**DEÜ MÜHENDİSLİK FAKÜLTESİ**
**MÜHENDİSLİK BİLİMLERİ DERGİSİ**

**Cilt: 16 No: 1 Sayı: 46 sh. 43-57 Ocak 2014**

# BELLEK YÖNETİMİNDE SAYFA DEĞİŞİM ALGORİTMALARININ PERFORMANS ANALİZİ

## *(COMPARISON OF THE PAGE REPLACEMENT ALGORITHMS FOR MEMORY MANAGEMENT)*

**Ünal ÇAVUŞOĞLU[1], Ahmet ZENGİN[2]**

## ÖZET/*ABSTRACT*

Bu makalede veri tabanı yönetim sistemlerinde önemli bir yere sahip olan bellek yönetim teknikleri ele alınmıştır. Veri tabanı yönetim sistemleri üzerinde gerçekleşen işlemlerin hız ve performans değerleri, kullanılan bellek yönetim algoritması tarafından büyük ölçüde etkilenmektedir. Bu çalışmada bellek yönetim algoritmaları, tampon boyutu, blok sayısı ve olasılıksal değerler gibi farklı parametler kullanılarak test edilmiştir. Sonuçlar üzerinden bellek yönetim algoritmalarının performans analizleri gerçekleştirilmiştir. Bellek yönetim algoritmalarından, temel bellek yönetim algoritması, FIFO, LRU, LRM ,CLOCK algoritmaları oluşturulan simülasyon üzerinde test edilmiştir. Simülasyon sonuçları değerlendirildiğinde LRU ve CLOCK algoritmalarının diğer test edilen algoritmalardan daha başarılı sonuçlar elde ettiği tespit edilmiştir.

*In this article, memory management techniques which are very essential in database management system are discussed. The speed and performance values of transactions that occur on database management systems are greatly effected by buffer management algorithm. Buffer management algorithms are evaluated by creating parameters as different buffer size, block number and probabilistic values. Tests are performed and the results are discussed performance analysis of algorithms. Buffer management algorithms such as Basic, LRU, Clock and LRM were compared. As a result of the LRU and Clock algorithms are produced more successful results than other algorithms.*

## ANAHTAR KELİMELER/*KEYWORDS*

Bellek değişim algoritmaları , FIFO, LRU, Clock, LRM
*Buffer replacement algorithms, FIFO, LRU, Clock, LRM*

[1] Sakarya Ün., Teknoloji Fak., Bilgisayar Müh. Böl., ADAPAZARI, e-posta: unalc@sakarya.edu.tr
[2] Sakarya Ün., Teknoloji Fak., Bilgisayar Müh. Böl., ADAPAZARI, e-posta: azengin@sakarya.edu.tr

## 1. GİRİŞ

Sistemler üzerinde gerçekleşen işlem hacimlerinin büyümesi sonucu veri tabanları üzerinde bulunan veri miktarları oldukça artmış, aşırı miktardaki veriler yapılan işlemlerin performanslarını ciddi şekilde etkilemiş ve bu verilerin yönetimi, işlenmesi başlı başına ele alınması gereken çok ciddi bir konu haline gelmiştir. Bu sistemlerden beklenen, depolanacak olan verilerin efektif bir şekilde saklanması ve depolanan veriler içerisinden ihtiyaç duyulan bilgilerin hızlı ve doğru bir şekilde elde edilecek şekilde bir yapıya sahip olmalarıdır. Veri tabanı yönetim sistemleri çok büyük miktarlardaki verileri depolama aygıtları üzerinde saklamakta, ihtiyaç duyulan veriyi diskler üzerinden alarak, belleğe ve oradanda işlemlerin gerçekleşmesi için işlemci üzerine geçirilmesi işlemlerini gerçekleştirmektedir. Bu işlemi gerçekleştirirken işletim sistemi ile koordineli olarak işlem görmektedirler. İşletim sistemi tarafından veri tabanı yönetim sistemine belli miktarlarda kaynak tahsisi yapılmaktadır (DBMS Bellek Dizaynı, 2011). Bu bağlamda veri tabanı yönetim sisteminin görevlerinden bir tanesi kendisine tahsis edilmiş olan kaynakları en etkili bir biçimde kullanmak ve yönetmektir. Disk ile bellek arasındaki bu ilişkide kullanılacak olan bilgilerin bellek üzerinde bulunması veri tabanı yönetim sistemlerinin işlemlerinin gerçekleştirilmesini çok büyük oranda etkilemektedir. Kullanılacak olan ve ihtiyaç duyulan verilerin bellek üzerinde tutulmasını sağlamak maksadıyla bellek üzerinde bulunan tampon bölgesi kullanılmakta, tampon bölgesi üzerinde tutulacak olan verilerin belirlenmesi için tampon değişim algoritmaları kullanılmaktadır. Bu çalışmada tampon değişim algoritmalarından 5 adet temel algoritma incelenerek, bu algoritmaların başarımlarını test etmek için simülasyon üzerinde testleri gerçekleştirilmiştir. Elde edilen test sonuçlarına göre performans analizleri gerçekleştirilmiştir.

Birinci bölümde girişin ardından, 2. Bölümde bu çalışma için temel teşkil eden konular ile ilgili bilgelendirme yapılmış, 3. bölümde karşılaştırma ve değerlendirme için gerekli testler gerçekleştirilmiş, 4. bölümde test ve karşılaştırma sonuçları verilerek bu sonuçlar üzerinde çıkarımlar yapılmış, 5. Bölümde ise tampon değişim algoritmaları ile ilgili sonuç ve değerlendirmelerde bulunulmuştur.

## 2. GENEL BİLGİLER

### 2.1. Bellek Yönetimi ve Tampon Bölgesi

Bilgisayar sistemleri üzerindeki donanımsal yapılarda, disk kapasiteleri bellek kapasitelerine göre oldukça büyük boyutlardadır. Bilgisayar üzerinde çalışmakta olan uygulamaya ait programsal yapıların çalışması ve sonuç üretmesi için kullanılacak olan verinin disk üzerinden belleğe oradan da işlemciye getirilmesi gerekmektedir. Programların çalışması ve işlemlerin gerçekleşmesi için ihtiyaç duyulan veri öncelikle ön bellek üzerinde, daha sonra da ana bellek üzerinde olup olmadığı kontrol edilir, eğer ihtiyaç duyulan veriye ulaşılamazsa veri disk üzerinde demektir. Diskin genel yapısına bakıldığında, sabit disk bilgisayardaki bilgileri saklayan ve bilgisayar kapandığı zaman bu bilgileri kaybetmeyen donanımdır. Bilgisayardaki bütün veri sabit disk üzerindedir. Bilgisayar açılırken gereken veriler ana bellek (RAM) üzerine alınır, ve bilgisayar bundan sonra ihtiyaç duyduğu bilgiye sabit disk yerine ana bellekten ulaşır. Çalıştırdığımız her program, çalışması için gereken veriyi belleğe taşır.

Sabit disk yapıları plakalardan oluşmaktadır. Her bir plaka izlerden ve her bir iz de sektörlerden oluşmaktadır. İşletim sistemleri, sektörleri gruplayarak onları küme denen yapılar halinde topluca işlerler. Okuma/yazma kafalarını bir izden başka bir ize taşımak maliyetli olduğundan, bir iz dolduğu zaman, aynı plaka üzerindeki başka bir iz yerine, aynı

silindir üzerindeki başka bir iz kullanılır. Böylece her bir iz dolduktan sonra değil, her bir silindir dolduktan sonra okuma/yazma kafası hareket ettirilir. İşletim sistemi diske erişim, veri transferi, veri yerleşimi, sektör büyüklüğü gibi disk detaylarını saklar, tutar ve bu işlemler için blok arayüzünü kullanır. Byte dizisinin büyüklüğü işletim sistemi tarafından belirlenmektedir. Bütün disk üzerinde sabit büyüklüğe sahiptir. Ana bellek ile sabit disk arasında transfer edilebilen minimum veri miktarıdır. Blok içeriğinde tutulan veriye disk üzerinde iken erişmek mümkün değildir. Veriye erişim için verinin ana belleğe getirilmesi gerekmektedir. Ana bellekteki bir blok için ayrılan bölgeye sayfa denir (Jung vd., 2009). Veri tabanı yönetim sistemi üzerinde kullanılacak olan bilginin diskten alınarak işlemciye getirilmesi erişim süresi olarak tanımlanır ve Eşitlik 1 ile ifade edilmektedir (Wang , 2001). Toplam disk erişimi zamanaşağıdaki şekilde hesaplanmaktadır:

$$T_{\text{erişim zamanı}} = T_{\text{arama}} + T_{\text{dönme}} + T_{\text{transfer}} \quad (1)$$

$T_{\text{arama}}$ : Disk üzerindeki kafanın iz üzerinde uygun pozisyona gelmesi ve aranılan verinin bulunması için geçen süredir.
$T_{\text{dönme}}$: Hedef sektöre ulaşım için disk üzerinde bulunan okuma işlemini gerçekleştiren kafanın dönme işlemi sırasındaki gecikme süresidir.
$T_{\text{transfer}}$ : Okuma ve yazma işlemlerinin yapılması sırasında geçen zaman birimidir.

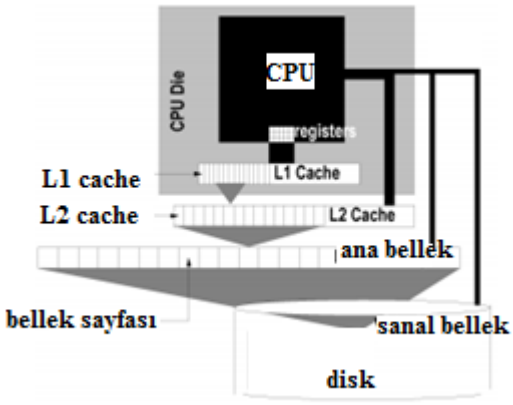
İşletim sistemi veri tabanı yönetim sistemi üzerinde işlemlerin gerçekleştirilmesi için ana belleğin belirli bir kısmının kontrolünü veri tabanı yönetim sisteminin kontrolüne bırakmaktadır. Sistem üzerinde işlemlerin gerçekleşmesinde hız ve güvenilirlik esas tutulmaktadır. İşlemlerin gecikmesine sebep olan en büyük etkenlerden birisi, diskten ilgili verinin aranıp bulunarak belleğe oradan da işlemci üzerine aktarımı sırasında gerçekleşmektedir. Diskin donanımsal yapısı ve diğer donanımsal kısıtlar işlem süresini olumsuz yönde etkilemektedir. İşte bu noktada tampon bölge kullanımı devreye girmektedir. Ana bellek üzerinde sabit sayıda tampondan oluşan bölgeye tampon havuzu denilmektedir (Sacco ve Schkolnik, 1982). Tampon havuzunda bulunan her tampon üzerinde bir disk bloğu bilgisi tutulmaktadır. Ayrıca kısıtlı olan ana bellek bölgesinin yetersiz kaldığı durumlarda disk üzerinde bir bölge ayrılarak ana bellek gibi kullanılabilmektedir. Ayrılmış olan bu bölgeye de sanal bellek adı verilmektedir (Peter, 1970).

İşletim sistemi tarafından tahsis edilen bu tampon bölgesi veri tabanı yönetim sisteminin kontrolüne verilmektedir. Tampon bölgesinin kullanımı ve yönetiminin maksadı, kullanılacak olan veriye erişimde zaman kaybına sebep olan disk erişimini azaltmak ve sanal belleğe olan ihtiyacı ortadan kaldırmaktır (Sciore, 2007).

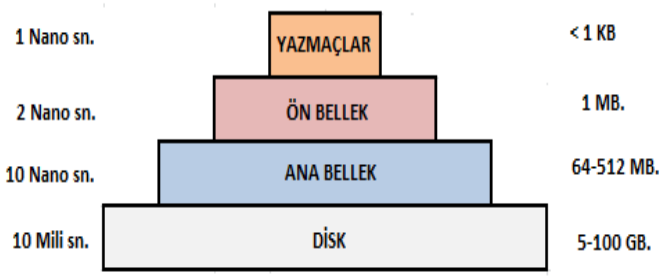
Tampon bölgesinin efektif kullanımı sonucunda veri tabanı üzerinde işlem görecek olan verilerin bu bölgede bulunma ihtimali artacak ve işlem sürelerini ciddi oranda hızlanmasına sebep olacaktır.

Şekil 1'de disk, ana bellek, ön bellek ve yazmaç birimlerinin donanımsal olarak yapısı görülmektedir. İşlemci üzerinde bulunan yazmaçlar üzerinde gerçekleşen işlemler sonucu verilerin işlenmesi gerçekleşmektedir. Ayrıca işlemci üzerinde yazmaçlar ile direk olarak bağlı olan L1 ön bellek ve onun ile bağlı olan L2 ön bellek bulunmaktadır. Ön bellek üzerinde bulunan verilerin işlemleri çok daha hızlı şekilde gerçekleştirilmektedir. Donanımsal yapı incelendiğinde L2 ön bellek, ana bellek ve disk üzerinde bulunan takas alanı, sanal bellek ile işlemci arasında veri yolları ile aktarım işlemi yapılmaktadır. Donanımsal yapılar üzerinde işlemlerin gerçekleşmesini gösteren şekil 2 incelediğinde, kullanılacak olan verinin disk üzerinde veya bellek üzerinde bulunması durumunda zamansal olarak çok ciddi fark olduğu görülmektedir. Yapıların kapasitelerine bakıldığında sabit disk en geniş kapsamlı ve bilgisayar kapandığında verilerin silinmeden üzerinde tutulduğu yapıdır. Disk üzerinde

bulunan bir veriyi erişim için diğer yapılara göre oldukça uzun bir zamana ihtiyaç vardır. Ana bellek işlemci üzerinde işlem görecek olan verilerin işlemciye geçişinden önce disk üzerinden alınarak yerleşiminin yapıldığı bölgedir. Boyutlara sabit disk boyutlarına oldukça küçük ve ana bellek üzerindeki verilerin işlenme zamanı diske oranla çok daha hızlı ön belleğe göre düşüktür. Ön bellek ve yazmaçlar kapasite olarak çok daha küçük boyutlarda yapılardır. işlemci ile direk olarak bağlantı ve aktarıma sahip olduklarından dolayı bu donanımlar üzerinde işlemler çok hızlı gerçekleşmektedir. Piramitsel yapıda genel olarak yukarıya doğru çıkıldıkça boyut küçülmekte ve işlemci üzerinde işlem görme hızı çok ciddi oranda azalmaktadır.




Şekil 1. Bilgisayar üzerindeki donanımsal yapı (Stefan, 2002)




Şekil 2. Donanımsal yapılar ve işlem süreleri

## 2.2. Tampon Sayfa Değişim Algoritmaları

Dinamik bellek yapısını kullanan sistemlerde, ciddi problemlerden biri bellek üzerindeki blok yapılarının yerleştirileceği bölgenin tespit edilmesidir. Bellek sayfalarına verilerin yerleştirilmesi ve çıkarılması esnasında, karar verecek olan yapı sistemde kullanılan sayfa değişim algoritmasıdır. İşlemci tarafından işlem görecek olan verinin tampon bölgesinde bulunması işlem süresini ve performansı artıracağından dolayı kullanılan algoritmanın seçimi de ciddi anlamda önemlidir.

Algoritmaların performans değerlendirilmesinin yapılmasında tampon bölgesinde aranılan sayfanın bulunup bulunulmadığı baz alınarak değerlendirmeler yapılabilir. Algoritmalardan beklenen; ihtiyaç duyulacak olan sayfaların tampon ve bellek bölgesinde bulundurulması, kullanılma ihtimali olmayan sayfaların bellekten çıkarılarak, tampon bölgesinin efektif bir şekilde kullanımının sağlanmasıdır. Programların çalıştırılma esnasında ihtiyaç duydukları sayfa referansları baz alınarak bir sonraki çalışmada hangi sayfalara ihtiyaç duyacağı tahmini yapılabilir. Optimal sayfa yerleşim algoritmaları programların bir sonraki zaman dilimlerinde

ihtiyaç duyacakları sayfaların bellek üzerinde bulunması için tahminlerde bulunurlar. Algoritmaların çalışması esnasında iki durum söz konusudur. Bunlardan birisi tampon yapısında işlem görecek olan bloklar sırasıyla geldikleri esnada tampon üzerinde yerleştirildikleri yerler o bloklara tahsis edilir. Diğer durumda ise kullanılan algoritmanın çalışma mantığına uygun olarak tampon bölgesinde yapılan arama işlemi sonucu işlem görmüş olan veri tampon bölgesinden çıkarılır. Algoritmaların çalışmaları da işte bu bölgelerin tahsis edilmesi veya boşaltılacak olan bölgenin seçimine göre farklılık göstermektedir. Aşağıda bu algoritmalardan bazıları açıklanmıştır. Algoritmaların açıklanması sırasında Şekil 3'teki senaryo kullanılmıştır. Aşağıda belirtilen 10 adet pin/unpin operasyonu (**pin**: tampona yerleştirme, **unpin**: tampondan çekme) senaryo gereği aşağıdaki 10 adet operasyondan sonra tamponun son durumu Şekil 3'te verilmiştir.

**1-) Pin(10) 2-) pin (20) 3-) Pin(30) 4-) Pin(40) 5-) Unpin (10)**

**6- ) Pin (50) 7- ) Unpin (40) 8- ) Unpin (10) 9- ) Unpin (30)**

**10- ) Unpin (50) 11-) pin (60) 12-) pin (70)**

| Buffer: | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| block# | 10 | 50 | 30 | 40 |
| time read in | 1 | 6 | 3 | 4 |
| time unpinned | 8 | 10 | 9 | 7 |

Şekil 3. Uygulanan operasyonlardan sonra tamponların son durumu (Sciore, 2007)

Senaryoda bulunan 4 adet tamponun son durumları bu şekilde iken; gelen **11 ve 12. istek olarak pin (60) ve pin (70)** isteklerinin tampon bölgelerine yerleştirilmesi sırasında aşağıdaki algoritmalar üzerinde ne şekilde işlem gördüğü ve hangi tampon bölgelerine yerleşimlerinin gerçekleştirildiği ilgili algoritma maddesinde incelenecektir. Algoritmaların bu senaryo üzerindeki yerleşimleri değerlendirilirken 10. istekten sonra gelen 11. ve 12. istekler değerlendirilerek birbirinden bağımsız olarak yerdeğiştirme gerçekleştirilecektir.

1. **Temel sayfa değişim algoritması (basic):** En basit sayfa değiştirme algoritmasıdır. Algoritmanın çalışması yapısı incelendiğinde tampon bölgesi üzerine yerleştirilecek olan blokların yerleşimi esnasında tampon bölgesinin başından itibaren arama başlatılarak ilk boş bulunan bölgeye yerleşim gerçekleştirilmektedir. Algoritmanın çalışma prensibine göre gelen pin 60 ve pin 70 isteklerinin en baştan tampon bölgelerini taramaya başlayarak ilk bulduğu unpin edilmiş bölgeye yerleştirdiği için pin 60 isteğini buffer 0 bölgesine ve pin 70 isteğini buffer 1 bölgesine yerleşimini gerçekleştirecektir.
2. **Rastgele sayfa değişim algoritması (random):** Diğer algoritma yöntemleri programların çalışması esnasındaki verileri kullanılarak programların çalışması için gerekli olacak olan verileri tahmin ederek çalışırlar. Fakat rastgele sayfa yerleşim algoritmasında böyle bir durum söz konusu değildir. Random sayfa algoritması uygulama açısından bakıldığından kolay bir yapıdadır. Rastgele zamanlar içerisinde sayfa yenileme işlemini gerçekleştirirerek ihtiyaç duyulan sayfaların bellekte tutulmasına çalışır. Rastgele olarak değiştirilen veya bırakılan sayfaların kullanımı durumunda bu algoritmadan iyi sonuçlar elde edilebilir. Random sayfa

yerdeğiştirme algoritmasından yerdeğiştirecek olan sayfa random olarak seçildiği için bir tahminde bulunmak mümkün değildir.

3. **İlk giren ilk çıkar algoritması (FIFO):** FIFO algoritması işletmelerde de yoğun olarak kendisine uygulama alanı bulmuş ve yaygın olarak kullanılan bir yapıdır. Tampon bölgesine bir blok yerleşimi gerçekleştirileceği zaman FIFO algoritması tampon üzerinde ilk olarak tahsis edilmiş alanları tespit eder. Yerleştirilecek olan blok sayısına göre ilk olarak yerleşen bloktan itibaren ihtiyaç duyulan miktar kadar blok yapısı yer boşaltarak yeni gelen blokların boşaltılan yerlere yerleşimini gerçekleştirir. Işletmelerde çok kullanışlı bir yapı olarak kullanılmasına ragmen bilgisayar sistemlerinde aynı performansı sağladığı söylenemez. FIFO algoritmasına göre gelen pin 60 ve pin 70 istekleri şekil 3 te göründüğü gibi ilk olarak pin edilmiş bölgeler olan 0 ve 2 bölgeleridir. Bu bölgeler gelen pin 60 isteği buffer 0 bölgesine ve pin 70 isteği buffer 2 bölgesine yerleşimleri gerçekleştirilir.
4. **En son kullanılan sayfa algoritması (LRU):** LRU algoritması tampon bölgesi üzerinde yerleştirme yapacağı zaman tampon bölgesinde arama işlemi gerçekleştirerek en eski boşaltılmış olan yerden itibaren başlayarak ihityaç duyulan miktar kadar ileri doğru gelerek yer değiştirme işlemini icra eder. Bu algoritmada blokların daha önceki yerleştirilme sırası değil, bloklar üzerindeki işlemler sonucu boşaltılma değeri kriter alınarak işlemler gerçekleştirilir. LRU algoritması senaryo üzerindeki gelen 11 ve 12. istekleri en eski unpin edilmiş alan olan buffer 3 (unpin 7) ve buffer 0 (unpin 8) olan bölgeler üzerine yerleştirmektedir. 7 ve 8. sırada gelmiş olan istekler en eski unpin işlemleridir. Bu durumda pin (60) isteği buffer 3, pin (70) isteği buffer 0 bölgesine yerleştirilecektir.
5. **Saat yerleşim algoritması (Clock):** Clock algoritması tampon bölgesi üzerinde saat önünde bir çevrim gerçekleştirerek işlemlerini gerçekleştirir. Saat çevrimi yönünde arama işlemine başlanarak en son yer değiştirme işlemi gerçekleştirmiş olan buffer tespit edilir ve bu tespit edilen bloktan sonraki ilk tahsis edilmemiş olan blok üzerine yerleşim gerçekleştirilir. Yerleşimi gerçekleştirilecek olan her blok için işlem tekrarlanır. Clock algoritması senaryo üzerinde gelen 11 ve 12 nolu istekler için ilk once en son yerdeğiştirme işlemi gerçekleştirmiş olan bölgeyi tespit eder bu bölge buffer 1 bölgesidir. Bu bölgenin tespitinden sonra unpin durumdaki bölge üzerine yerleştirme işlemini gerçekleştirir. Bu durumda gelen pin 60 isteği buffer 2 ve pin(70) isteği buffer 3 bölgesine yerleştirilecektir.
6. **LRM algoritması (least recently made):** LRM algoritması tampon bölgesinde sayfa yerdeğiştirme işlemini gerçekleştirirken tamponda bulunan blokların en eski yerleşim zamanından itibaren en az kullanılanları (okuma ve yazma işlemini gerçekleştirenleri) tespit ederek yeni gelen blokları tespit edilenler ile yer değiştirirek işlemleri gerçekleştirir.

### 2.3. Simpledb İlişkisel Veri Tabanı

Simpledb ilişkisel veri tabanı üzerinde gerçekleştirilen simülasyonlar ayrık olay tanımlı olarak gerçekleştirilmiştir (Simpledb, 2007). Veri tabanı yönetim sistemleri üzerinde gerçekleşen uygulamaların anlaşılması ve uygulamalar geliştirilmesi uygun yapıda olan ilişkisel ve modüler yapıda kurulmuş olan bir sistemin varlığını gerektirmektedir. Simpledb, java programlama dili ile geliştirilen tamamen işlevsel yapıda veritabanlarının ilişkisel iç işleyişi ile eğitim amaçlı kullanılan bir veritabanıdır (Sciore, 2007). Çok kullanıcılı aksiyon ile birlikte, jdbc arayüzü ile iletişim fonksiyonel olması, mini-kit sql/tampon yönetimi, eşzamanlılık sistemi, sabit disk ve basit modülleri içeren temel kontrol ve sorgu optimizasyonu, algoritmalar bahsedilen işlevleri kapsamında basit ve fonksiyonel modülleri içerir. Her modül yeni algoritma geliştirme sağlayan bir altyapı sağlar.

Makalede simpledb üzerindeki modüller ve özellikle tampon yönetim modülü kullanılarak, tampon bölgesi sayfa değiştirme algoritmaları gerçeklenmiştir. Simpledb ilişkisel ve eğitimsel veritabanı E.Sciore tarafından geliştirilmiştir. Java programlama dilinde yazılmış 3500 satır kod, 85 adet sınıf ve 12 paketten oluşan bir yapıya sahiptir.

Simpledb ilişkisel veri tabanı yönetim sistemi yapısında aşağıdaki modüller bulunmaktadır.

**Remote:** kullanıcıdan gelen JDBC isteklerini karşılar.
**Planner:** SQL ifadesi için işleme planı oluşturur ve karşılık gelen ilişkisel cebir ifadesini oluşturur.
**Parse:** SQL ifadesindeki tablo, nitelik ve ifadeleri ayrıştırır.
**Query:** Algebra ile ifade edilen sorguları gerçekler.
**Metadata:** Tablolara ait katalog bilgilerini organize eder.
**Record:** Disk sayfalarına yazma/okumayı kayıt seviyesinde gerçekleştirir.
**Transaction&Recovery:** Eşzamanlılık için gerekli olan disk sayfa erişimi kısıtlamalarını organize eder ve veri kurtarma için kayıt defteri (log) dosyalarına bilgi girer.
**Buffer:** En sık/son erişilen disk sayfalarını ana hafıza tampon bölgesinde tutmak için gerekli işlemleri yapar.
**Log:** Kayıt defterine bilgi yazılmasını ve taranması işlemlerini düzenler.
**File:** Dosya blokları ile ana hafıza sayfaları arasında bilgi transferini organize eder.

### 2.4. İlgili Çalışmalar

Veri tabanı yönetim sistemlerinin yönetiminde tampon bölgesinin yönetimi ile ilgili günümüze kadar bir çok çalışma gerçekleştirilmiştir (Chou vd., 1986; Sacco vd., 1986; Lily, 2001; Wang ve Bunt, 2000; Daula vd., 2012). Bazı çalışmaların içerikleri ile ilgili bilgi aşağıda verilmiştir.

Chou ve arkadaşları yaptıkları çalışmada, tampon bölgesinin yönetimi için DBMIN adını verdikleri yeni bir algoritma tasarlamışlardır. Tasarlanan bu yeni algoritma ilişkisel very tabanı sorgulama için yeni bir model önermekte ve QLSM adı verilen yeni bir sorgu modelinden oluşmaktadır. Çalışmada diğer algoritmalar ile önerilen yeni algoritmanın karşılaştırılması yapılmıştır. Testler için farklı sorgu sınıfları oluşturulmuş ve bu sınıfların donanımsal gereksinimleri sınıflandırılmış, sorgu sınıflarının donanımsal kullanım oranları test edilmiştir. Tasarlanan algoritmanın çok kullanıcılı sistemlerde ve dağıtık simülasyon modellerinde, kıyaslama yapılan diğer algoritmalara göre daha başırılı sonuçlar ürettiği tespit edilmiştir.

Lily gerçekleştirdikleri veri tabanı yönetim sistemlerinde değişen tampon bellek bölgesi ihtiyacına göre otomatik olarak ayrılan bölgenin değişmesinin performans açısından büyük önem taşıdığı ifade edilmiştir (Lily, 2001). Uygulamalar için bölgedede tahsis edilecek olan alanın tahmin edilmesinin performansı ciddi oranda artıracak bir kriter olduğu vurgulanmıştır. Tezde gclock algoritması için markov zincir modelinde, tampon havuzunun hit oranının tahmini için bir model oluşturulmuştur.

W. Wang tarafından yapılmış olan çalışmada veri tabanı yönetim sistemlerinin kontrol ve yönetimi incelenirken, fiziksel bellek üzerindeki tampon bölgesi yönetimi ele alınmıştır (Wang, 2001). Tampon bölgesi yönetiminde kullanılan temel algoritmalar ve daha karmaşık yapıda algoritmalar ve çalışma prensipleri açıklanmış burada kullanılan algoritmaların veri tabanı performansının artırılmasındaki önemine vurgu yapılmıştır. Bu algoritmalardan LRU ve CLOCK algoritmalarının popüler olarak kullanıldığı ve ihtiyaç duyulan blok yapılarını tampon bölgesinde tutma oranlarının başarılı olduğu belirtilmiştir.

Ravazi tarafından yayınlanan makalede 1975-2000 yıllları arasında tampon yönetimi konusunda gerçekleştirilen çalışmaları genel olarak değerlendirmektedir. Özellikle donanımsal yapıların değişmesi ile ortaya çıkan problemler açıklanmış, son on yılda bellek ve tampon yönetiminin bir darboğaza girdiği konusuna vurgu yapılmış, problemler ve muhtemel çözüm yolları üzerinde açıklamalar gerçekleştirilmiştir.

W.Effelsberg ve T. Haerder tarafindan 1984 yılında yayinlananveri tabanı yönetiminde tampon bölgesi yönetim prensipleri adlı makalede, tampon bölgesi yönetim temelleri detaylı bir şekilde ele alinmiş, kullanilan algoritmalar açıklanmış ve LRU, FIFO, CLOCK, DGCLOCK algoritmalarinin gerçekleştirilen simülasyonlar sonucu başarım oranları üzerinde incelemeler gerçekleştirilmiştir (Effelsberg ve Haerder, 1984). Yapılan bu çalışmada testler ile eşleşen LRU algoritmasının başarılı sonuçlar ürettiği görülmüştür.

Literatürde bellek yönetiminin gerçekleştirilmesi için geliştirilen algortitmalar ile ilgili bir çok çalışma bulunmaktadır. LRU, LFU, LRFU, Gclock, LRU-K, EELRU, LRFU, ILRU ve OLRU algoritmalarıyla ilgili çalışmalardan bazılarıdır (Schoening, 1998; Teng ve Gumaer, 1984; Willick vd., 1990; Lee vd., 1999; Haas vd., 1990; Nicola vd., 1992; O'Neil vd., 1993; Smaragdakis vd., 1999; Lee vd., 1999; Sacco, 1987).

## 3. TAMPON YÖNETİM ALGORİTMALARI KARŞILAŞTIRMA TESTLERİ

Veri tabanı yönetim sistemleri üzerinde tampon bölgesinin yönetiminde kullanılan algortimaların performans karşılaştırması için Simpledb ilişkisel veri tabanı üzerinde tampon yönetim modülü kullanılarak algoritmalar gerçeklenmiştir. Tampon yönetim modülü üzerinde tanımlanan algortimaların gerçeklenmesi için ayrık olay simülasyonu kullanılarak test ortamı hazırlanmıştır. Ayrık olay simülasyonun üretmiş olduğu blokların işlem görmesi sağlanmıştır. İstek üreticisinin oluşturmuş olduğu bloklara kullanılan algortimaya göre, tampon bölgesinde alan tahsis edilmiş veya tampon üzerinde bulunan veriler tampon üzerinden alınmıştır. Tasarımı yapılan sistem üzerinde kullanılan algortimaların başarım oranları tespit edilmiştir. Sistemin ihtiyaç duyduğu blokların tampon bölgesi üzerinde bulunması üzerinden değerlendirme yapılarak başarım oranları tespit edilmiştir. Testler sırasında disk üzerinden yapılan okuma ve yazma işlemleri üzerinde değişik parametreler oluşturularak farklı durumlar için başarım oranları karşılaştırılmıştır.

### 3.1. Simülasyon Parametreleri ve Test Aşaması

Simülasyon ortamının hazırlanması sırasında aşağıdaki kriterler belirlenerek, bu kriterler ile işlemlerin gerçekleşmesi sağlanmıştır. Kriterlerin belirlenmesi esnasında gerçek bir uygulama ortamında meydana gelecek olan olaylara benzer durumlar oluşturmak üzere seçim yapılmıştır.

1. **Simülasyon sayısı:** Gerçekleştirilecek olan simülasyonun tekrarlanma sayısı. Simülasyon belirtilen adet kadar tekrarlanarak, her algoritmanın gerçekleşen simülasyon sırasında istenilen blok yapısının tampon bölgesinde bulunup bulunmadığı (hit/miss) sayılarak oransal başarım değeri elde edilecektir.
2. **Tampon Büyüklüğü**: Gerçekleştirilecek olan simülasyon sırasında kullanılacak olan tampon büyüklüğü
3. **Diskteki Toplam Blok Sayısı:** Gerçekleştirilecek olan simülasyon sırasında kullanılacak olan diskte bulunacak olan toplam blok sayısı

4. **Odaklanılan Veri Grubunun Yüzdesi (ovgy)**: Okuma ve yazma işlemleri sırasında kullanılacak olan veri grubundan belli bir yüzdesine odaklanma için veri grubu yüzdesi belirtilerek belli bir alan kapsamında işlemlerin gerçekleştirilmesi amaçlanmıştır.
5. **Odaklanılan Veri Grubundan Okuma İhtimali (ovgoi)**: Odaklanılan yüzdelik veri grubu içerisinden işlemler sırasında okuma ihtimali yüzdesi. Bu parameter üzerinde yapılacak olan değişimler ile tampon bölgesinde tutulacak olan verininin tamponda bulunma ihtimalinin artırılması amaçlanmıştır.
6. **Okunan Verinin Değiştirilme İhtimali (ovdi)**: Okunan verinin değiştirilme ihtimali paramatresi tampon bölgesinde bulunan ve üzerinde okuma işlemi gerçekleşmiş olan bloklar üzerinde değişim oranı belirlenerek uygulanacak olan simülasyon için farklı bir durum oluşturulmuştur.

Test aşamasında belirlenen simülasyon parametreleri üzerinde farklı değerler ile sistem test edilerek sonuçlar elde edilmiştir. Her bir parametre değişimi sonucu farklı algortimaların kullanım değerleri tespit edilmiştir. Tampon bölgesi yönetimi sayfa değişim algoritması olarak, Basic, FIFO, LRU, Clock, LRM olmak üzere 5 adet algoritma Simpledb ilişkisel veritabanı üzerinde gerçeklenmiştir.

Çizelge 1. Test konfigürasyonu 1

| Konfigürasyonlar & Parametreler | Tampon büyüklüğü | Simülasyon adeti | Blok sayısı |
|---|---|---|---|
| **Konf 1,2,3,4** | 20 | 100 | 1000 |
| **Konf 5,6,7,8** | 50 | 100 | 1000 |
| **Konf 9,10,11,12** | 20 | 500 | 1000 |
| **Konf 13,14,15,16** | 50 | 500 | 1000 |
| **Konf 17,18,19,20** | 50 | 500 | 3000 |

Çizelge 1'de gerçekleştirilmiş olan testlerde kullanılan konfigürasyon yapılarına ait tampon büyüklüğü, simülasyon adeti ve blok sayıları görülmektedir.

Çizelge 2. Test konfigürasyou 2

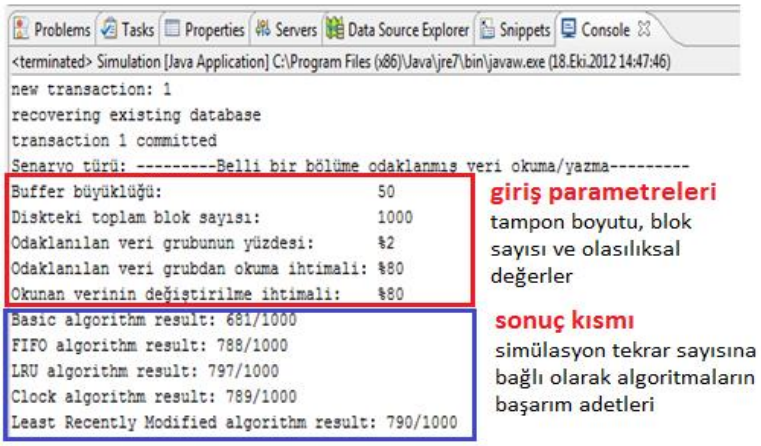
| Konfigürasyonlar & Parametreler | Ovgy | Ovgoi | Ovdi |
|---|---|---|---|
| **Konf 1,5,9,13,17** | 0,02 | 0,8 | 0,8 |
| **Konf 2,6,10,14,18** | 0,04 | 0,8 | 0,8 |
| **Konf 3,7,11,15,19** | 0,02 | 0,6 | 0,8 |
| **Konf 4,8,12,16,20** | 0,02 | 0,8 | 0,6 |

Çizelge 2'de gerçekleştirilmiş olan testlerde kullanılan konfigürasyon yapılarına ait odaklanılan veri grubu yüzdesi , odaklanılan veri grubundan okuma ihtimali ve okunan verinin değiştirilme ihtimali değerleri görülmektedir.

## 4.TEST SONUÇLARI, KARŞILAŞTIRMA VE DEĞERLENDİRME

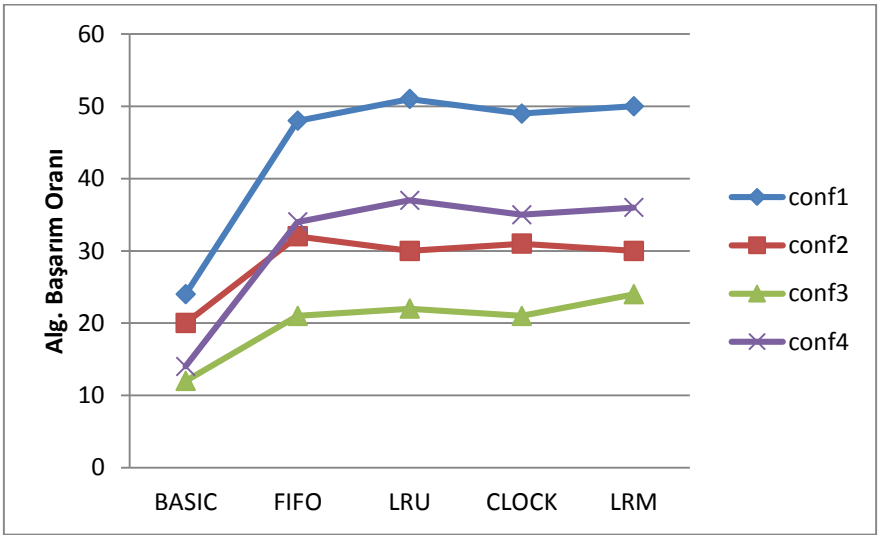
Şekil 4'de simpledb ilişkisel veri tabanı ve eclips java platformu kullanılarak geliştirilen simülasyon ortamında gerçekleştirilen bir test durumuna ait ekran çıktısı görülmektedir. Ekran çıktısı incelendiğinde sonuç ekranında simülasyonun çalışması için gerekli olan parametrik değerler görülmektedir. Tampon boyutu,disk blok sayısı, çizelge 2 de girilmiş olan olasılıksal değerler ve simülasyon adeti girişi gerçekleştirilmektedir. Sonuç olarak ise girilen simülasyon adetine bağlı olarak hangi algoritmanın kaç adet simülasyon değerinde istenilen veriyi tampon

bölgesinde bulundurabildiği değeri sonuç olarak üretilmektedir. Test işlemleri sırasında 20 adet farklı konfigürasyon için işlemler gerçekleştirilmiş ve değerler kaydedilmiştir.



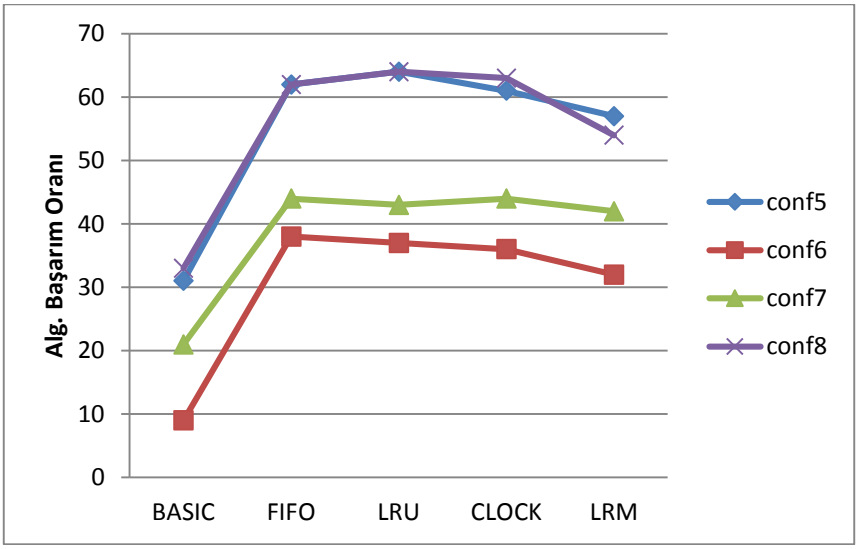

Şekil 4. Test ekran çıktısı örneği

### 4.1. Test Sonuçları



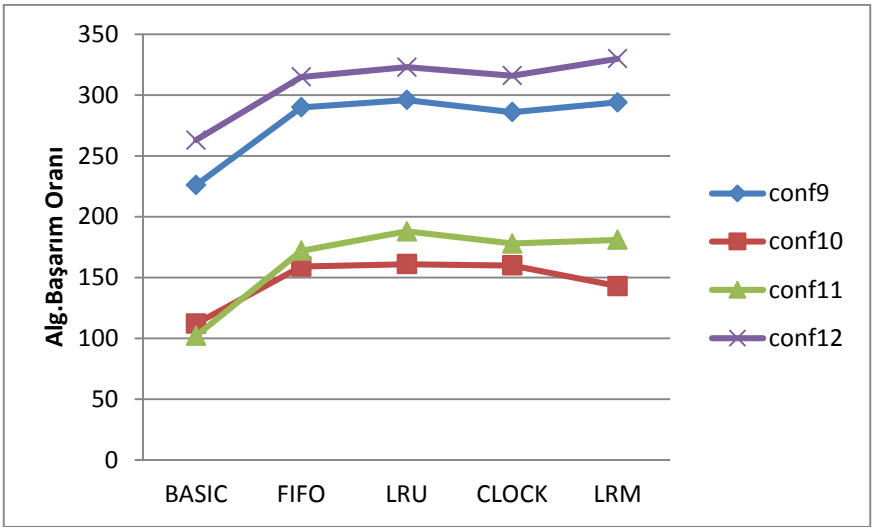

Şekil 5. Buffer boyutu: 20, simülasyon sayısı:100, blok adeti : 1000

Şekil 5'te tampon büyüklüğü 20, blok adeti 1000, çizelge 2 de belirtilen konfigürasyon 1, 2, 3, 4'e ait olasılıksal değer ile simülasyon 100 adet tekrarlandığında algoritmaların başarımını gösteren grafik elde edilmiştir.



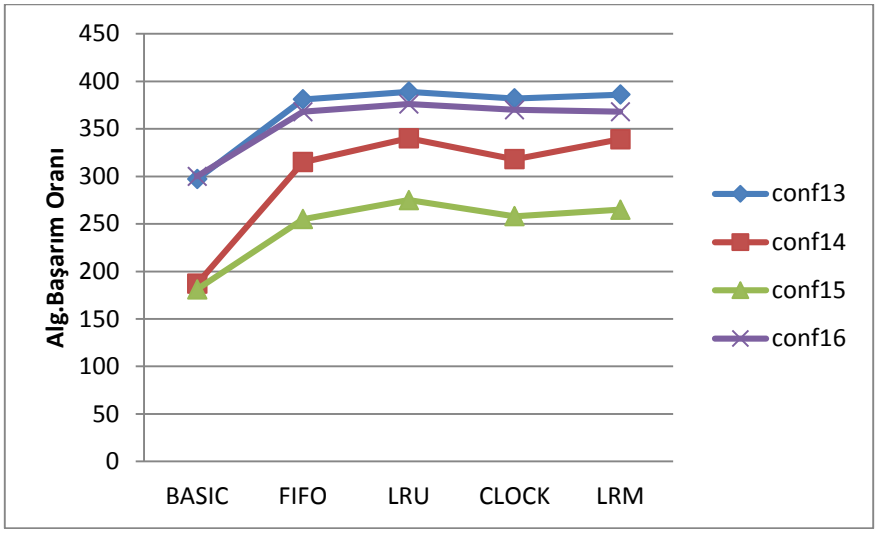

Şekil 6. Buffer boyutu: 50, simülasyon sayısı: 100, blok adet : 1000

Şekil 6'da tampon büyüklüğü 50, blok adeti 1000, çizelge 2de belirtilen konfigürasyon 5, 6, 7, 8'e ait olasılıksal değer ile simülasyon 100 adet tekrarlandığında algoritmaların başarımını gösteren grafik elde edilmiştir.



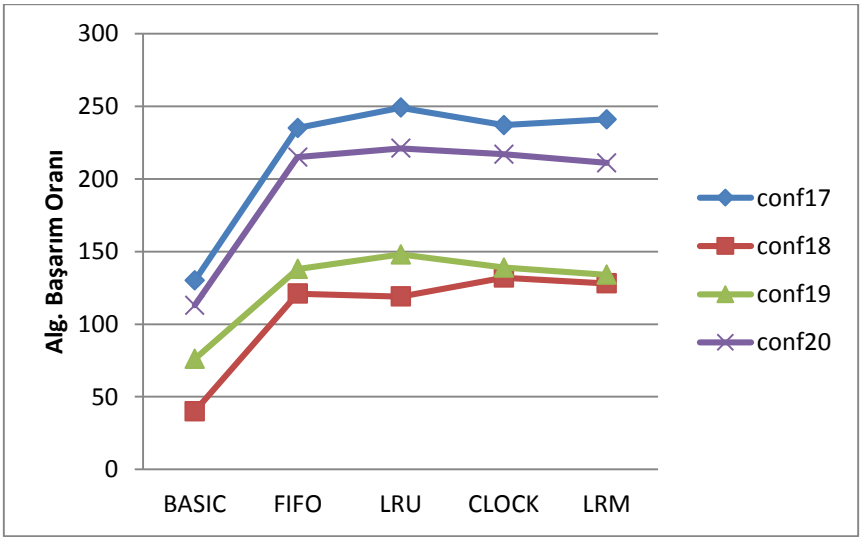

Şekil 7. Buffer boyutu: 20, simülasyon sayısı: 500, blok adeti: 1000

Şekil 7'de tampon büyüklüğü 20, blok adeti 1000, çizelge 2de belirtilen konfigürasyon 9, 10, 11, 12' ye ait olasılıksal değer ile simülasyon 500 adet tekrarlandığında algoritmaların başarımını gösteren grafik elde edilmiştir.




Şekil 8. Buffer boyutu: 50, simülasyon sayısı: 500, blok adeti : 1000

Şekil 8'de tampon büyüklüğü 50, blok adeti 1000, Çizelge 2'de belirtilen konfigürasyon 13, 14, 15, 16' ya ait olasılıksal değer ile simülasyon 500 adet tekrarlandığında algoritmaların başarımını gösteren grafik elde edilmiştir.




Şekil 9. Buffer boyutu: 50, simülasyon sayısı: 500, blok adeti: 3000

Şekil 9'da tampon büyüklüğü 50, blok adeti 3000, çizelge 2 de belirtilen konfigürasyon 17, 18, 19, 20' ye ait olasılıksal değer ile simülasyon 500 adet tekrarlandığında algoritmaların başarımını gösteren grafik elde edilmiştir.

Simülasyon testlerinde, farklı parametrelerin değişimi sağlanarak, bu parametrelerin simülasyon üzerindeki başarımları, faklı algoritmalar üzerindeki etkileri tespit edilmiştir. Kullanılan algoritmalar ve değişik parametrelerin simüle edilmesi sonucu Şekil 5, 6, 7, 8, 9 da görülen sonuçlar elde edilmiştir. Simülasyon sonuçlarında simülasyon tekrar sayısı temel alınarak tekrar edilen değerin kaç adedinde algoritmanın başarım sağladığı tespit edilmiştir. Örneğin şekil 8 de b durumu için LRU algoritmasının başarım değeri tekrarlanan 500 adet simülasyonda 340 olarak tespit edilmiştir. Gerçekleştirilen simülasyonlarda belli parametreler sabit tutularak diğerleri değiştirilmek suretiyle testler gerçekleştirilmiştir. Simülasyon için kullanılan tüm parametreler gerçek bir işlem sırasında oluşabilecek olan durumların

gerçekleştirilmesi için tasarlanmıştır. Parametrelerin değişimi gerçek ortamda kullanılan donanımsal ve yazılımsal kaynaklara göre değişiklik göstermektedir.

Parametreler bakımından sonuçlar değerlendirildiğinde, tampon sayısı artırıldığında sonuçların ciddi şekilde arttığı tespit edilmiştir. Tampon bölgesinin büyültülmesi olumlu etkiler getirmekle beraber belli bir oranda tutulması önemlidir. Simülasyon sayısının artırımında ise oransal olarak değerlendirildiğinde başarım oranının düştüğü, blok sayısı artırılması durumunda ise gerçekleştirilen simülasyonda algoritmaların başarım oranlarının azaldığı tespit edilmiştir. Odaklanılan veri grubu yüzdesi azaltıldığında algoritmaların başarım oranı artmış, odaklanılan bölgenin artırılması sonucu seviyesinin düştüğü tespit edilmiştir. Odaklanılan veri grubundan okuma ihtimali azaltıldığında algoritmaların başarım oranınının azaldığı görülmüştür. Okunan verinin değiştirilme ihtimali açısından değerlendirildiğinde, değiştirilme ihtimali azaltığında başarım oranının %10 civarında düştüğü tespit edilmiştir.

Simülasyon sırasında kullanılan algoritmalar açısından değerlendirildiğinde; Basic algortimasının testlerin tamamında en düşük başarım oranına sahip olduğu tespit edilmiştir. FIFO algoritmasının Basic algoritmasından oldukça başarılı sonuçlar elde ettiği, LRM algoritmasının ise farklı test simülasyon durumlarında dalgalı bir şekilde bazı durumlarda diğer algoritmalardan daha iyi bazılarında ise biraz daha kötü sonuçlar ürettiği tespit edilmiştir. LRU ve Clock algoritmaları ise tüm testlerde en istikrarlı yapıya sahip sonuçları üretmişlerdir. Test sonuçlarına bütün olarak bakıldığında LRU algoritması diğer algoritmalardan daha başarılı değerler üreterek temel algoritmalar içerisinde en başarılı algoritmik yapı olarak tespit edilmiştir.

## 5. SONUÇ VE DEĞERLENDİRME

Bu çalışmada veri tabanı yönetim sistemlerinin performansı üzerinde büyük etkiye sahip olan bellek yönetiminde tampon bölgesi sayfa değişim algoritmaları ele alınmıştır. Veri tabanı üzerinde işlemler gerçekleştirilirken işletim sistemi tarafından veri tabanı yönetim sisteminin kontrolüne verilmiş olan bellek üzerinde bulunan tampon bölgesinde gerçekleşen sayfa değişim algoritmaları kullanılmaktadır. Bu algoritmaların kullanılmasıyla, sistem üzerinde gerçekleşecek olan işlemlere ait ihtiyaç duyulan veriler için tekrar disk erişimi veya sanal belleğe olan ihtiyacı azaltacak veya tamamen ortadan kaldıracaktır. Bu şekilde gerçekleşen uygulamalar sistemin hız ve performansını çok büyük ölçüde artıracaktır.

İşlemci üzerinde işlenecek olan verilerin tampon bölgesinde hazır bir şekilde tutulması için çeşitli algoritmalar kullanılarak bu verilerin seçim işlemi gerçekleştirilir. Kullanılan algoritmanın çalışma mantığına göre tampon bölgesindeki veriler yerinde bırakılıp veya bulundukları bölgeden çıkarılarak diske gönderilir veya diskten bazı veriler getirilerek tampon bölgesine yerleştirilirler. İşte bu noktada kullanılan algoritma ihtiyaç duyulacak olan veriyi ne kadar çok tampon bölgesinde tutabilirse başarımı o oranda artacaktır. Algoritmaların amacı işlem görecek olan veriyi tahmin edip, öngörüde bulunarak tampon bölgesini kullanılacak olan veriler ile efektif bir şekilde yönetmektir.

Makalede tampon bölgesinde sayfa değişimlerini gerçekleştirecek olan temel yapıda algoritmalardan 5 adeti incelenerek, bu algoritmalar simpledb ilişkisel veri tabanı üzerinde hazırlanan senaryoda test edilmiştir. Bu çalışmada temel algoritmalar üzerinde yapılan testler farklı parametreler kullanılarak gerçekleştirilmiş, gerçek test ortamının üretimi için farklı parametreler kullanılmıştır. Test sonuçları değerlendirildiğinde, Basic algoritması en düşük başarım oranına, ardından FIFO algoritmasının Basic algoritmasından daha iyi sonuçlara sahip olduğu tespit edilmiştir. LRM algoritmasının sonuçları parametric değerlere göre değişkenlik göstermiş fakat FIFO algoritmasından daha iyi sonuçlar elde ettiği görülmüştür. 5 temel algoritma içinde LRU ve Clock algoritmaları en iyi test değerlerine sahip

algoritmalardır. LRU ve Clock algoritmaları gerçekleştirilen tüm test durumlarında istikrarlı sonuçlar üretmişlerdir. Tüm test sonuçları değerlendirildiğinde LRU algoritmasının diğer algoritmalardan daha başarılı sonuçlar ürettiği tespit edilmiştir.

## KAYNAKLAR


Chou H. T., Witt D. J. (1986): "An Evaluation of Buffer Management Strategies for Relational Database System", *Algorithmica1*, sf. 311-336 .

Daula S., Murthy K .E S., Amjad K. G. (2012): "A Throughout Analysis on Page Replacement Algorithms in Cache Memory Management", *International Journal of Engineering Research and Applications*, Cilt 2, No. 2, sf. 126-130.

DBMS Bellek Dizaynı (2011),http://en.wikibooks.org/wiki/Design_of__Main_Memory Database_System/Overview of DBMS, Erişim tarihi: 24.05.2013.

Effelsberg W., Haerder T. (1984) :"Principles of Database Buffer Management", *ACM Transactions on Database Systems*, Cilt 9, No. 4, sf. 560-595.

Haas L. M., Chang W., Lohman G. M., McPherson J., Wilms P. F., Lapis G. B., Lindsay G., Pirahesh H., Carey M. J., Shekita E. J. (1990):"Starburst Mid-Flight: As the Dust Clears", *IEEE Transactions on Knowledge and Data Engineering*, Cilt 2, No. 1, sf. 143–160.

JDBC (2008): http://www.oracle.com/technetwork/java/javase/jdbc/index.html, Erişim tarihi:10.05.2013.

Jung H., Han H., Kim S. G., Yeom H. Y. (2009): "A Practical Evaluation of Large-Memory Data Processing on a Reliable Remote Memory System", *In Proceedings of the 2009 ACM symposium on Applied Computing*, sf. 343-344.

Lily Y. X. (2001): "Analytical Modeling for Buffer Hit Rate Prediction", Queen's University, Y. Lisans Tezi , Kanada.

Lee D., Choi J., Kim J. H., Noh S. H., Min S. L., Cho Y., Kim C. S. (1999): "On the Existence of a Spectrum of Policies that Subsumes the Least Recently Used (LRU) and Least Frequently Used (LFU) Policies", *International Conference on Measurements and Modeling of Computer Systems*, sf. 134–143, Atlanta, Amerika.

Nicola V. F., Dan A., Dias D. M. (1992): "Analysis of the Generalized Clock Buffer Replacement Scheme for Database Transaction Processing", *International Conference on Measurement and Modeling of Computer Systems*, sf. 35–46.

O'Neil E. J., O'Neil P. E., Weikum G. (1993): "The LRU-K Page Replacement Algorithm for Database Disk Buffering", *International Conference on Management of Data*, sf. 297–306.

Peter J. D. (1970): " Virtual Memory", *Computing Surveys*, Cilt 2, No. 3, sf. 120-135.

Sacco G. M., Schkolnick M. (1982): "A Mechanism for Managing the Buffer Pool in a Relational Database System Using the Hot Set Model", *In Proceedings of the 8th International Conference on Very Large DataBases*, sf. 257–262, Mexico City, Meksika.

Sacco G. M., Mario S. (1986): "Buffer management in relational database systems", *ACM Transactions on Database Systems*, sf. 473-498.

Sacco G. M. (1987): "Index Access with a Finite Buffer", *In Proceedings of the 13th International Conference on Very Large Data Bases*, sf. 301–309, Brighton, İngiltere.

Sciore E. (2007): "*Database Management: A Systems Approach Using Java*", John Wiley and Sons, Boston College, sf. 357 ve sf. 372.

Sciore E., Simple D. B. (2007): "A Simple Java-Based Multiuser System for Teaching Database Internals", *ACM SIGCSE Bulletin*, Cilt 39, No 1, sf. 561–565.

Schoening H. (1998): "The ADABAS Buffer Pool Manager", *In Proceedings of the 24th International Conferenceon Very Large Databases*, sf. 675–679, New York, Amerika.

Simpledb (2007), http://www.cs.bc.edu/~sciore/simpledb/intro.html , Erişim tarihi: 15.05.2013.

Smaragdakis Y., Kaplan S., Wilson P. (1999): "Simple and Effective Adaptive Page Replace", *International Conference on Measurements and Modeling of Computer Systems*, sf. 122–133, Atlanta, Amerika.

Stefan M., Peter B. (2002):"Optimizing Main-Memory Join on Modern Hardware", *IEEE transactions on knowledge and data engineering*, Cilt 14, No. 4, sf. 210-220.

Teng J. Z., Gumaer R. A. (1984): "Managing IBM database 2 Buffers to Maximize Performance", *IBM Systems Journal*, Cilt 23, No. 2, sf. 211–218.

Wang W., Bunt R. (2000): "Simulating DB2 Buffer Pool Management", *In Proceedings of CASCON 2000*, Departmant of computer Sience University of Saskatchewan.

Wang W. (2001): "Storage Management in RDBMS", Department of Computer Science University of Saskatchewan.

Willick D. L., Eager D. L., Bunt R. B. (1990): "Disk Cache Replacement Policies for Network Fileservers, *In Proceedings of the 10th International Conference on Distributed Computing Systems*, sf. 212–219, Paris, Fransa.