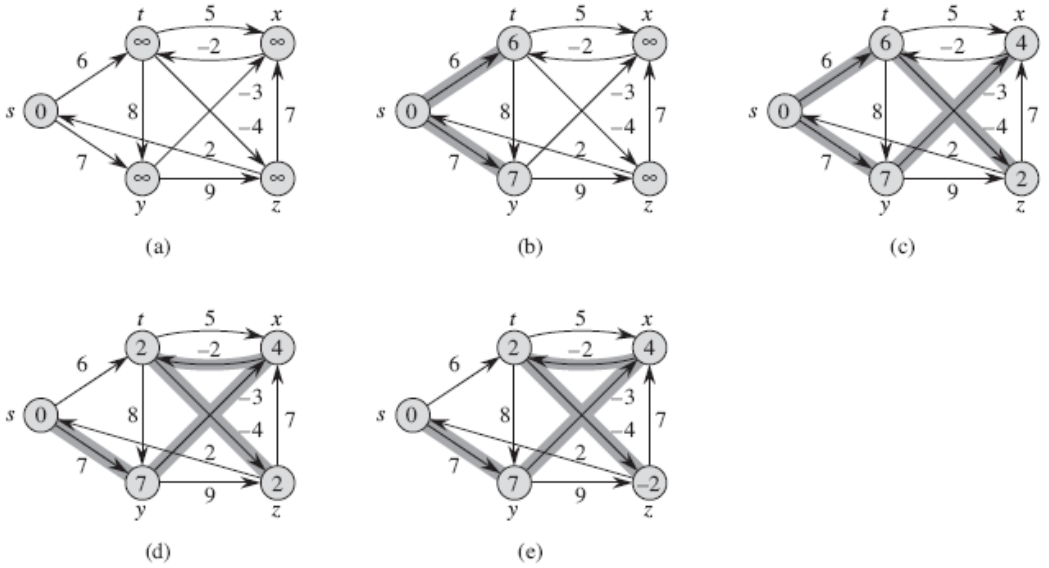
Bellman-Ford algoritması, yarış programlamasında çok nadiren kullanılır; özellikle negatif kenarları olan bir grafikte en kısa yolu aramak gerektiğinde. İşte burada :

1. İlk köşeye bitişik her köşe için, o an için en kısa yolun değeri, yanındaki kenardır ve geri kalanı için sonsuzdur.
2. Grafikteki her bir köşe için, ona giden en kısa yolun, ona ulaşılabilecek her bir köşenin yollarının toplamından + aralarındaki kenarın uzunluğundan daha büyük olup olmadığını kontrol ederiz.

Adım 2 N-2 kez tekrarlanır, ardından grafikte son en kısa yollara sahip oluruz.

Bellman-Ford algoritması

5 köşeli bir grafik için Bellman-Ford algoritmasının uygulanması

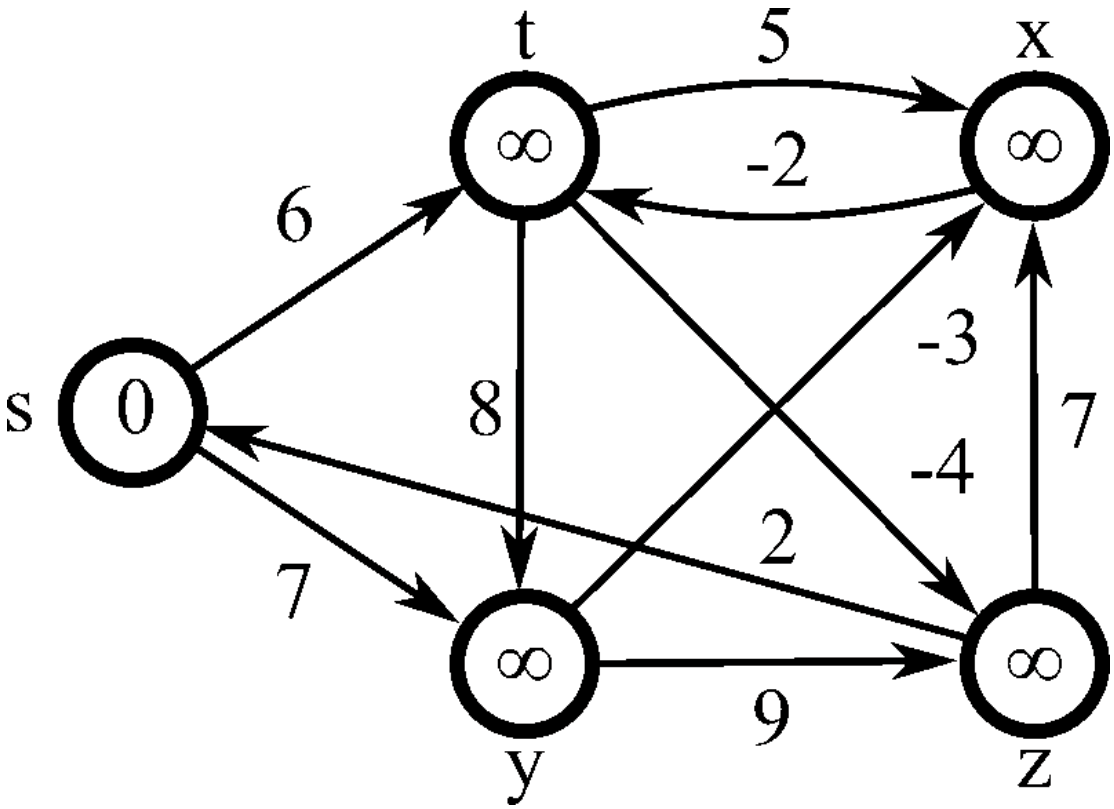


a) Köşelerin ilk geçişinden hemen önceki durum.

(b) - (e) Köşelerden sonraki her geçişten sonraki durum.

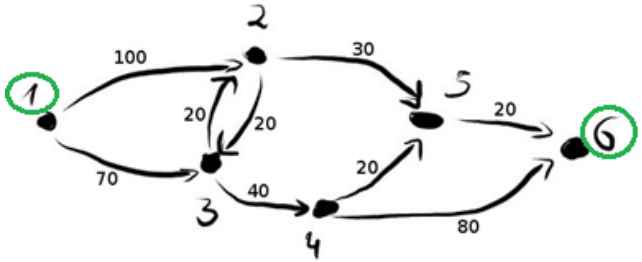
Bellman-Ford algoritması

Etkileşimli örnek:



Veritabanında grafik geçişi uygulaması

- Önceki derslerden birindeki örneğe geri dönelim. Düğümlerin bir listesini ve aralarındaki bağlantıların bir listesini içeren bir **veritabanımız** olduğunu varsayalım (bunları şehirler ve yollar olarak düşünebilirsiniz). Görevimiz düğüm 1'den düğüm 6'ya en kısa yolu bulmaktır.



- her iki tablodan da tüm satırları alıp tek bir **SQL** sorgusuyla Dijkstra algoritmasını veya Bellman-Ford algoritmasını uygulayabilirdik!

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Bir grafikteki en kısa yol nedir?
2. Dijkstra Algoritması neden negatif ağırlıklarda başarısız oluyor?
3. Dijkstra'nın algoritması neden açgözlü bir algoritma olarak değerlendiriliyor?
4. Bellman Ford'un ve Dijkstra'nın algoritmaları arasındaki farklar nelerdir?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

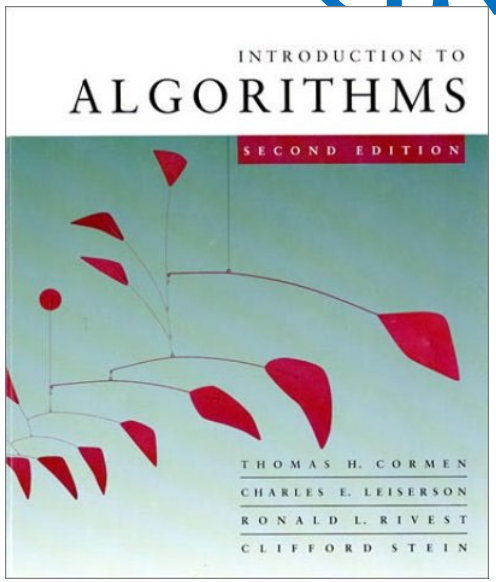
VERİTABANLARINDA ONLINE UZAKTAN EĞİTİM

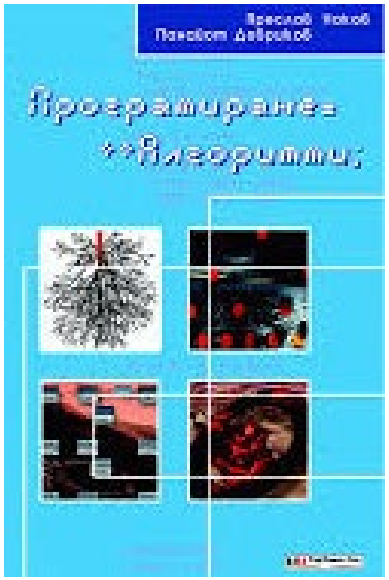
- **Modül 3. Sorgu optimizasyonu için algoritmalar ve veritabanlarındaki uygulamaları**
  - **Konu 4. Grafik algoritmaları**
    - Ders 4. Grafikte maksimum akış.

Tackling COVID-19 – Challenges Aiming at Bettering the Higher Education Quality (TACOHEQ)
Project Number: 2020-1-BG01-KA226-HE-095131

Co-funded by the Erasmus+ Programme of the European Union

MAKSİMUM AKIŞ GRAFİĞİ





Carmen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2009) [1990]. Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill.

Preslav Nakov, Panayot Dobrikov (2002). Pragramming= ++Algorithms). Top Team Co, Sofia.

Co-funded by the Erasmus+ Programme of the European Union

Taslak

- Ağ akışı tanımları
- Akış örnekleri
- Yolları Artırma
- Kalıntı Grafiği
- Ford Fulkerson Algoritması
- Kesimler
- Maxflow-MinCut Teoremi

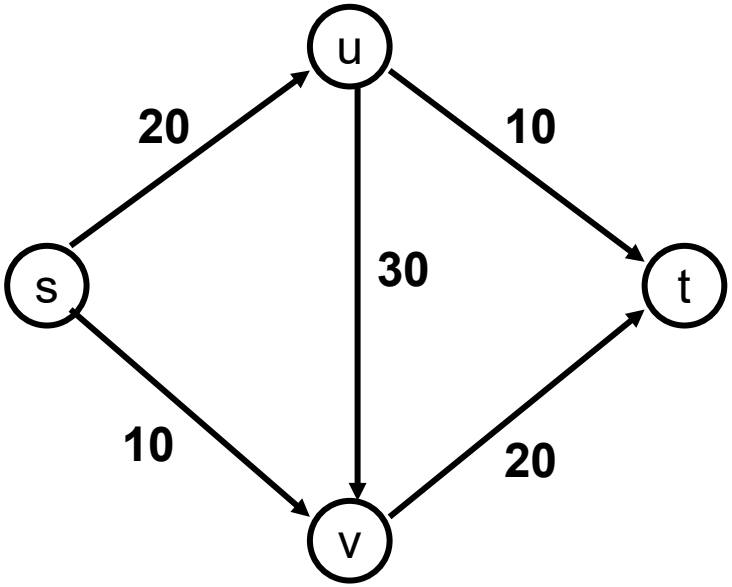
Co-funded by the Erasmus+ Programme of the European Union

Ağ Akışı Tanımları

- Kapasite
- Kaynak, Batma
- Kapasite Durumu
- Koruma Durumu
- Bir akışın değeri

Co-funded by the Erasmus+ Programme of the European Union

Akış Örneği



Co-funded by the Erasmus+ Programme of the European Union

Giriş

- **Akış,** nesneleri bir grafiğin bir köşesinden diğerine yayları (veya kenarları) boyunca aktarmanın bir yolunu belirler.
- Bu nicelik aktarımının başladığı ilk tepe noktasına kaynak adı verilir ve genellikle s ile gösterilir.
- Bu aktarımın gerçekleştirilmesi gereken tepe noktasına lavabo (stok) adı verilir. Lavabo genellikle t ile gösterilir.
- Yönlendirilmiş bir grafik düşünün.

Giriş

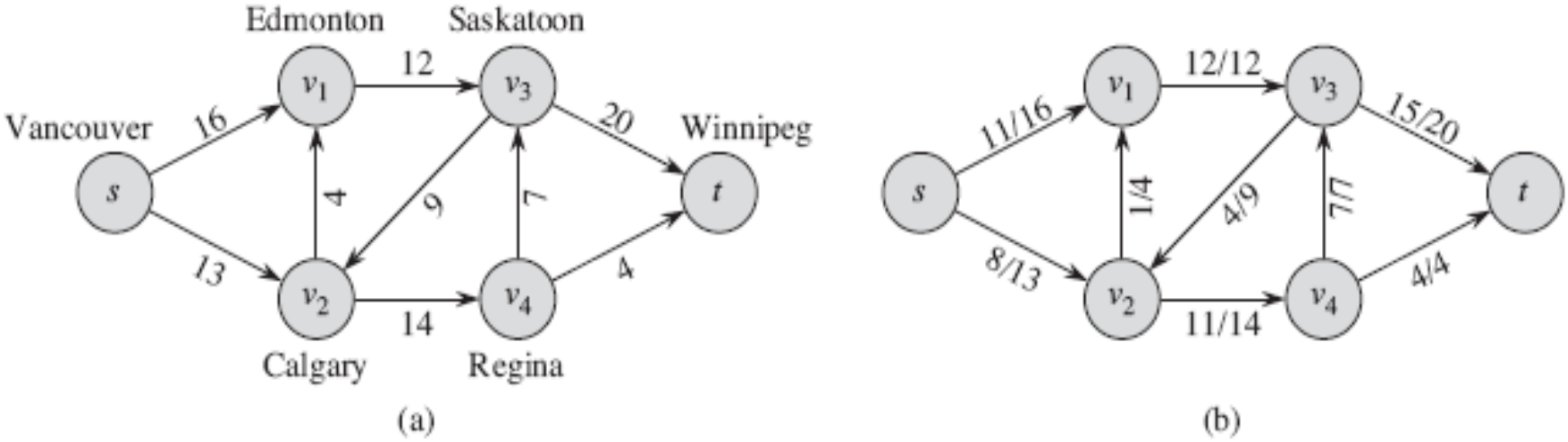
- Bunu, bir maddenin kaynaktan lavaboya doğru hareket ettiği bir boru ağı olarak düşünebiliriz.
- Nesne akışları, elektriğin kablolar, boru parçaları, bilgi bağlantıları veya mallar aracılığıyla üreticiden tüketiciye akışı olarak düşünülebilir.
- Bir kaynaktan havuza doğru hareket eden, "akan" nesnelere yalnızca akış birimleri veya birimleri denir.
- Arktan geçebilecek akış birimlerinin miktarına kapasite (verim) denir.
- Ağ trafiği maliyetlerini azaltmak veya zaman aşımı hatalarını önlemek için SQL sorgu optimizasyonunda max flow algoritmasını kullanabiliriz.

Akış ağları

- Bir ağ, her yaya bir miktar verim (kapasite) atanan bir *G=(V, E)* grafiğidir.
- Yönlendirilmiş bir $G=(V,E)$ grafiğinde, her bir kenarı $(u,v)\in E$ verim (kapasite) adı verilen $c(u,v)\geq 0$ sayısıyla gösterilen bir akış ağını ele alalım.
- $(u,v)"\notin" E$ durumunda. *c(u,v)=0* olduğunu varsayıyoruz. Grafikte iki köşe belirledik: kaynak *s* ve sink *t*.

Akış ağları

- Kolaylık sağlamak için, her $v \in V$ köşesinin bir şekilde kaynaktan havuza $s \to v \to t$ yolu üzerinde bulunduğunu varsayabiliriz. Bu durumda grafik aynı zamanda $|E| \geq |V|$-1'i de bağlar.
- Aşağıdaki şekil bir örneği göstermektedir:



Akış ağları

- Yukarıdaki şekilde şunu görüyoruz:
- $G=(V,E)$ cinsinden akış, malların Vancouver'daki kaynak fabrikadan Winnipeg'deki bir lavabo t - deposuna nakledilme olasılığını açıklar. Her kenarda günde gönderilebilecek maksimum kutu sayısı yazılıdır.
- f $|f|= 19$ değerindeki G ağındaki akış. Akış/kapasite her yayda gösterilir.
- $|f|$ değeri akışın f şu şekilde tanımlanır:

$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s) ,$$

Ford-Fulkerson yöntemi

- Ford-Fulkerson yöntemi veya Ford-Fulkerson algoritması (FFA), bir akış ağındaki maksimum akışı hesaplayan açgözlü bir algoritmadır.
- Bu yöntemi uygulamak için çeşitli algoritmalar olduğundan, algoritmadan değil yöntemden bahsediyoruz; operasyon sırasında farklılık gösterirler.
- Ford-Fulkerson yönteminde üç kavram anahtar rol oynamaktadır: artık ağ, tamamlayıcı yol ve kesim.
- Maksimum akışın Ford-Fulkerson yöntemiyle bulunması adım adım gerçekleştirilir.
- Başlangıçta akış sıfırdır (ve değeri sıfıra eşittir).

Co-funded by the Erasmus+ Programme of the European Union

Ford-Fulkerson yöntemi

- Her adımda akışın değerini arttırıyoruz. Bunu yapmak için, daha fazla "maddeyi" atlayabileceğimiz bir artırıcı yol buluruz ve bunu akışı artırmak için kullanırız.
- Arttırma yolu oluşana kadar bu adım tekrarlanır.
- Bu Maksimum akış minimum kesim teoremini gösterir.

Ford-Fulkerson yöntemi

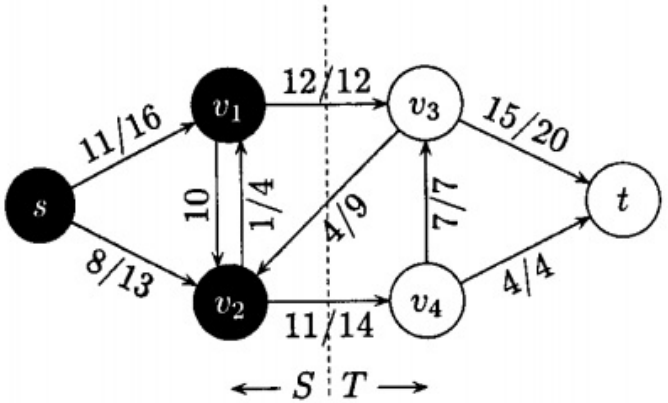
- **Artık ağlar**
- Bir ağ ve onun içindeki bir akış verilsin. Gayri resmi olarak, artık ağ, akış ağını temsil eder, ancak yalnızca akışın iletilebildiği kenarlar kalır.
- Artık ağ aynı *G(V, E)* grafiğinde tanımlanır, ancak yeni bir fonksiyon eklenir - artık kapasite: *cf(u, v) = c(u, v) - f(u, v).* O zaman *Gf(V, E'),* burada *E' cf(u, v) > 0* olan kenarlardır. Biz bunlara artık kapasite kenarları diyeceğiz.

Ford-Fulkerson yöntemi

- **Ağ kesintileri**
- Ford - Fulkerson yöntemi, maksimum akış elde edilene kadar artırma yolu boyunca ardışık akışlar ekler.
- Maksimum akış ve minimum kesme teoremine göre, artık ağ bir genişletme yolu içermediğinde akış maksimumdur.
- Minimum kesinti, en düşük verime sahip olanıdır (ağdaki tüm kesintiler arasında).

Ford-Fulkerson yöntemi

- **Ağ kesintileri**
- Aşağıdaki şekil bir ağ kesintisini göstermektedir. $(\{s, v_1, v_2\}, \{v_3, v_4, t\})$



- Bu kesimden geçen akış :

$$f(v_1, v_3) + f(v_2, v_3) + f(v_2, v_4) = 12 + (-4) + 11 = 9$$

ve kesimin verimi :

$$c(v_1, v_3) + c(v_2, v_4) = 12 + 14 = 26$$

Ford-Fulkerson yöntemi

- **Ağ kesintileri**
- Yukarıdaki şekildeki ağdaki *(S,T)* kesimini yapın.
- Burada S={s,v_1,v_2} (siyah köşeler) ve T={v_3,v_4,t} (beyaz köşeler).
- $f(S,T)$=19 (kesim boyunca akış) ve s(S,T)=26 (çıktı) konumunda
- Görülebileceği gibi, kesimin veriminin aksine kesimden geçen akış negatif terimler de içerebilir.

Ford-Fulkerson yöntemi

- **Ford-Fulkerson algoritmasının genel şeması**
- Ford-Fulkerson yöntemi, keyfi olarak artan bir p yolu seçerek ve her adımda akışıf artırarak çalışır.
- $C_f(p)$ akış değerini p yoluna ekleyin.
- Bu algoritma mevcut akış değerlerini saklamak için f[u, v] dizisini kullanır.
- $c(u,v)$=0, eğer (u,v) E'ye ait değilse.
- Ford-Fulkerson prosedürünün çalışma süresi, kaç kez çalıştırılması gerektiğine bağlıdır.

Ford-Fulkerson yöntemi

- Prensip olarak, akış değeri bir maksimuma ulaşmadan daha küçük adımlarla artmaya devam ederse algoritma hiç durmayabilir.
- Ancak algoritmayı polinom zamanda çalıştırmak da mümkündür.
- Ford-Fulkerson algoritması iyidir çünkü diğer ağ akışı algoritmalarına göre daha hızlıdır ve önemli ölçüde daha az karmaşıktır.
- Optimizasyon yöntemleri grubuna aittir ve 1956 yılında oluşturulmuştur.

Ford-Fulkerson yöntemi

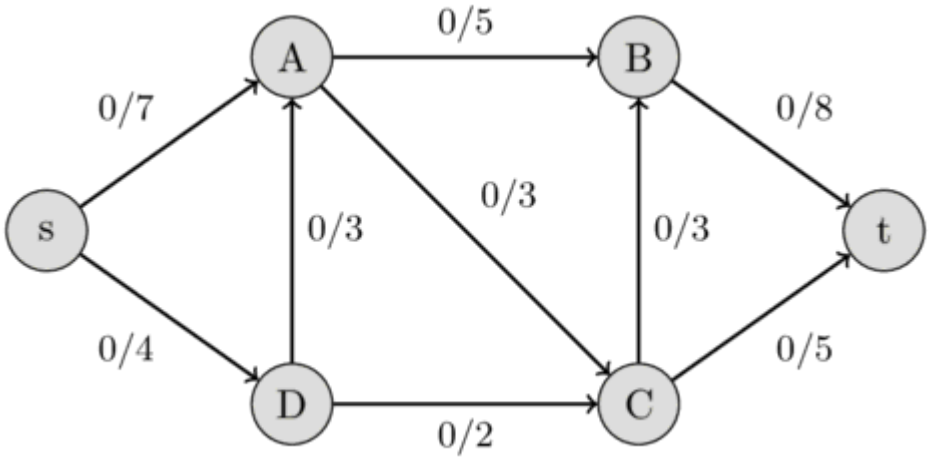
- Diğer olası çözümler, Ford-Fulkerson algoritmasının kapasitelerin büyüklüğüne bağımlılığını ortadan kaldıran **Edmonds-Karp** algoritması, Goldberg algoritmaları, Onaga vb.'dir.
- Pratik olarak, oldukça büyük ağlar için kabul edilebilir bir sürede maksimum verimi elde edebilecek bir algoritma yoktur.
- Maksimum akış minimum kesim teoremi, minimum kesimin sayısal değerini bulmak için Ford-Fulkerson algoritmasını kullanmamız için bize bir neden verir.

Ford-Fulkerson yöntemi

- *Ancak önemli bir soru da aslında nerede bulunduğudur (minimum kesim)?*
- Ağın iki düğümü arasında ihtiyaç duyulan trafiğin, onları bağlayan hatların kapasitesinden daha fazla olduğu ortaya çıkarsa, bulgu hangi hatların onu sınırladığını gösterecektir (kullanılmayan rezervler nedeniyle yalnızca bir hattın trafiği sınırlandırması gerekebilir). genişletilebilir) ve bu nedenle hangi satırların optimize edilmesi gerektiği sorusuna cevap verecektir.

Ford-Fulkerson yöntemi

- *ÖRNEK:*
- Aşağıdaki resimde bir akış ağı gösterilmektedir. Her bir kenarın ilk değeri başlangıçta 0 olan akışı, ikinci değeri ise kapasiteyi temsil eder.

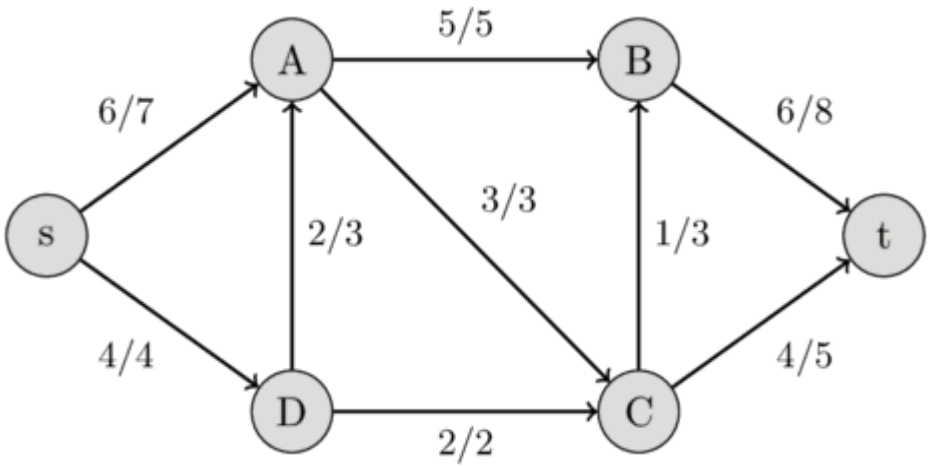


Ford-Fulkerson yöntemi

- *ÖRNEK:*
- Bir ağın akışının değeri, kaynakta üretilen tüm akışların toplamı s veya eşdeğer olarak havuz t tarafından tüketilen tüm akışların toplamıdır. Maksimum akış, mümkün olan maksimum değere sahip akıştır. Bir akış ağının bu maksimum akışını bulmak, çözmek istediğimiz sorundur.
- Su borularıyla yapılan görselleştirmede sorun şu şekilde formüle edilebilir: Borulardan kaynaktan lavaboya ne kadar su itebiliriz?

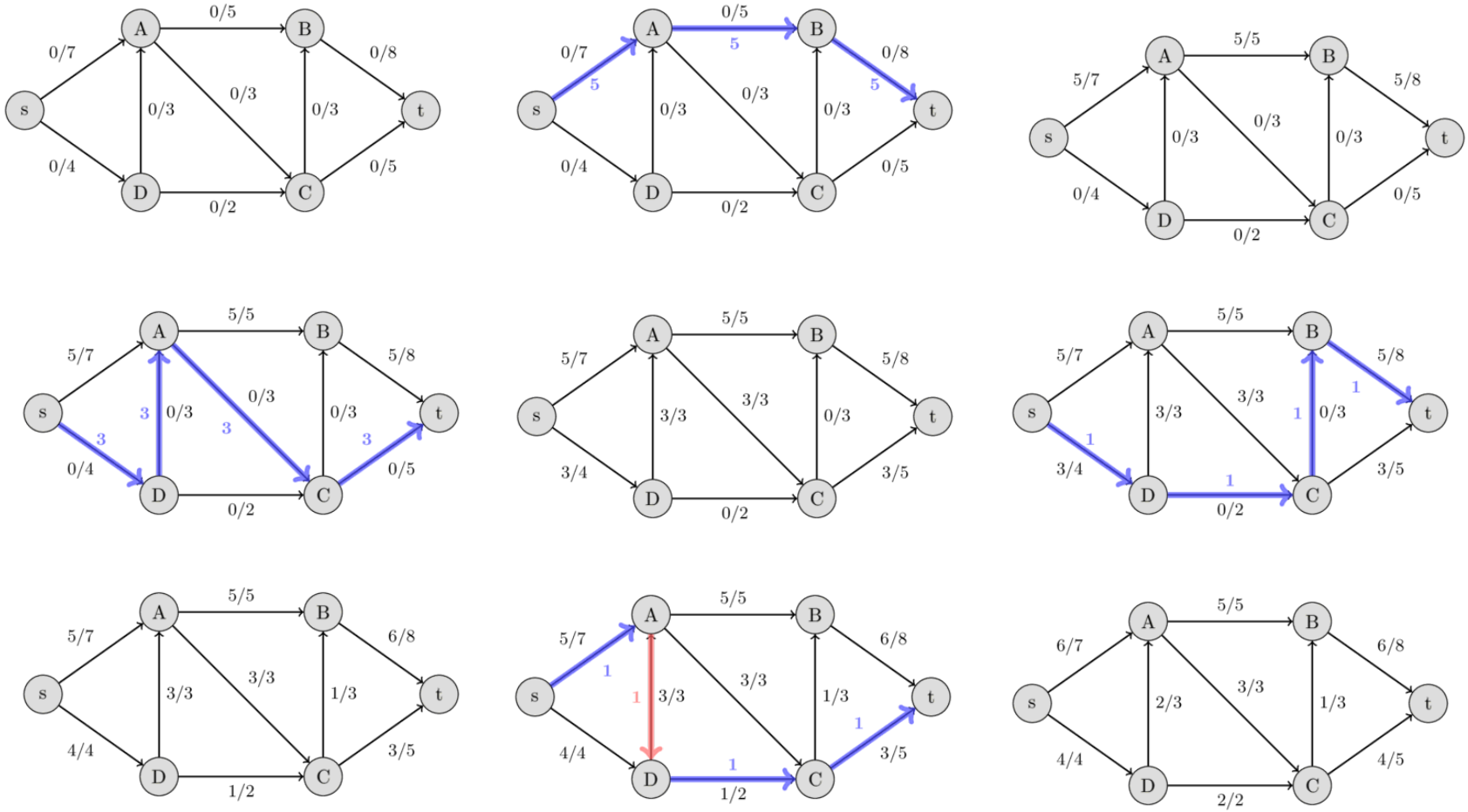
Ford-Fulkerson yöntemi

- *ÖRNEK:*
- Aşağıdaki resim akış ağındaki maksimum akışı göstermektedir:



Ford-Fulkerson yöntemi

- *ÖRNEK:*



10 is the maximal flow

Ford-Fulkerson yöntemi

birden fazla kaynak ve tüketici

Eğer problemde birden fazla kaynak ve tüketici verilirse konuyu yukarıda anlattığımız duruma kolaylıkla indirgeyebiliriz.

- Kaynaklar $s_1, s_2, \ldots, s_p$ ve tüketiciler $t_1, t_2, \ldots, t_q$ olsun.
- Grafiğe süper kaynak adı verilen yeni bir s köşe noktası ve süper tüketici olan yeni bir t köşe noktası ekliyoruz.
- Bu iki yeni köşeye ek olarak kenarları da ekliyoruz:

- (s, s_i), **such that** $c(s, s_i) = +\infty$, **for** $i = 1, 2, \ldots, p$.
- (t_j, t), **such that** $c(t_j, t) = +\infty$, **for** $j = 1, 2, \ldots, q$.

Ford-Fulkerson yöntemi

birden fazla kaynak ve tüketici

Ford-Fulkerson algoritmasını kullanarak, problemin orijinal versiyonunda aranana eşit olacak, s'den t'ye maksimum akışı bulabiliriz..

- Her köşeden geçen akışa bir kısıtlama getirerek sorunu daha da karmaşık hale getirmek mümkündür..

Let be given a function $v(i),\ v(i) \geq 0,\ i \in V$ such that

$$\sum_{j \in V} f(i,j) \leq v(i),\ i \in V$$

Ford-Fulkerson yöntemi

birden fazla kaynak ve tüketici

Kaynaklardan t tüketicisine kadar maksimum akışı arıyoruz. Yine sorunu standart maksimum akış arama problemine indirgeyerek çözebiliriz. Bu amaçla yeni bir G'(V',E') grafiği oluşturacağız:

- 1) her bir i∈V köşesinde, bir kenar (i_1, i_2) ile bağlanacak olan iki v’:i_1 ve i_2 köşesine karşılık gelir. Bu kenarın verimi $c(i_1, i_2)$ $v(i)$'ye eşit olacaktır.
- 2) G'deki her bir kenar (i, j), (i_2, j_1) kenarı olarak G'ye aktarılır.
- 3) Yeni G’ grafiğinde köşelerdeki akışı kısıtlayan bir v fonksiyonu olmayacaktır.
- Önceki paragrafta olduğu gibi Ford-Fulkerson algoritması tarafından bulunan G'deki maksimum akış, G grafiği için de maksimum olacaktır.

Co-funded by the Erasmus+ Programme of the European Union

Sorular ve alıştırmalar:

1. Maksimum akış problemi neleri içerir?
2. Bir ağın yalnızca bir kaynağı ve bir havuzu olabilir! **Doğru ya da yanlış** ?
3. Kaynak nedir?
4. Maksimum akış problemini çözmek için hangi algoritma kullanılır?
5. Maksimum akış problemini çözmek için Ford-Fulkerson algoritması kullanılıyor! Doğru ya da yanlış ?

Co-funded by the Erasmus+ Programme of the European Union

İlginiz için teşekkür ederiz!