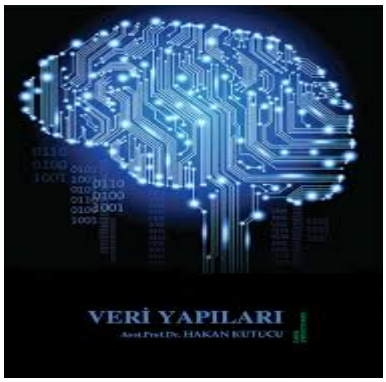
BİLGİSAYAR MÜHENDİSLİĞİ

VERİ YAPILARI

ÖDEV RAPORU



ÖDEV KONUSU: VERİ YAPILARININ GÜNCEL TEKNOLOJİLERDE KULLANIMI

HAZIRLAYAN

MEHMET EMRE DURUR

ÖDEV ÖZETİ

Bu çalışmada beş temel yapı/model incelenecektir: **Graf**, **Ağaç**, **İlişkisel Veri Modeli**,**Liste ve Bağlantılı liste veri yapıları.** Her bir yapı/modelin tanımı, kullanım alanları, avantajları,dezavantajları ve sınırlamaları ele alınacak; ardından bunlar arasında bir karşılaştırma yapılacaktır. Bu sayede veri organizasyonu ve modelleme konusundaki kavramsal altyapı netleşecektir.

GİRİŞ

Veri yapıları, verilerin bellekte nasıl düzenlendiğini tanımlar. Doğru veri yapısının seçimi, algoritmaların etkinliğini doğrudan etkiler. Örneğin, bir arama algoritmasının performansı, kullanılan veri yapısına bağlı olarak büyük ölçüde değişebilir.Veri yapıları bilgisayarların büyük ve karmaşık veri setlerini işleme yeteneğini artırır, bu da kodun etkinliğini ve anlaşılabilirliğini artırır.Dizi (array), bağlı liste (linked list),yığın (stack), kuyruk (queue), ağaç (tree), hash table veri yapıları örnekleridir.

Veri modelleri, verilerin mantıksal yapısını ve ilişkilerini tanımlar. Bu modeller, verinin nasıl organize edileceğini ve birbirleriyle nasıl ilişkilendirileceğini belirler. Veri modelleri, veriyi doğru bir şekilde temsil etmeye yardımcı olur, eksik verileri bulma da ve veri fazlalığını azaltma da önemli bir rol oynar.İlişkisel model (relational model), hiyerarşik model (hierarchical model),ağ (network) modelleri veri modellerine örnektir.

Sonuç olarak veri yapıları ve veri modelleri, yazılım sistemlerinin verimli, ölçeklenebilir ve sürdürülebilir olmasını sağlar. Bu kavramlar, yazılım mühendislerinin karmaşık veri problemlerini çözmelerine ve etkili çözümler geliştirmelerine olanak tanır.

GRAF VERİ MODELİ

Günümüzde birçok uygulama, yalnızca verilerin saklanmasını değil, veriler arasındaki ilişkilerin güçlü bir şekilde temsil edilmesini gerektirir. Özellikle sosyal ağlar, öneri sistemleri, yol bulma algoritmaları gibi alanlarda ilişkilerin derinliği ve çok katmanlı yapısı, geleneksel ilişkisel modellerin sınırlı kalmasına yol açmaktadır. Bu tür durumlarda, düğümler (varlıklar) ve kenarlar (ilişkiler) üzerinden çalışan graf veri modeli öne çıkmaktadır

KULLANIM ALANLARI:

Sosyal Ağlar: Facebook, Instagram ve Linkedln gibi ağlarda kullanıcılar düğüm,bağlantılar ise kenar olarak modellenir.Böylece “arkadaşın arkadaşı” gibi sorgular hızlıca çözülebilir.

Öneri Sistemleri: Netflix veya Amazon, kullanıcıların tercihlerini ve içerikler arasındaki ilişkileri graf yapısıyla göstererek kişiselleştirilmiş öneriler sunar.

Navigasyon Sistemleri: Google Maps gibi uygulamalarda konumlar düğüm, yollar ise kenar olarak tutulur ve en kısa yol problemleri algoritmalarla çözülür.

Siber Güvenlik: Ağ üzerindeki bağlantıların izlenmesi ve anormal trafiklerin tespiti için graf tabanlı ilişkisel analiz yapılır.

NEDEN TERCİH EDİLİR?

İlişki Odaklıdır: Karmaşık bağlantılar doğal olarak temsil edilir, JOIN işlemlerine gerek kalmaz.

Esnektir: Veri yapısı değiştikçe kolayca yeni düğüm veya kenar eklenebilir.

Karmaşık Sorgular İçin Etkilidir: “Bir kişinin üçüncü derece bağlantıları” gibi çok seviyeli sorgular performanslı şekilde yanıtlanır.

ALTERNATİFLERE GÖRE;

AVANTAJLARI:

1-) İlişkileri doğrudan modellediği için karmaşık bağlantılar doğal olarak temsil edilir

2-) Derin ve çok seviyeli sorgularda (örneğin: sosyal ağ analizi) yüksek performans gösterir.

3-) Yeni düğüm ve kenar eklemek kolaydır, bu nedenle esnektir.

DEZAVANTAJLARI:

1-) Basit, tablo tabanlı veriler için gereksiz karmaşıklık yaratabilir.

2-) Yüksek sayıda kenar olduğunda bellek tüketimi artar.

3-) SQL kadar yaygın standartlaşmaya sahip değildir.

ÖLÇEK VE PERFORMANS:

Graf tabanlı veri tabanları milyonlarca düğüm ve kenarı milisaniyeler içinde işleyebilir.

Zaman Karmaşıklığı:

En Kısa Yol —-> O(E+ V log V)
Grafik gezintisi (DFS/BFS) —> O(V+E)

E: Grafın kenar (edge) sayısı, yani düğümler arasındaki bağlantılar.
V: Grafın düğüm (node) sayısı.

O (E+ V log V) —--> Daha ağır işlemler
O (V+E) —--> Daha hızlı işlemler

Bellek: Kenar sayısı arttıkça bellek tüketimi artar. Bu nedenle büyük ölçekli sistemler de dağıtık graf veri tabanları kullanılır.

MİNİ DENEY ÇALIŞMASI:

AMAÇ: Arkadaş sisteminde graf ve ilişkisel modelin karşılaştırılması.

METODOLOJİ:
=> 100K kullanıcı ve 1M arkadaşlık ilişkisi içeren bir veri seti oluştur.

=> “Bir kullanıcının 3.derece arkadaşlarını getir” sorgusunu hem NEO4J hem SQL üzerinde çalıştır.

BEKLENEN SONUÇ: NEO4J gibi graf veri tabanı, çoklu JOIN işlemlerine kıyasla daha düşük gecikme ile sonuç döndürür.

NEO4J: En bilinen graf tabanlı veri tabanıdır.(DÜĞÜMLER + KENARLAR)

GRAF MODELİ —--> NEO4J

SQL: SQL (Structured Query Language), ilişkisel veri tabanlarının sorgulama dilidir. (TABLOLAR + JOIN)

İLİŞKİSEL MODEL —--->SQL (MYSQL,POSTGRESQL VB.)

AĞAÇ VERİ MODELİ

Verilerin hiyerarşik bir yapı içinde tutulması gerektiğinde ağaç veri modeli devreye girer. Ağaç yapısı; kök (root), dallar (branches) ve yapraklar (leaves) şeklinde düzenlenir. Bu yapı, özellikle ebeveyn-çocuk ilişkilerinin önemli olduğu durumlarda tercih edilir. İlişkisel modelde çoklu JOIN işlemleriyle kurulabilecek bu hiyerarşi, ağaç yapısında doğal olarak temsil edilir.

KULLANIM ALANLARI:

Dosya Sistemleri: İşletim sistemlerinde klasör-alt klasör-dosya yapısı ağaç şeklinde tutulur.

XML VE JSON VERİ YAPILARI: Verilerin hiyerarşik biçimde saklandığı bu formatların arka planında ağaç yapısı vardır.

Oyun Geliştirme: Karar ağaçları yapay zeka karakterlerin karar verme mekanizmalarında kullanılır.

Arama Algoritmaları: İkili arama ağaçları (BST), AVL ağaçları, B-ağaçları gibi yapılar hızlı veri erişiminde kritik rol oynar.

NEDEN TERCİH EDİLİR?

1-) Doğal olarak hiyerarşik yapıları temsil eder.

2-) Veri ekleme ve silme işlemleri, uygun ağaç yapısı seçildiğinde oldukça verimlidir.

3-) Veri erişiminde logaritmik zaman performansı sunabilir.

ALTERNATİFLERE GÖRE;

AVANTAJLARI:

1-) Hiyerarşik veri yapılarında en doğal çözüm.

2-) Bazı yapılarda arama ve ekleme O(log n) performansı sağlar.

3-) Uygulaması görece basittir.

DEZAVANTAJLARI:

1-) Çapraz ilişkiler veya çoklu ebevyn ilişkilerini modellemek zordur.

2-) Dengesiz ağaçlarda performans O(n)'e kadar düşebilir.

ÖLÇEK VE PERFORMANS:

Arama işlemleri (BST, AVL, Red-Black Tree): O(log n)
Ekleme/Silme işlemleri: O(log n) (dengeli ağaçlarda)
Kötü senaryo (dengesiz ağaç): O(n)
Büyük ölçekli veriler: Veri tabanlarında kullanılan B-Tree ve B+Tree yapıları milyonlarca kaydı logaritmik zamanda aramayı mümkün kılar.

İLİŞKİSEL VERİ MODELİ

İlişkisel veri modeli, 1970'te Edgar F.Codd tarafından tanımlanmış olup, verilerin satır-sütun yapısında tablolar (relation) halinde organize edilmesini sağlar. Her tablo bir varlığı temsil eder (örneğin öğrenciler,dersler), tablolar arasındaki bağlantılar ise birincil anahtar (primary key) ve yabancı anahtar (foreign key) ile kurulur. Bugün kullanılan SQL tabanlı veri tabanlarının temelini oluşturur.

KULLANIM ALANLARI:

Bankacılık Sistemleri: Müşteri hesapları, işlemler, kredi kayıtları ilişkisel veri tabanlarında saklanır.

E-Ticaret: Ürün-sipariş-kullanıcı-stok gibi veriler ilişkisel tablolarla yönetilir.

Kurumsal Uygulamalar: İnsan kaynakları, CRM, ERP sistemleri gibi çok boyutlu verileri yönetir.

Sağlık Sistemleri: Hasta kayıtları, laboratuvar sonuçları, randevular.

ALTERNATİFLERE GÖRE;

AVANTAJLARI:

1-) Stardartlanmış bir modeldir; SQL dünya çapında kabul gören dildir.

2-) Veri tutarlılığı ve bütünlük kısıtları güçlüdür.

3-) Karmaşık sorgular için esnek JOIN işlemleri yapılabilir.

4-) Büyük ölçekli, güvenlik ve denetim gerektiren kurumsal uygulamalarda güçlüdür.

DEZAVANTAJLARI:

1-) Büyük veri kümelerinde JOIN işlemleri pahalı hale gelir.

2-) Çok esnek olmayan veri şeması: Veride değişiklik için çoğu zaman tablo yapısını yeniden düzenlemek gerekir.

3-) Yoğun ilişkili (graph benzeri) verilerde performans düşer.

4-) Dağıtık sistemlerde (ör. bulut tabanlı NoSQL alternatiflerine kıyasla) ölçeklenebilirliği sınırlı olabilir.

ÖLÇEK VE PERFORMANS:

**Basit sorgular genellikle O(1) veya O(log n) indeksleme ile çözülür.

**Karmaşık sorgular (JOIN, nested query) çoğunlukla O(n log n) veya daha kötü karmaşıklığa çıkabilir.

**Performans, indeksleme stratejileri ve sorgu optimizasyonu ile doğrudan ilişkilidir.

**Yatay ölçekleme zordur; genellikle dikey ölçekleme (daha güçlü donanım) tercih edilir.

LİSTE VERİ YAPISI

Liste, aynı türden öğelerin sıralı biçimde saklandığı bir veri yapısıdır.Her elemana indeks üzerinden erişebilir. Programlama dillerinde array ya da dynamic array (python list, java arraylist) olarak uygulanır.

KULLANIM ALANLARI:

Arama Motorları: Kullanıcıya önerilen sorgular bir liste içinde tutulur.
Veritabanı İndeksleri: Sıralı depolama yapıları (B-tree yapılarında alt seviyelerde listeler).
Oyunlarda Sıralama Tabloları (Leaderboard): Skorların tutulması.
Uygulamalarda Görev Yönetimi: To-do list, alışveriş listesi uygulamaları.

ALTERNATİFLERE GÖRE;

AVANTAJLARI:

1-) İndeks üzerinden O(1) zaman karmaşıklığı ile erişim yapılabilir.

2-) Bellekte ardışık saklandığı için önbellek dostudur.

3-) Basit yapısı sayesinde kolay uygulanır.

DEZAVANTAJLARI:

1-) Ekleme / silme işlemleri O(n) zaman alır (ortadaki bir elemanı çıkarmak tüm elemanların kaydırılmasını gerektirir),

2-) Bellekte önceden ayrılmış sabit boyut (statik dizilerde).

3-) Dinamik dizilerde kapasite dolunca yeniden boyutlandırma maliyetidir.

ÖLÇEK VE PERFORMANS:

Erişim: O(1)
Arama: O(n)
Ekleme/Silme: O(n)
Büyük veri kümelerinde daha çok okuma ağırlıklı senaryolarda avantajlıdır.

BAĞLANTILI LİSTE VERİ YAPISI

Bağlantılı liste, her elemanın (node) değer ve bir sonraki elemanın adresini (pointer) sakladığı veri yapısıdır.

=>Tek yönlü bağlantılı liste (singly linked list): Her düğüm yalnızca bir sonraki düğümü işaret eder.

=>Çift yönlü bağlantılı liste (doubly linked list): Düğümler hem önceki hem sonraki düğümü işaret eder.

KULLANIM ALANLARI:

Tarayıcı Geçmişi: İleri–geri gezinme çift yönlü bağlantılı liste ile uygulanır.

Metin Editörleri: Satırlar arasında ekleme/silme işlemlerinde.

Bellek Yönetimi: İşletim sistemlerinde boş bellek bloklarının yönetimi.

Blockchain Öncesi Yapılar: Basit zincirleme veri yapıları bağlantılı liste mantığına benzer.

ALTERNATİFLERE GÖRE;

AVANTAJLARI:

1-) Dinamik Bellek Yönetimi: Eleman ekleme/silmede yeniden boyutlandırmaya gerek yok.

2-) Ortaya hızlı ekleme/silme: O(1) (Önce pointer ile doğru düğüme ulaşman gerek).

3-) Boyutu önceden bilmeye gerek yoktur.

DEZAVANTAJLARI:

1-) Erişim O(n) çünkü rastgele erişim yapılamaz.

2-) Pointer'lar ek bellek maliyeti yaratır.

3-) Önbellek dostu değildir, çünkü elemanlar bellekte dağınık saklanır.

ÖLÇEK VE PERFORMANS:

Erişim: O(n)
Arama: O(n)
Ekleme/Silme (bilinen noktada): O(1)
Büyük veri setlerinde yoğun ekleme/silme gereken senaryolarda avantajlıdır.

GENEL ÇIKARIMLARIM

Bu ödev raporunu hazırlamak için araştırırken edindiğim bilgiler kadarıyla bu veri yapılarına dair bazı çıkarımlarda bulunacağım. Bu çalışmada incelenen graf,ağaç, ilişkisel veri modeli, liste ve bağlı liste yapıları, bilgisayar bilimlerinin en temel veri modellerini temsil etmektedir. Liste ve bağlı liste gibi yapılar daha çok algoritma ve temel veri işleme görevlerinde kullanılırken, ağaç yapıları ve ilişkisel model özellikle veri tabanı yönetimi ve arama / indeksleme gibi büyük ölçekli problemlerde halen etkinliğini korumaktadır. Graf yapıları ise karmaşık ilişkilerin modellenmesi gereken alanlarda günümüz teknolojilerinde kritik bir konumda bulunmaktadırlar.

Fakat bu konumları, edinilen veri büyüklükleri ve çeşitlilikleri arttığı için eskiden olan veri yapıları gibi eskilemelerine, tek başlarına yeterli olmamalarına yol açıyor. LSM Tree yapısı, büyük yazma ağırlıklı NoSQL veri tabanlarında yüksek performans sağlamaktadır. Merkle Tree yapısı, blockchain teknolojilerinde veri bütünlüğünü ve güvenirliliğini garanti altına almak için kullanılmaktadır. B+ Tree ise ilişkisel veri tabanlarında indeksleme için halen standarttır ve gelecekte de kullanılmaya devam edecektir.

Genel olarak değerlendirdiğimde, veri yapıları yazılım da değişmez yapı taşları olmaya devam ederken, gelecek eğilimler daha karmaşık, yüksek performanslı bir şekilde ilerlemektedir. Bu durum, bu sektörün bir temelin üzerine inşa edilen yeni veri yapılarının ihtiyaç duyulduğunu gösteriyor.

KAYNAKÇA:

=> Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms (CLRS). MIT Press.*

=> Selçuk Alp. *Veri Yapıları ve Algoritmalar*. Türkmen Kitabevi, 2016.

=> https://codigno.com/veri-yapilari-nelerdir-2023/

=> https://tr.wikipedia.org/wiki/Veri_yap%C4%B1s%C4%B1

=> https://youtu.be/mKPTA7hAewY?si=BzQjn1E64KJiRMqY

=> https://youtu.be/r4SJ7QnOFqE?si=bd5CbMy3uvo7Am1U