7. HASHING

7.1. Klasik Hash Yöntemi

Bir dosyadaki kayıtların birincil anahtarların değerlerinin 1 ile 1000 arasında olduğunun bilindiği kabul edilsin. Bu bilgiler kullanılarak kayıtlar disk üzerine düzenli bir şekilde kayıt edilebilirler. Birincil anahtar değeri 1 olan kaydın adresi 1 olan bloğa kayıt edilmesi, birincil anahtar değeri 2 olan kaydın adresi 2 olan bloğa kayıt edilmesi ve bu şekilde devam edilerek birincil anahtar değeri n olan kaydın adresi n olan bloğa kayıt edilmesi şeklinde bir organizasyon yapılabilir. Bu şekilde yapılan bir organizasyondan sonra, bir kaydın aranması çok kolay olur. Bu şekilde yapılan bir organizasyonda tek kayıt erişimi B+-ağaçlarından daha hızlı olur, çünkü tek disk erişimi ile işlem tamamlanıyor, bunun aksine B+-ağaçlarında bir kaydın aranması için en az iki tane disk erişimi yapılması gerekmektedir.

Hashing yöntemi, yukarıda anlatılan yöntemin genel ismidir. Verilen bir kaydın anahtarından, bu kaydın kayıt edilecek olan disk bloğunun adresini bulma işlemine **Hash algoritması** veya **Hash Fonksiyonu** denir. Bu hash yönteminin değişik terimlerle ifade ediliş şekilleri vardır. Doğrudan erişim (direct access), gelişigüzel erişim (random access), anahtardan adres transformasyonu (key-to-address transformation).

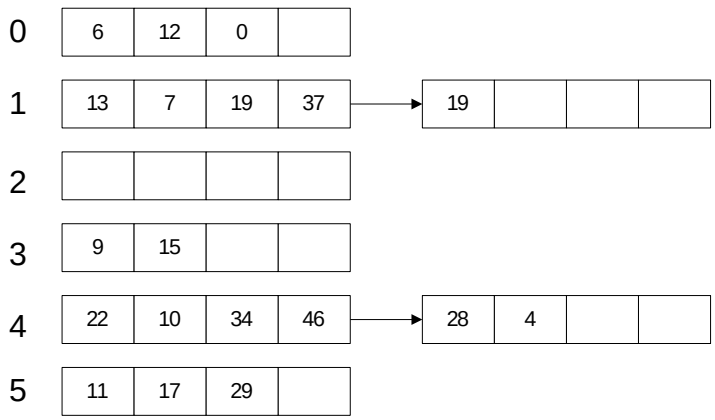
Kayıtların saklanması diskten yer ayrılması gerekir ve bu yer ayırma işlemi yapılırken, mümkün olduğunca boş yer bırakılmamaya çalışılır. Kayıtlar için ayrılan bu alana **birincil alan** denir. Birincil alan, sepetlere bölünür ve her sepet bir veya daha fazla kayıt içerebilir.

Hash fonksiyonu kullanılarak bilgilerin diske kayıt edilmesi için blok adresi hesaplanır ve genelde hash fonksiyonu kayıtların sırasını korumaz. Bazı durumlarda farklı kayıtlar için hesaplama sonucunda aynı blok adresi bulunabilir. Bu duruma çakışma (collision) denir. Çakışma meydana geldiği zaman, bu problemi çözmek için birincil alandan hariç taşma alanı kullanılır. Buna zincirleme de (chaining) denir. Aynı hash değerli kayıtların hepsi birincil alandaki sepet doluncaya kadar bu

sepete yazılırlar. Birincil alandaki sepet dolduktan sonra taşma alanındaki sepete yazılmaya başlanır. Taşma alanındaki birinci sepet dolduktan sonra bir sepet daha taşma alanına eklenir ve bu sepete yazılmaya başlanır. Bu şekilde devam edilerek çakışma problemi çözülür. Şekil 7.1' de f(x)=x mod 6 fonksiyonuna göre gelen kayıtların disk adreslerinin nasıl hesaplandığı ve diske nasıl yazıldıkları görülmektedir.

7.1.1. Yükleme Faktörü

Bir dosya için gerekli olan bütün uzay birincil alan için ayrılırsa, daha sonraki güncellemelerden dolayı zincirleme durumu ortaya çıkabilir ve bu zincirlerdeki sepet sayısı fazla olabilir. Hash fonksiyonu kayıtları düzenli dağıtmayacağından dolayı bazı hash değerine karşılık çok sayıda kayıt denk gelirken, bazı hash değeri boş kalabilir veya çok az sayıda kayıt denk gelebilir. Taşmayı sınırlamak için birincil alana gerekli olan alandan daha fazla alan ayrılır. Bundan dolayı birincil alanın yüzde kaçının en fazla dolu olacağının belirtilmesi gerekir. Birincil alan olarak ayrılmış bölgeye yazılması istenen kayıt sayısının bu alana yazılabilecek maksimum kayıt sayısına oranı **yükleme faktörü** (Lf) denilen bir değer verir.



Şekil 7.1. Birincil alan ve taşma alanındaki sepetler. Burada hash fonksiyonu f(x)=x mod 6 şeklinde tanımlanmıştır.

Bkfr daha önce belirtildiği gibi bir sepete sığan kayıt sayısıdır ve M ise birincil alandaki sepet sayısı olmak üzere

$$Lf = \frac{n}{MxBkfr} \tag{7.1}$$

olur. n dosya içindeki kayıt sayısı ve MxBkfr ise birincil alana yazılabilecek kayıt sayısıdır. Sepet boyutu ve yükleme faktörünün okuma zamanına olan etkisi incelenebilir.

7.1.2. Sepet ve Zincir Kullanılarak Kayıt Okuma

Eğer veri dağılımı dengeli ve ayrılan alan yeterince büyükse, hashing iyi sonuç verir ve bir kayıt okuma zamanı yaklaşık olarak bir disk erişimi gerektirir. Bir kayıt okuma zamanı

$$T_F = s+r+dtt \tag{7.2}$$

olur. Bir sepet içinde birden fazla blok olabilir ve dtt zamanı bir sepet içindeki blokların toplam transfer zamanıdır.

Yükleme faktörü küçük seçildiğinde taşma olma olasılığı düşer ve okuma işlemlerin zamanı azalır ve bunun yanında küçük yükleme faktörünün anlamı uzayın önemli bir kısmının boş tutulmasıdır.

Büyük sepet faktörü seçilmesi durumunda da taşma olma ihtimali düşer ve bunun yanında sepet faktörünün büyümesi dtt değerinin artması anlamına gelir ve bu da beraberinde okuma/yazma işlemlerinin zamanlarının artması anlamına gelir. Verilen bir yükleme faktörü ile birlikte büyük sepet faktörü için gerekli olan disk

erişimi her zaman küçük olan sepet faktörü için gerekli olan disk erişiminden küçüktür.

Bu durumdan yola çıkılarak birincil alan için sepet faktörü büyük ve taşma bölgesi için sepet faktörü küçük olan bir strateji uygulanabilir. Her iki alan için sepet boyutları eşit olacak şekilde bir ayarlama da yapılabilir. Bunun yanında birden fazla birincil alandaki sepet taşma alanındaki bir tane sepeti ortaklaşa olarak kullanabilirler. Buradan da görülebileceği gibi bölgelere göre sepet faktörü ayarlaması yapılabileceği gibi çok değişik hash fonksiyonu da seçilebilir. Bütün bu tercihler tasarımı yapan kişiye aittir.

Uygun sepet ve yükleme faktörlerinin seçimi sonucunda performans iyileştirilebilir. Uzay kullanımı ise yükleme faktörünün seçimine ve taşma bölgesinin kullanımı ile ilgili olan stratejiye bağlıdır. bu basit bir dosya tasarım problemidir.

7.1.3. Veri Tabanlarının Büyümesi

Veri tabanlarının hızlı büyümesi durumunda, yükleme faktörünün seçiminin iyi yapılması ve taşma alanı ile ilgilenme için takip edilen stratejinin iyi olması, sistemin performansının iyi olması anlamına gelmez, çünkü veri tabanının büyümesi takip edilen bütün stratejileri etkilemektedir. Veri eklemesinin çok yapılması, taşma zincirinin uzaması anlamına gelir. Bundan dolayı hash yöntemini kullanan sistemlerin çoğu tekrar organize etme birimine de sahiptirler.

Taşma var ve ortalama taşma zincirinin uzunluğu L ise, bir kaydı okuma zamanı

$$T_F(\text{başarılı}) = \underbrace{s + r + dtt}_{\text{birinci alan}} + \underbrace{\frac{L}{2}(s + r + dtt)}_{\text{Taşaş alanı}} \qquad (7.3)$$

olur. Bu zaman bağıntısının anlamı, birincil alana bir erişim yapılır ve kaydın orada olup olmadığı kontrol edilir. Kayıt birincil alanda değilse, ortalama taşma bölgesindeki sepet sayısının yarısına erişim yapılınca kayıt okunmuş olur.

Dosyada olmayan bir kaydın aranması için harcanan zaman dosyada olan bir kaydın aranması için harcanan zamandan daha fazla olur. Dosyada olmayan bir kaydın aranması durumunda birincil alandaki sepet okunur ve taşma bölgesindeki bütün sepetler okunur. Bu durumda başarısız okumanın zaman bağıntısı (7.4)' te görülmektedir.

$$T_F(\text{başarısız}) = \underbrace{s + r + dtt}_{\text{Birincil alan}} + \underbrace{L(s + r + dtt)}_{\text{Taşaş alanı}} \qquad (7.4)$$

B+-ağaçlarında başarılı ve başarısız aramalarda disk erişim sayıları aynıydı, zincir içeren hash yönteminde aynı değildir. Eğer taşma zinciri çok uzun olan bir hash tablosunda başarısız arama başarılı aramaya göre çok daha fazla zaman alır. Eğer taşma zinciri yok veya ortalama olarak uzunluğu çok küçükse, bu durumda yaklaşık olarak başarılı ve başarısız aramaların zamanları aynı olur.

7.1.4. Kayıt Silme

Kayıt silmenin iki tane iyi yolu vardır. Bunlardan biri çoğu sistem tarafından kullanılan silinen kaydın işaretlenmesi ve işlemler sırasında bu kaydın sürekli ihmal edilmesidir. Hash tablosu tekrar organize edildiği zaman, işaretli olan kayıtlar yeni tabloya aktarılmazlar.

Diğer yöntem ise, silinen kaydın yerine taşma zincirinin en sonunda bulunan kaydın silinecek kaydın üzerine kopyalanması işlemidir. Bu silme işleminde dikkatli olunması gerekir, çünkü bir kere silinen kayıt tekrar getirilemez.

Bu durumda taşma zincirinin sonunda bulunan kayıt kopyalandıktan sonra, eski yeri boş hafızaya iade edilir. Bu yöntemin uygulanması birinci yönteme göre daha karmaşıktır, çünkü ilk olarak taşma zincirinin sonundaki kayıt kopyalandıktan sonra taşma zincirinin sonundaki kayıt silinecektir. Bunun anlamı bir kayıt silme işlemi, bir başarılı arama, bir başarısız arama ve bir kayıt ekleme işlemlerinden oluşur. Bu yöntemin avantajı, arama işlemlerini hızlandırır. Bu durumda zaman bağıntısı

$$T_D=T_F(\text{başarısız})+2r \quad (7.5)$$

olur. (7.5) bağıntısında taşma zincirinin sonundaki kaydı boş hafızaya iade etme işlemi için harcanacak zaman ihmal edilmiştir.

7.1.5. Kayıt Ekleme

Kayıt eklemede, eğer birincil alan dolu ise, eklenecek kayıt taşma zincirinin sonuna eklenir. Eğer kayıt eklemede taşma zincirinin sonuna sepet eklemesi yapılmayacaksa, kayıt ekleme sadece taşma zincirinin sonundaki sepet içinde güncelleme yapma işleminden oluşur. Kayıt ekleme için zaman bağıntısı

$$T_I=T_F(\text{başarısız})+2r \quad (7.6)$$

olur. Yeni bir sepet eklemeye ihtiyaç olduğu durumlar olabilir; bu durumlarda sepet ekleme için harcanacak zaman ihmal edilmiştir. Genelde kayıt ekleme bir kayıt arama ve bir kayıt güncelleme işleminden oluşur.

7.1.6. Ardışıl İşlemler

Yapılacak işlemler sınır sorgulamaları veya belli bir kritere göre sıralama işlemlerini içeriyorsa, hash yöntemi kullanışsız olur. Çünkü, hash yönteminde hiçbir zaman sıra korunması olmaz. Bundan dolayı sadece tek kayıt işlemleri için etkili bir yöntem olur. Hash yöntemi sırayı korumadığından dolayı, bir dosyanın kayıtlarını sırayla okumak için zaman bağıntısı

$$T_X=nT_F \quad (7.7)$$

olur. Tek kayıt erişimlerinde hash yöntemi genelde B+-ağacı yönteminden daha iyidir.

7.2. Lineer Hash Yöntemi

Oluşturulan bir hash tablosunun tekrar organize edilmemesi için değişik yöntemler geliştirilmiştir ve bu yöntemlerden biri de lineer hash yöntemidir. Lineer hash yöntemi, ne kadar kayıt eklemesi yapılırsa yapılsın, yükleme faktörü sürekli sabit tutulur. Yükleme faktörünü sabit tutabilmek için, kayıt eklemeleri yapılırken belli bir kayıt eklemesi sonunda birincil alana sepet eklemesi yapılır.

7.2.1. Bir Kayıt Okuma

Lineer hash yönteminde her kaydın ikili tabanda yazılmış hash numarasının son bitleri kullanılarak tabloya kayıt eklemesi yapılır. Bundan dolayı ilk önce verilen kaydın hash numarası hesaplanır ve ondan sonra hash numarasının son k bitine göre tabloya yerleştirme işlemi gerçekleştirilir.

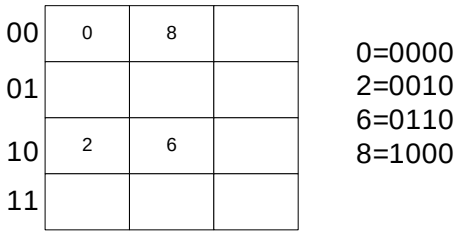
Tablo genişletildiği zaman, son k bite göre bütün kayıtları içeren sepetler son k+1 bitine göre iki parçaya bölünür. Örneğin son üç biti 011 olan bir sepet 1011 ve 0011 şeklinde iki tane sepete bölünür.

İki parçaya bölme işlemi belli bir algoritmaya göre yapılmaktadır ve bu algoritma kayıt ekleme kısmında anlatılacaktır. Bir kaydı okumak için bazı sepetlerin son k bitine göre ve bazı sepetlerde son k+1 bitine göre yerleştirme yapıldığının bilinmesi yeterlidir. Sepetlerin hangilerinin son k biti ve hangileri için son k+1 bit kullanılacağını gösteren değere **sınır değeri** denir. Sınır değeri ve k' nin değeri hafızada tutulur. İlk olarak tabloda kayıt arama için arama algoritması aşağıda verildiği gibidir.

Algoritma. Lineer Hash Tablosunda Kayıt Arama

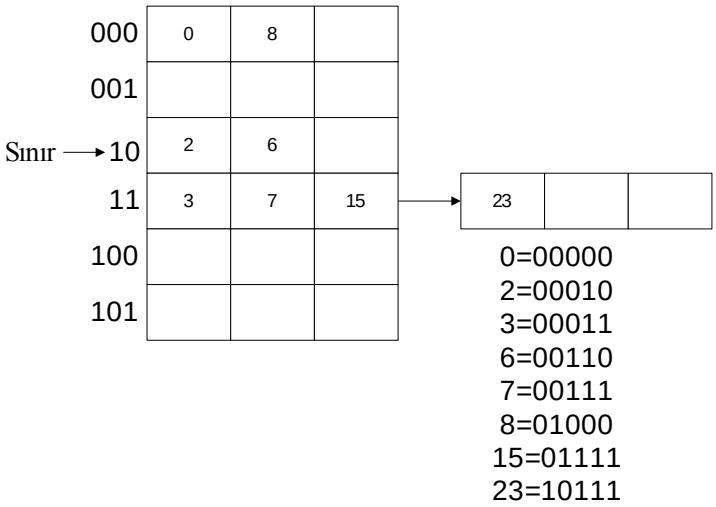
1. Hash fonksiyonun değerini hesapla
2. Hash değerinin son k bitine bak. Eğer son k bitin değeri sınır değerinden küçükse, aranan kaydı bulmak için son k+1 bitin kullanılması gerekir. Eğer son k bit değeri sınır değerine eşit veya büyükse, kaydın yeri son k bit kullanılarak bulunabilir.

3. Birincil alanda kayıt yoksa, taşma zincirini ardışıl olarak oku.



Şekil 7.2. Lineer hash yöntemi ile oluşturulan tablo.

Şekil 7.2' de görülen tabloda hash değerlerinin son 2 bitine göre yerleştirme yapılmıştır ve sınır değeri 00 değeridir.



Şekil 7.3. Son 2 veya 3 biti kullanan lineer hash tablosu.

Şekil 7.3' te verilen hash değerlerinin son 2 veya 3 bitleri kullanılarak yerleştirme işlemi yapılmıştır ve sınır değeri de 10' dir. Hash değeri 10100 olan bir kaydın aranması için 00<10 olduğundan son 3 bit kullanılır. Eğer hash değeri 10110 olan bir kaydın aranması için 10=10 olduğundan son 2 bit kullanılır.

Aynı yükleme ve sepet faktörüne sahip olan lineer hash ve zincirli hash aramalarında, zincirli hash tablosunda arama yapma zamanı lineer hash tablosunda arama yapma zamanından daha küçüktür. Çünkü Lineer hash tablosunda son bitlere göre yerleştirme işlemi yapıldığından dolayı, bazı sepetler diğer sepetlerin yaklaşık iki katı doluluğunda olmaktadırlar.

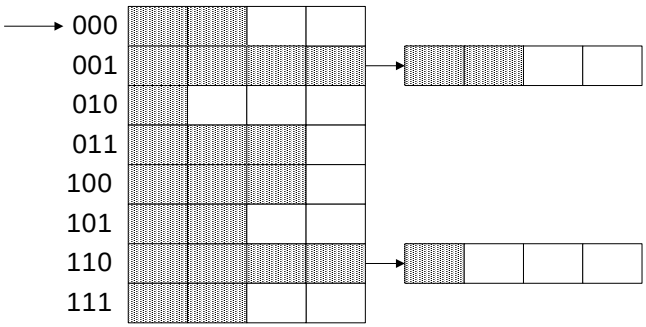
Yükleme faktörü %75 ve sepet faktörü Bkfr=50 olan bir lineer hash tablosunda başarılı ve başarısız aramaların zamanı yaklaşık olarak

$$T_F(\text{başarılı})=1.05(s+r+dtt) \qquad (7.8)$$
$$T_F(\text{başarısız})=1.27(s+r+dtt)$$

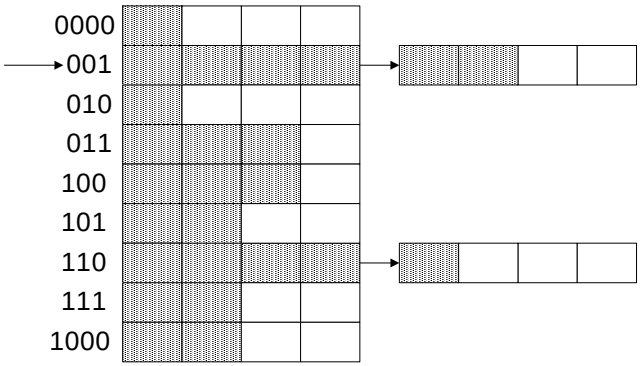
olur. Bu değerlerin toplam kayıt sayısına bağlı olmadığının unutulmaması gerekir. Lineer hash yönteminde okuma zamanının kayıt sayısına bağlı olmaması, bu yöntemin klasik hash yöntemine (zincirli hash) göre daha iyi bir yöntem olduğu aşikardır. Bununla birlikte aynı yükleme faktörlü lineer hash tablosunda bir kaydı okuma zamanı klasik hash tablosundan bir kayıt okuma zamanından biraz fazladır.

7.2.2. Lineer Hash Tablosuna Kayıt Ekleme

Lineer hash tablosuna bir kayıt eklemek için yükleme faktörünün %75 ve Bkfr=4 olsun. Şekil 7.4 (a)' da eleman eklemeden önceki tablonun şekli görülmektedir. Dikkat edilirse, tablonun verilen yükleme faktörüne göre dolu sınırda olduğu görülebilir ve en bir tane yeni eleman ekleme ile tablonun genişleyeceği aşikardır. Şekilde de görüldüğü gibi sınır değeri 000 olup, eklenecek elemanın son üç biti 000 olsun. Bu kaydın eklenmesi sonucu tablonun yükleme faktörü değeri aşılacağından dolayı, tabloya bir sepet eklemesi yapılır ve sınır değeri 000 olduğuna göre genişleme 000 sepeti üzerinde yapılır. Eski sepetin adresi 0000 olur ve eklenen sepetin adresi 1000 olur. Yeni eklenmek istenen kayıt ile eski sepet içindeki kayıtlar bu iki sepet arasında dağıtımı yapılırken son 4 bitine göre dağılım yapılır.



(a)



(b)

Şekil 7.4. Lineer hash tablosuna eleman ekleme. (a) Sınır değeri 000 ve eklenecek olan kaydın hash değerinin son 3 biti 000 olsun. (b) Kayıt eklendikten sonra elde edilen lineer hash tablosu.

Bir lineer hash tablosu oluşturulduktan sonra eklenen kayıt sayısı LfxBkfr değerine ulaştığında veya bu değeri geçtiğinde tabloda genişleme yapılır. Şekil 7.4' te verilen tablo için 3 kayıt eklendikten sonra tabloda genişleme yapılır. Kayıt ekleme algoritması aşağıdaki gibi özetlenebilir.

Algoritma. Lineer Hash Tablosuna Kayıt Ekleme

1. Eklenecek kayıt için doğru sepeti ara.

2. Eğer sepet dolu ise, yeni bir sepet ekle ve kaydı bu sepete ekle ve bu yeni sepetin adresini eski sepetin son bloğuna ekle. Bu şekilde zincir elde edilmiş olur.
3. Eğer LfxBkfr değerinden fazla kayıt eklemesi yapılacaksa, birincil alana bir sepet ekle. O anda sınır değerinin göstermiş olduğu sepet üzerinde genişleme yapılır ve yeni sepetin adresi, sınır değerinin başına 1 değerinin eklenmesi ile elde edilir ve eski sepetin adresinin başına da 0 değeri eklenir. Eski sepet içindeki değerler , varsa eski sepete bağlı zincir içindeki kayıtlar ve eklenecek olan kayıt bu iki sepet ve zincir arasında dağıtılır.
4. Sınır değerine 1 ekle.

Bu yöntemde, yüksek yükleme faktörü kullanıldığı durumlarda taşma bölgesinin oluşma ihtimali artar ve düşük yükleme faktörü de tabloda çok sık genişlemeye sebep olur. Bu tabloda 1/Bkfr olasılıkla yeni bir bağlantı (taşma bölgesine sepet ekleme) eklenir.

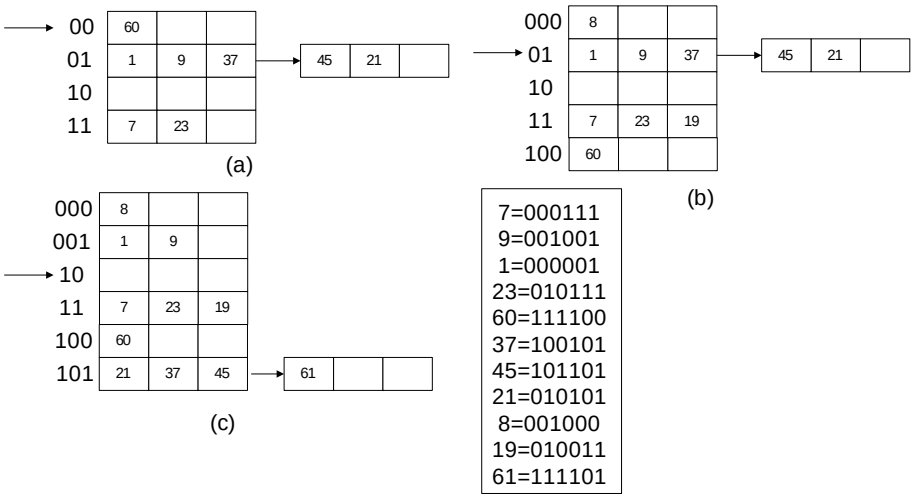
Bu tabloya bir kaydın eklenmesi sadece birincil alana eleman yazmadan ibaret olabilir veya taşma bölgesindeki bir sepete eleman ekleme olabilir veya taşma bölgesine sepet ekleyerek bu kayıt bu sepete eklenebileceği gibi birincil alana sepet ekleyerek yeni kayıt tabloya eklenir. Birincil alana sepet eklenmesi, durumunda eski sepet ve bu eski sepete bağlı olan taşma bölgesinde güncelleme gerçekleştirilir. Kayıt ekleme zaman bağıntısı aşağıdaki gibi olur.

$$T_I = \underbrace{T_F(\text{başaşarıa}) + 2r}_{\substack{\text{Birincil}\\ \text{alana}\\ \text{ekleme}}} + \underbrace{\frac{1}{\text{Bkfr}}(s + r + dtt)}_{\substack{\text{Zincir}\\ \text{alanı}}} + \underbrace{\frac{1}{\text{LfxBkfr}}(s + r + dtt)}_{\substack{\text{Birincil}\\ \text{alana}\\ \text{sepet}\\ \text{ekleme}}} \qquad (7.9)$$

Yükleme faktörü %75 ve sepet faktörü Bkfr=50 olan bir tablo için yaklaşık olarak kayıt ekleme zamanı

T_I=2.62(s+r+dtt) (7.10)

olur.



Şekil 7.5. Yükleme faktörü 2/3 ve Bkfr=3 olan lineer hash tabloları. (a) 7, 9, 1, 23, 60, 37, 45 ve 21 kayıtlarının tabloya eklenmesi. (b) 8 ve 19 kayıtlarının eklenmesi. (c) 61 kaydının eklenmesi.

Şekil 7.5' te yükleme faktörü 2/3 ve Bkfr=3 olan lineer hash yöntemi ile oluşturulan tablolar görülmektedir. Şekil 7.5 (a)' da 7, 9, 1, 23, 60, 37, 45 ve 21 kayıtlarının eklenmesi sonucunda oluşan tablo görülmektedir. Bu tabloya 8 ve 19 kayıtlarının eklenmesi için tabloda genişlemenin yapılması lazımdır. Sınır değeri 00 olduğundan genişleme bu sepet üzerinde yapılır ve bu sepetin adresi 000 ve yeni eklenen sepetin adresi de 100 olur. Daha önce 60 kaydı 00 sepetinde iken yeni sepet eklenmesi sonucunda bu kayıt 100 sepetine eklenir ve 8 kaydı da 000 sepetine eklenir (Şekil 7.5 (b)). Şekil 7.5 (c)' de görüldüğü gibi 61 kaydının eklenmesi için tablonun genişlemesi gerekir ve genişleme 01 sepeti üzerinde yapılır. Eklenen sepetin adresi 101 olurken eski sepetin adresi 001 olur. Eski sepete bağlı bir zincir

var ve bu zincir içinde 21 ve 45 kayıtları bulunurken eski sepet içinde 1, 9 ve 37 kayıtları bulunmaktadır. 61 kaydının eklenmesi sonucunda 21, 37, 45 kayıtları yeni eklenen sepete ve 61 kaydı da eklenen sepete bir zincir eklenerek bu kısma yazılır. Eski sepet içinde ise sadece 1 ve 9 kayıtları kalırken zincir kısmı ortadan kalkmıştır.

7.2.3. Lineer Hash Tablosunda Kayıt Silme

Lineer hash yönteminde bir kaydı silmek için, o kaydın bulunduğu sepete bağlı zincir varsa, zincirin içindeki son kayıt alınır ve silinmesi istenen kaydın üzerine yazılır ve bu şekilde kayıt silinmiş olur. Bu kaydın taşınması sonucunda zincir sepeti boşalırsa, zincir tamamen iptal edilir. Tabloda verilen yükleme faktörünün korunması gerekir. Eğer bu şart için gerekli olan kayıt sayısından BkfrxLf kadar az kayıt varsa, tabloda daraltma yapılır. Daraltma işlemi, genişletme işleminin tersidir. Bu işlemler sonucunda kayıt silme zamanı

$$T_D=T_F(\text{başarısız})+2r \qquad (7.11)$$

olur. Kayıt silinmesi sonucunda tablonun daraltılması veya boşalan zincirin iptal edilmesi durumları nadiren oluştuklarından dolayı zaman hesabı durumunda ihmal edilmişlerdir.

Bu yöntemde kayıt eklemeleri durumda tabloda genişleme meydana gelebileceği gibi kayıt silme durumlarında da tabloda daraltma meydana gelebilir. Bundan dolayı lineer hash yöntemini kullanan organizasyonlarda genellikle tekrar organizasyona başvurmaya pek ihtiyaç duyulmaz.

Lineer hash yönteminin uygulaması biraz karmaşıktır çünkü disk üzerinde birincil alanın genişlemesi için uzay ayırımı yapmak için bitişik bloklardan yer ayrılması her zaman ve hatta genellikle mümkün değildir. Bu problem sırasız yığın dosyaları, sıralı ardışıl dosyalar içinde geçerlidir.

7.2.4. Ardışıl Okuma

Hash fonksiyonu verilerin sırasını korumaz ve bundan dolayı bir dosyanın kayıtlarının sıralı okunması için etkili bir yöntem değildir. Dosyanın kayıtlarını sıralı okumak için zaman bağıntısı

$$T_X = nx1.05(s+r+dtt) \qquad (7.12)$$

olur. Bundan dolayı dosyanın sıralı ardışıl okunması ve sınır sorgulamaları için lineer hash yöntemi kullanılması etkili bir yöntem değildir.