Indeksli Sıralı Erişimli Dosya Yapıları (Indexed Sequential File Organization) ve Bit Seviyesinde İşlemler (Bit Level and Related Structures)

Indeksli Sıralı Erişimli Dosya Yapıları (Indexed Sequential File Organization)

- Sıralı dosyalara sıralı bir şekilde erişim gerçekleştirmek ve doğrudan erişimli dosyalara ise doğrudan erişim gerçekleştirmek performans açısından avantajlıdır. Fakat bu durumun tersi doğru değildir.

- Bunun için indexli sıralı erişimli (indexed sequential file organization) dosya türü geliştirilmiştir.

- Bu dosya yapısı ile kayıtlara ister sıralı ister doğrudan şekilde erişilebilir. Hibrid bir dosya türüdür.

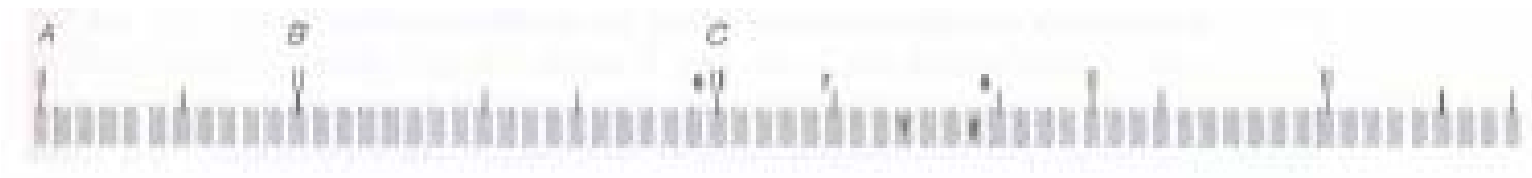
- Sıralı bir şekilde düzenlenmiş olan kayıtlara doğrudan erişimlerde veya doğrudan erişimli yapıya sahip kayıtlara sıralı erişimlerde performans artışı sağlar.

- Sıralı bir şekilde düzenlenmiş olan kayıtlara sıralı erişimlerde veya doğrudan erişimli yapıya sahip kayıtlara doğrudan erişimlerde performansı düşüktür.

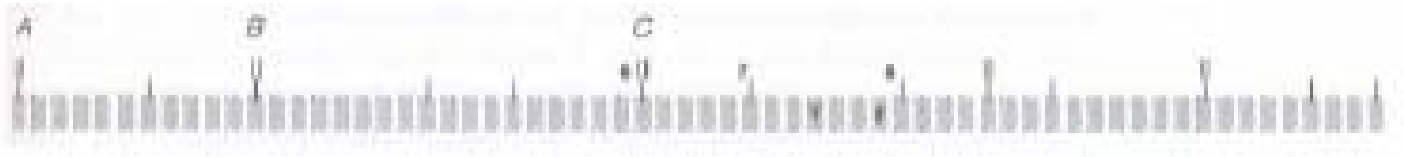
- Sıralı erişimli dosya yapısında hatırlanacağı gibi herhangi bir kayda ulaşmak istenildiğinde ilgili kayda ulaşılıncaya kadar kayıtlar üzerinde dolaşmak gerekir.

- Böyle bir yapıda arama süresini kısaltmak için kayıtlar üzerinde belli aralıklarla index noktaları (tablar) yerleştirilebilir. Böylelikle her bir kayıdı arama esnasında arama işlemine ilgili indexten itibaren

- Kayıt sayısının daha da arttığı durumlarda, aynı işlemi ikinci bir defa daha tekrarlamak mümkündür.



- Böyle bir yapıda istenilen bir kayda ulaşmak için gereken arama maliyeti azaltılmış olur ve etkililik sağlanır.



- Indexlemenin seviyesi kayıt sayısına bağlı olarak arttırılabilir.

- Indexleme görüldüğü üzere bir tür ağaç (Tree) yapısıdır.

- Ağaç yapısında arama işlemi kaydın bulunduğu dallar üzerinde gerçekleştirilir. Kaydın olmadığı dallar budanarak aramanın etkililiği arttırılır.

- Yeni bir s kaydının ekleneceği durumda ise kayıtların ötelenmesi gerekecektir. Oysaki taşma alanını (overflow) tanımlanması halinde buna gerek kalmayacaktır.

- Taşma alanı sayesinde ekleme işlemi sırasında ortaya çıkan performans kaybı kısmen azaltılmış olur.

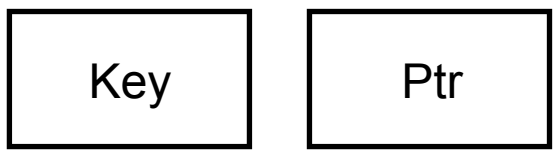
Uygulamada

- Blok adreslenebilir disklerde, birinci seviye indeks trackleri ve ikinci seviye indeks ise silindirleri gösterirler.
- Silindir indeks ilgili silindirdeki en yüksek anahtarı ve o silindir içindeki track indeksini tutar.
- Bir silindirdeki her bir track iki çift bilgi tutar. Birisi primary depolama alanıyla diğeri ise overflow alanıyla ilgilidir.
- || silindir indeksini ve | ise track indeksini göstermektedir.
- Indeksli sıralı şekilde yapılandırılan dosya organizasyonları ISAM (Indexed Sequential Access Method) olarak isimlendirilir.

- İndeks sıralı erişimli yapıyı kullanan blok adreslenebilir yapıdan olan ikincil depolama ünitesi olsun. Bu yapıda örneğimiz tek bir silindir üzerindeki kayıtlar üzerine yoğunlaşacağız.

- Silindir indeksi üzerinde birçok silindire ait olan pointer bilgileri de bulunur.

- Silindir indeksi üzerinde her bir silindir için bilgi içeren alanlar aşağıdaki şekildedir:

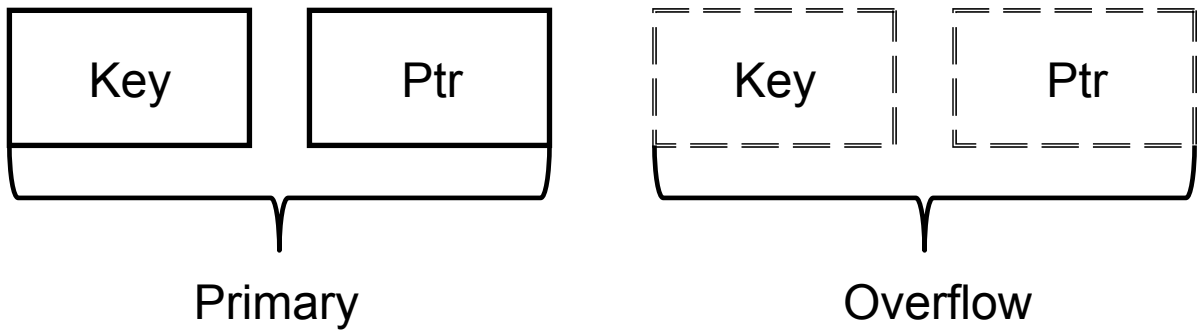


Silindir İndeksi İçin

- Anahtar değerleri silindir üzerindeki herhangi bir kaydın en yüksek değerli anahtarıdır.

- Pointer bilgisi ise o silindir için track indeksine işaret eder.

Bu şekilde bir çift birincil depolama alanı ile ilgili olarak bilgi barındırırken, diğer çift, iz üzerinden taşan kayırların bilgisini barındırır. Her bir track için olan görünüm ise;

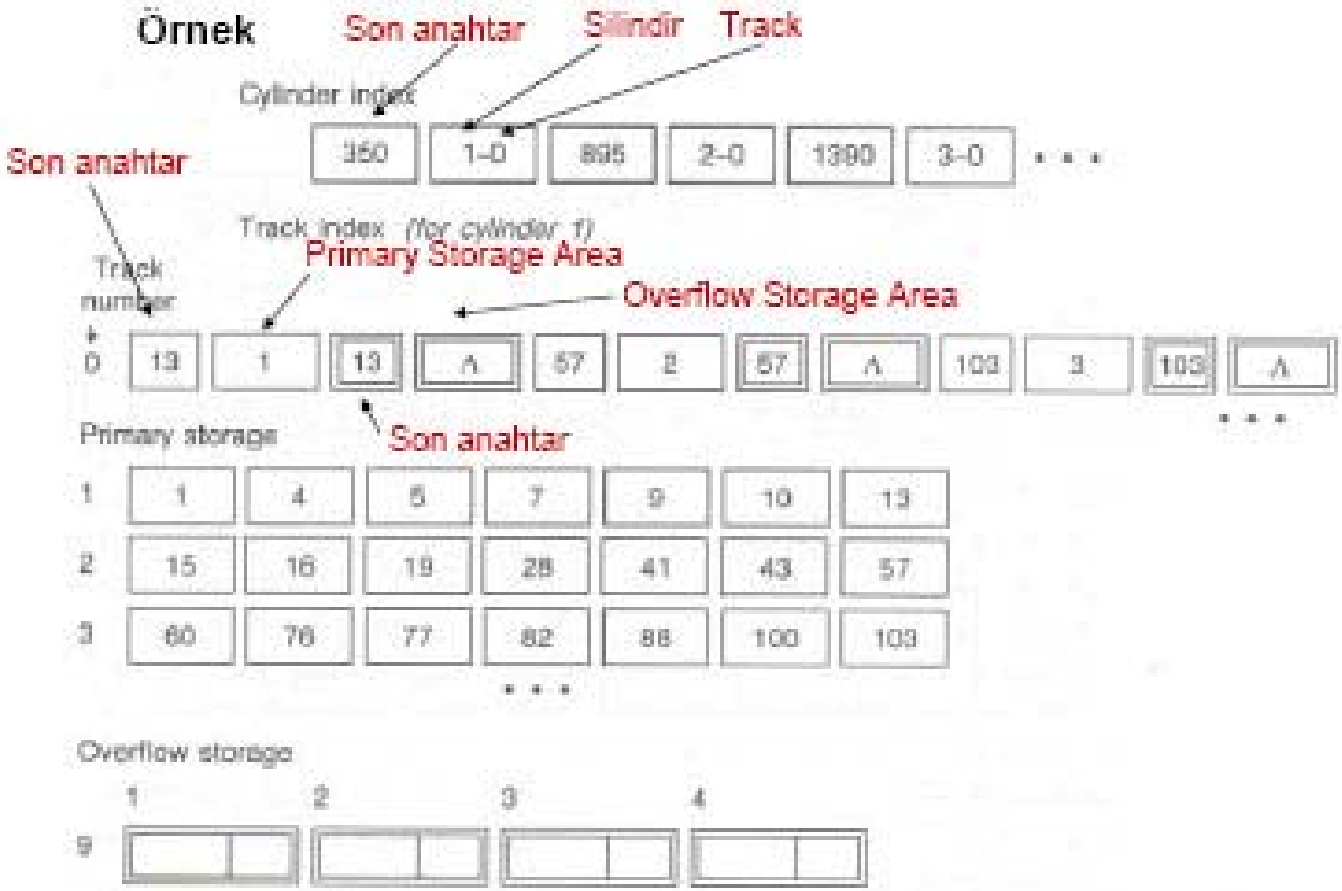


Track İndeksi İçin

Primary bölgedeki anahtar değeri, o track üzerindeki en büyük değerli anahtar değerini içerir. Overflow bölgedeki anahtar değeri ise, o track üzerindeki en büyük anahtar değerini içerir.

Primary bölgedeki pointer track üzerindeki birincil kayıtları, overflow bölgesindeki pointer ise varsa bu bölgedeki taşan kayda işaret eder.

Örnek


Örnek
Son anahtar
Silindir
Track
Cylinder index
Son anahtar
350
1-0
895
2-0
1380
3-0
Track index (for cylinder 1)
Primary Storage Area
Overflow Storage Area
Track number
0
13
1
13
A
57
2
87
A
103
3
103
A
Primary storage
Son anahtar
1
1
4
5
7
9
10
13
2
15
16
19
28
41
43
57
3
60
76
77
82
88
100
103
Overflow storage
1
2
3
4
9

- Silindir indeksi ilk 3 silindir için gösterilmiştir. Silindir sayısı daha fazla olabilir.

- Silindir indeksi pointer alanıda x-y şeklinde de ifade edilen gösterici bilgisinde; x o silindir için, silindir numarasını ve y ise yine o silindir için, track indeksinin bulunduğu track numarasını (0 nolu track) ifade eder.

- Dikkat edilirse primary ve overflow alanlarının anahtar değerleri aynıdır. Bunun nedeni herhangi bir overflow'un henüz meydana gelmemiş olmasıdır.

- ^ (Null) operatörü herhangi bir taşmanın henüz oluşmadığını belirtir.

- Yeni bir kayıt eklendiğinde sıradüzenselliğin bozulmaması için doğru pozisyona yerleştirilmesi gerekmektedir.

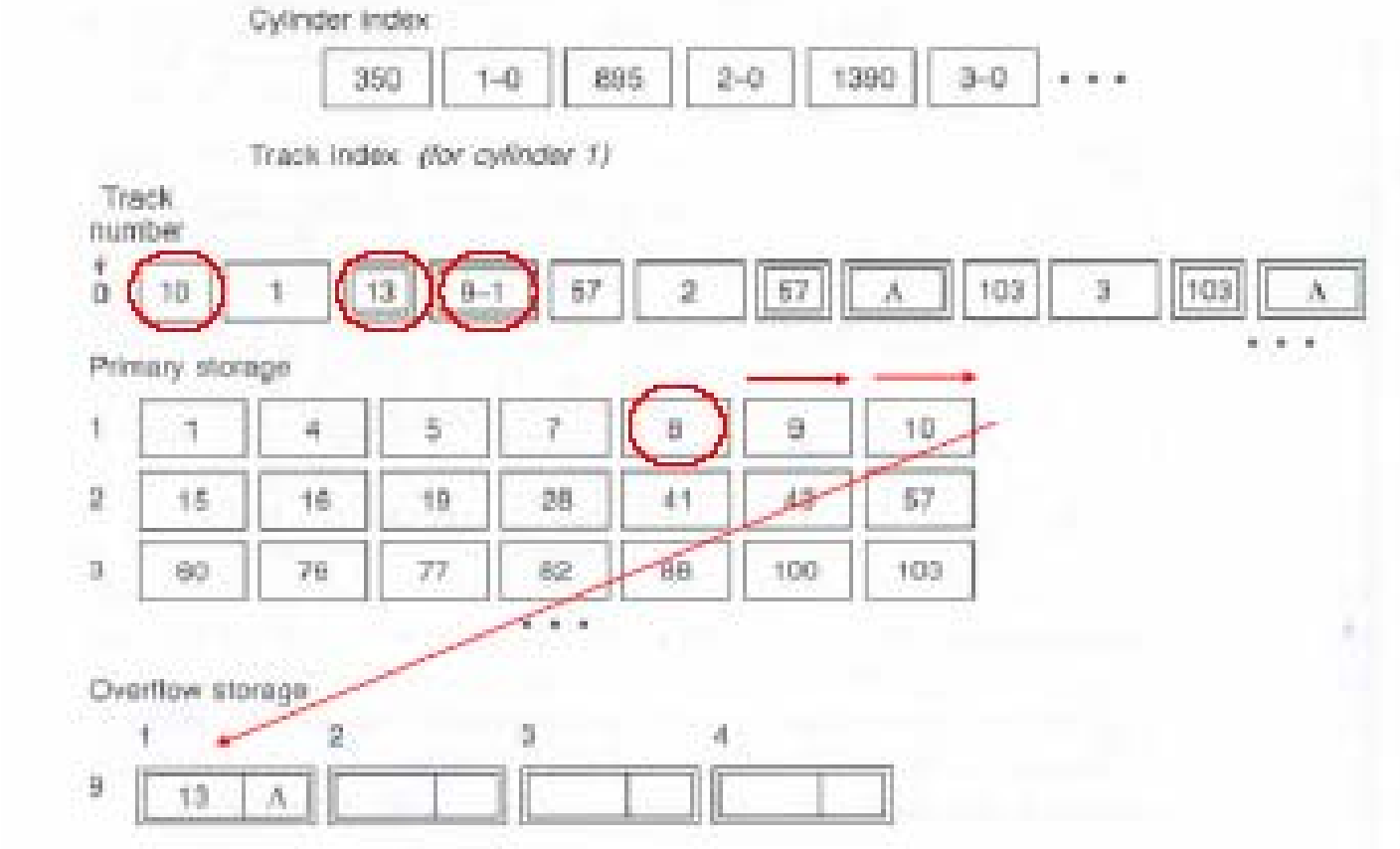
- Şimdi bu yapıya eklemeler yapalım. İlk olarak 8 kaydını ekleyelim

- İlk olarak yapılması gereken, bu kaydın dosya içerisinde olup olmadığının tespitinin yapılması ve eğer mevcut değilse hangi track'e yerleştirileceğinin belirlenmesidir.

- İkinci olarak silindir indeksi araştırılır. 8<350 olduğu için bu kaydın 1. silindir üzerinde olması gerekir. Silindir indeksi bizi 1.silindire ait olan track indeksine yönlendirir (0 nolu track).

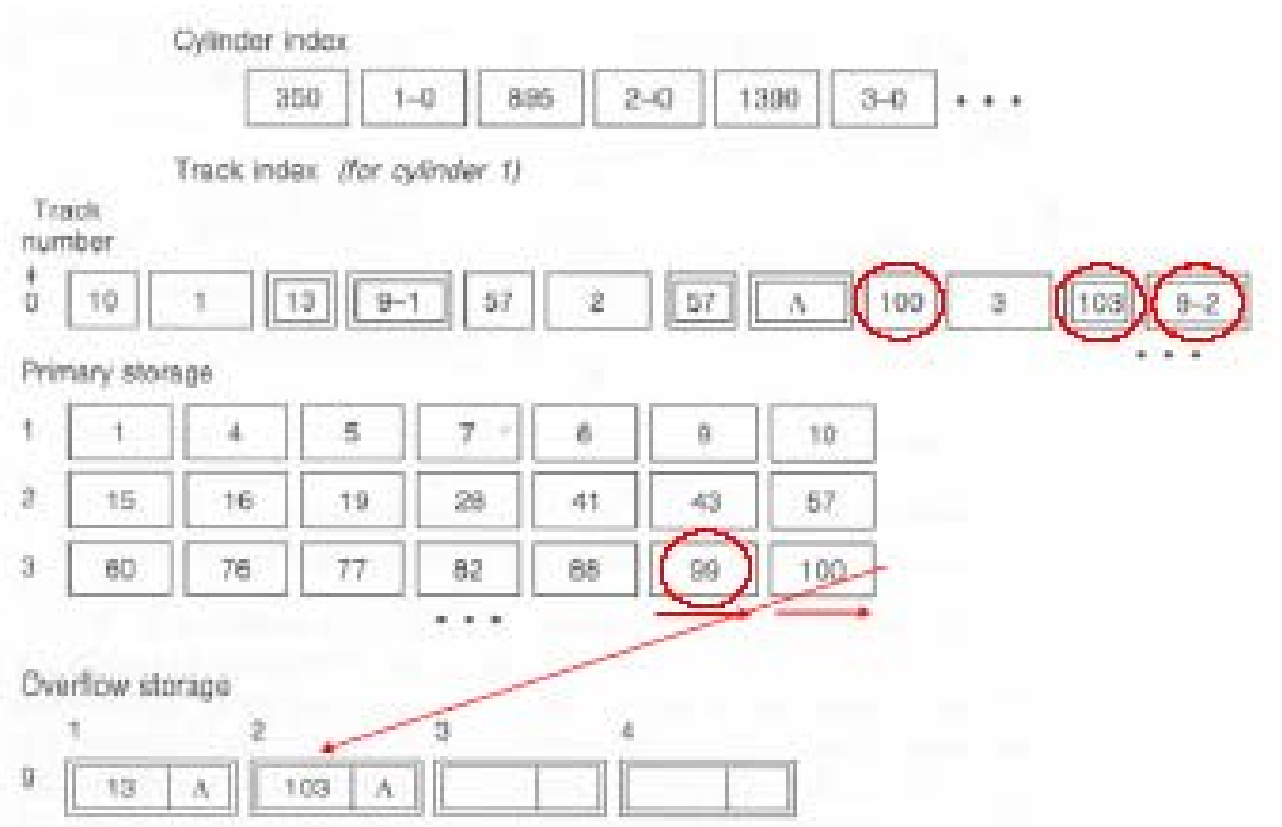
- Track indeksinde 8 kaydının hangi track'e yerleştirileceğini kararlaştırmak için karşılaştırma işlemi yapılır. 8<13 olduğu için 1. track üzerine yerleşecektir.

- Doğru pozisyona konumlandırmak için yerleştirilecek konumdan sonraki konumda bulunan tüm kayıtların birer kaydırılması gerekmektedir.
- Kaydırılacak olan kayıtlar 9, 10 ve 13'tür.



Kayıtlar kaydırıldıktan sonra güncellenen değerler kırmızı halka içerisinde gösterilmektedir.

İkinci olarak 99 kaydı eklenecektir. 99<350 olduğunda silindir 1 üzerine konumlandırlacaktır. 99<100 olduğundan track indeksi biri 3. track'e gönderir.



Bu durumda da yine kaydırma işlemi söz konusu olacaktır. Kırmızı halka içerisinde güncellenen kayıtlar görülmektedir.

- Üçüncü olarak ise 14 kaydı eklenecektir. Bu kayıt da 1. silindir üzerinde olacaktır. Burada önemli olan bu kaydın hangi track üzerinde yer alacağıdır.

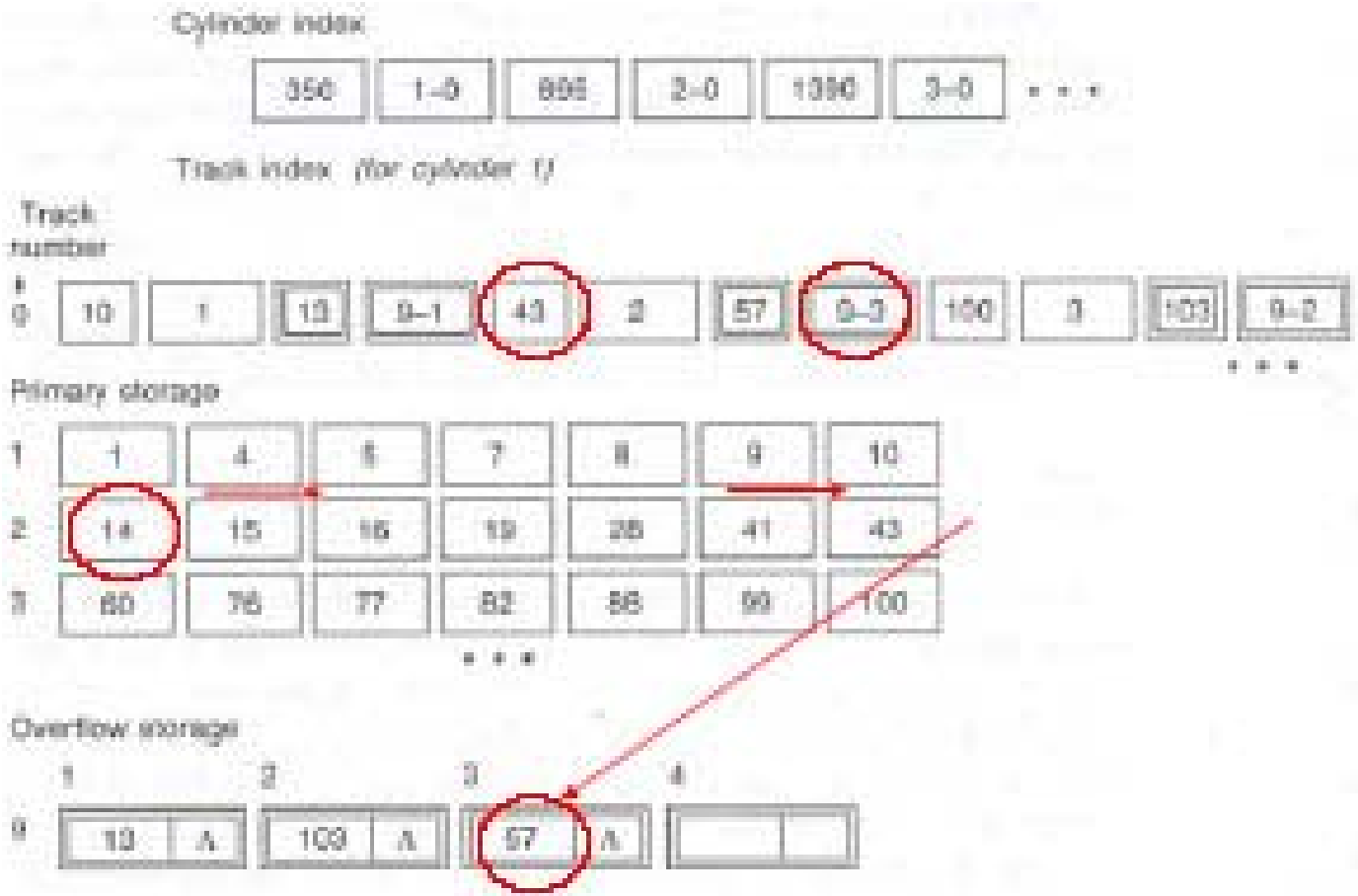
- Yerleşmesi gereken 2 nolu track'tir. Fakat bu track'e eklenmesi durumunda bütün kayıtların kaydırılması durumu söz konusudur.

- Peki onun yerine 1 nolu track'de overflow area içersine yerleşse idi ne olurdu? Bu durumda hiçbir kaydın ötelenmesine de gerek kalmazdı? Böyle bir şey olması anlamlı mıdır?

- Böyle bir yaklaşım her ne kadar anlamlı gibi görünse de tercih edilmez. Çünkü algoritmanın karmaşıklığını arttırmaktadır.

- Eğer bu yaklaşım tercih edilse idi, her ekleme işlemi sırasında 2 track'in birden araştırılması ve arasında seçim yapılması gerekirdi. Bu işlem ise ekleme işlemi sırasında daha fazla yük getirecek fakat retrieval'da ise o kadar da fazla bir performans artışı sağlamayacaktır.

- Ayrıca 14'ün eklenmesi gibi durumlar çok sık meydana gelmemektedir. Algoritmanın basitliğini korumak için bu yaklaşım tercih edilmez.

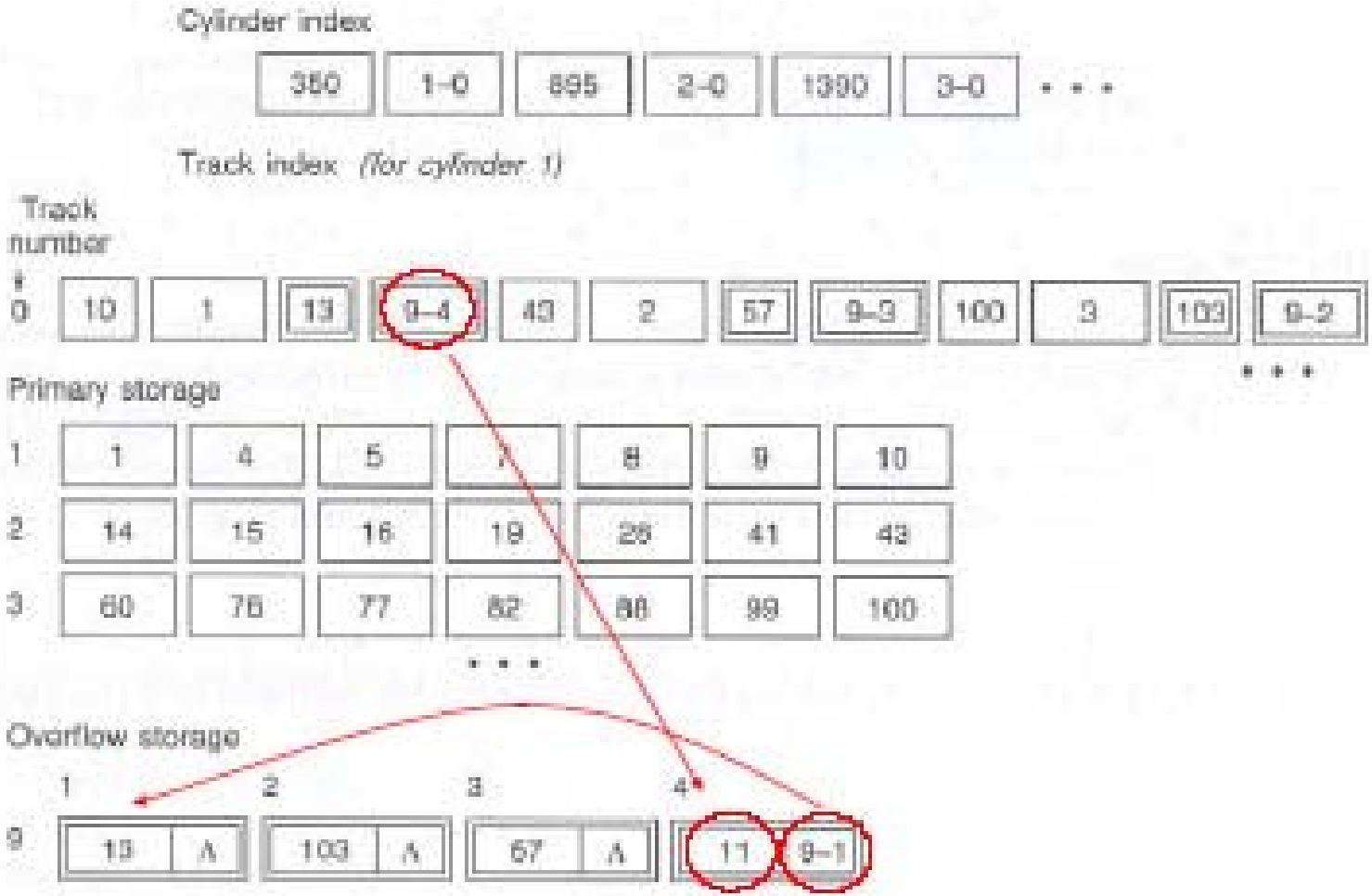


14 anahtarının eklenmesinden sonraki güncellenen değerler ile birlikte son durum.

- Dördüncü olarak ise 11 kaydı eklenecektir. Bu kayıt da 1. silindir üzerinde olacaktır. Ekleneceği track ise 1 nolu track'in overflow alanıdır. Peki bunu nasıl anladık?

- 1. track' ait olan kayıtlar içerisinde overflow area'da 13 kaydı bulunmaktadır ve 11<13 olacağından bu kaydında bu kuyruğa yerleştirilmesi gerekecektir.

- Burada yapılması gereken işlem sadece overflow alanındaki listenin pointer bilgilerini doğru bir şekilde güncellemektir.

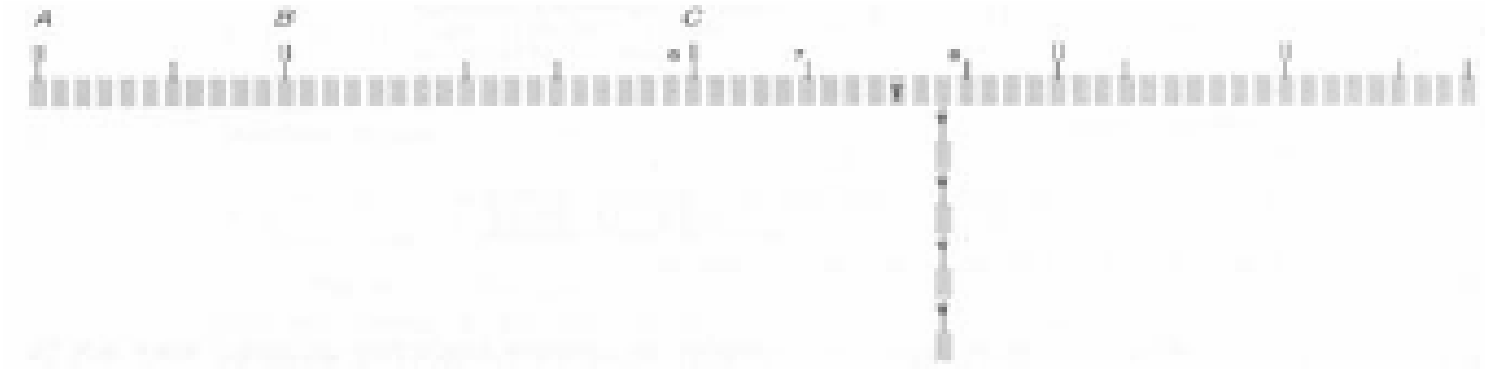

Cylinder index
350
1-0
895
2-0
1390
3-0
Track index (for cylinder 1)
Track number
0
10
1
13
9-4
43
2
57
9-3
100
3
103
9-2
Primary storage
1
1
4
5
8
9
10
2
14
15
16
19
26
41
43
3
60
76
77
82
88
99
100
Overflow storage
1
2
3
4
9
13
103
67
11
9-1

Genel Olarak Indeksli Sıralı Erişimli Dosyalar

- Dosya üzerinde başlayarak doğrudan okuma yapılmaz.
- Kayıtlar arasında sıralı erişim için silindir ve track bilgilerine ulaşılması gerekir.
- Indexed Sequential File yapısında sıralı erişim Direct Organized File yapısına göre daha hızlıdır ancak Sequential Organized File yapısına göre daha yavaştır
- Bir kayıt için index bilgilerine bakılarak erişim yapıldığı için Direct Organized File yapısına göre retrieval süresi büyüktür

Performans

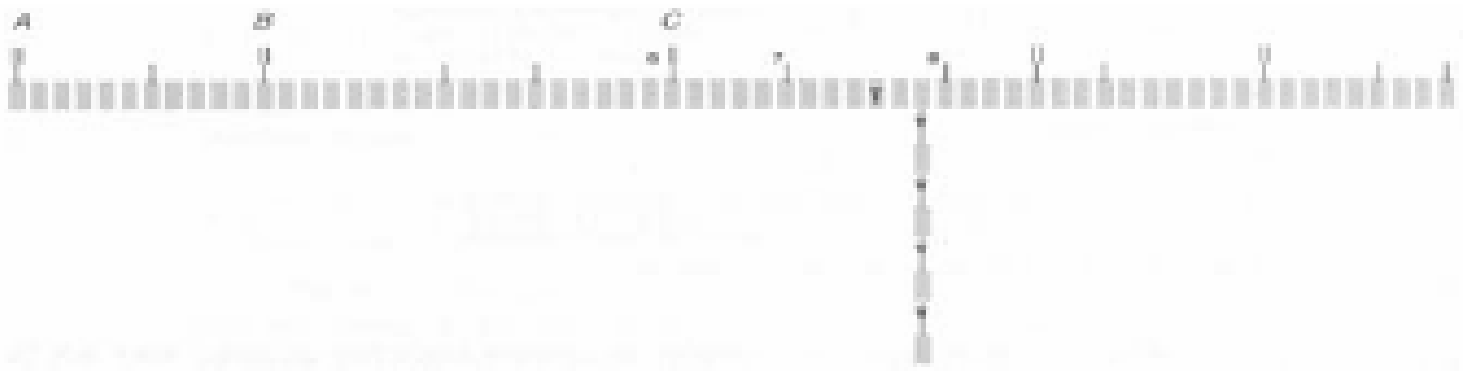
- Bir track üzerindeki overflow area içerisindeki kayıt sayısı arttıkça arama performansında düşüş görülür.



- Bu büyümeyi engellemek için periyodik olarak dosyanın yeniden organize edilmesi gerekir. Tüm bilgiler yeniden yerleştirilir.
- Zaman gerektiren böyle bir uygulama bilgisayarın kullanılmadığı zamanlarda gerçekleştirilir.

Kayıt Silme (Deletion)

- Silme işlemi, silinen kayıdın bulunduğu yere göre yapılır. Eğer silinen kayıt primary bölgede ise, silinen kaydın bulunduğu yere bir işaretçi (tombstone) koyulur veya boş kayıt yapılır.



- Silinen kayıt overflow bölgesinde ise, bağlı liste yapısı yeniden düzenlenir.

Taşma (Overflow)

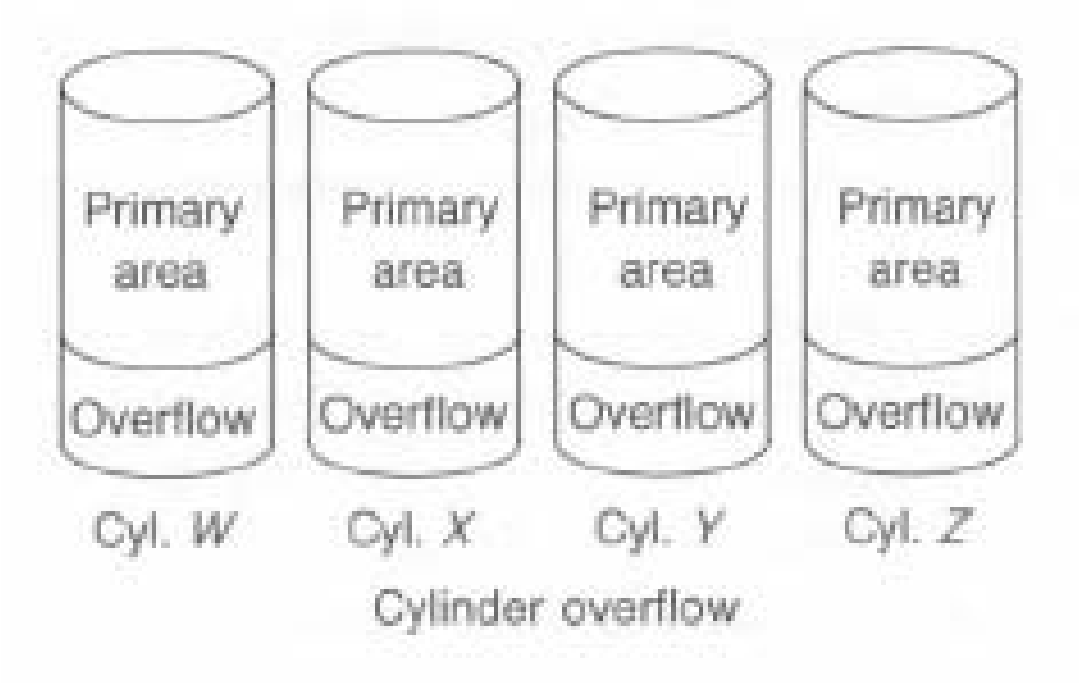
Overflow alanı iki farklı şekilde oluşturulabilir.

a)Cylinder Overflow

b)Independent Overflow

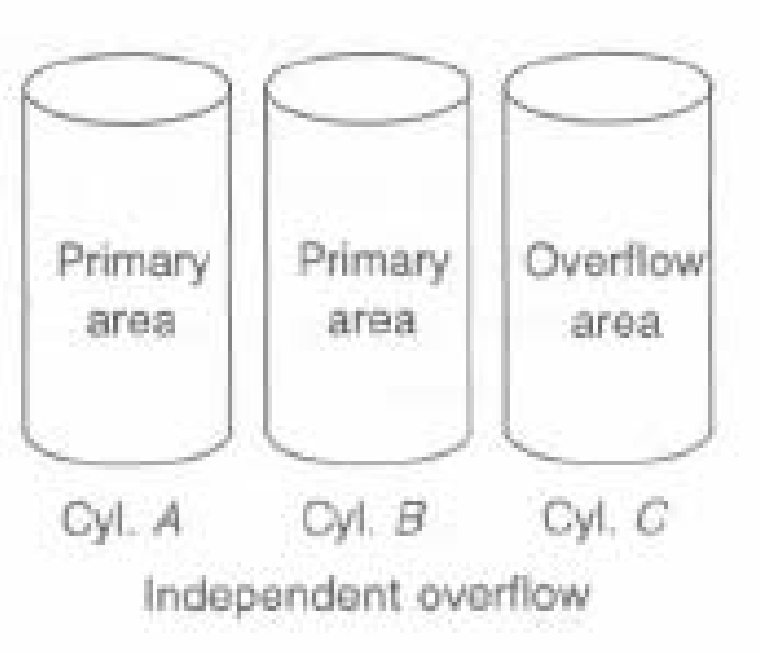
Cylinder Overflow

- Bu yapıda overflow area, primary area ile aynı silindir üzerinde yer almaktadır.



Independent Overflow

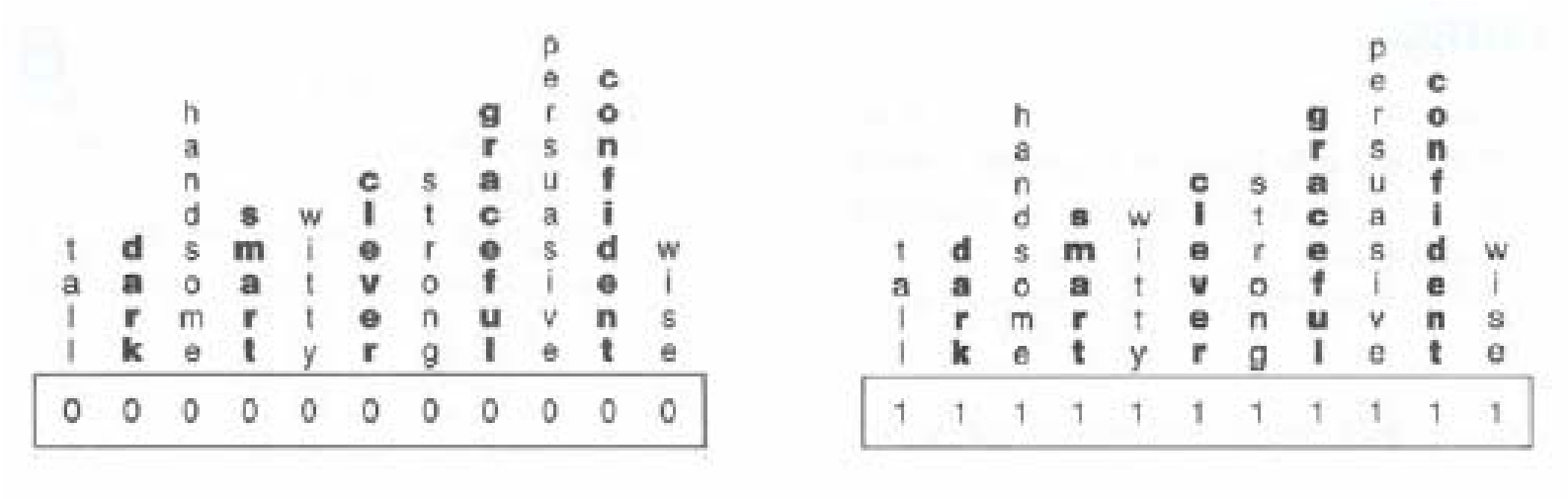
Bu yapıda, birden fazla sayıdaki primary area, tek bir silindir bölgesini overflow area olarak kullanmaktadır.



Bit Seviyesinde İşlemler
(Bit Level and Related Structures)

İkili Özellikler

- Bir binary özelliğin 2 farklı değer alması söz konusudur (1 veya 0).
- Bir kişiye ait özellikleri aşağıdaki gibi göstermek mümkündür.



- Bu yapıda, istenilen bir özelliğe ait sorgulama ilgili sütun üzerinden yapılabilir.

Örnek

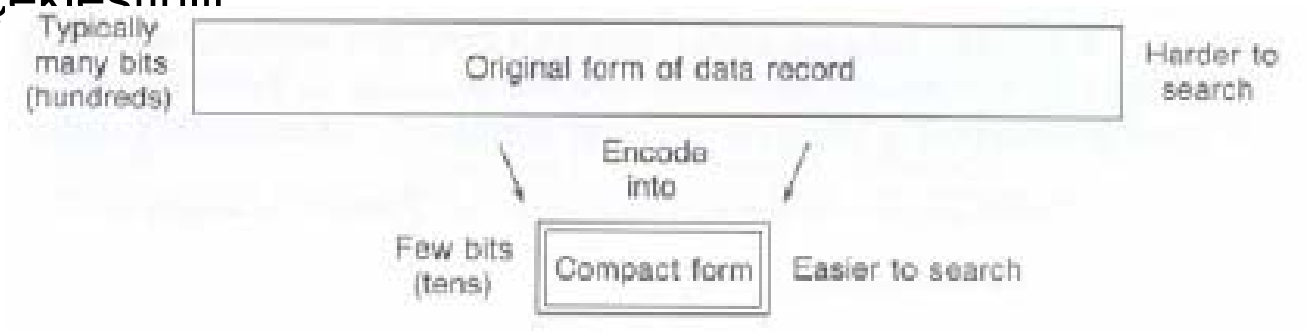
Bir restaurantda 8 farklı çeşit tatlı ve bunların bileşenlerini gösteren yapıyı aşağıda şekilde ifade etmek mümkündür.

	Ingredients																
Desserts	Baking powder	Blueberries	Brown sugar	Butter	Chocolate	Cinnamon	Eggs	Flour	Ice cream	Lemon juice	Milk	Nutmeg	Salt	Sugar	Vanilla	Water	Special ingredients
Banana sundaes	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	Bananas
Berry crumble	0	1	1	1	0	1	0	1	1	1	0	0	1	0	0	0	Oats
Chocolate toffee bars	1	0	1	1	1	0	1	1	0	0	0	0	1	0	1	0	Nuts
Custard	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	0	
Fresh apple pie	0	0	0	1	0	1	0	1	0	0	0	1	1	1	0	0	Apples
Glazed pound cake	1	0	1	0	0	0	1	1	0	0	1	0	1	1	1	0	Graham crackers
Peanut-fudge pudding cake	1	0	1	0	1	0	0	1	1	0	1	0	1	1	1	1	Peanut butter, vegetable oil
Southern blueberry pie	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	1	Cornmeal, cornstarch

Source: Betty Crocker's Microwave Cookbook, Random House, New York, Copyright © 1981. Adapted with permission.

Superimposed Coding

- Tatlılar ve bileşenleri için verilen örneğin daha kompak bir yapıda sunulması mümkündür. Bunun için superimposed kodlama gerçekleştirilir.



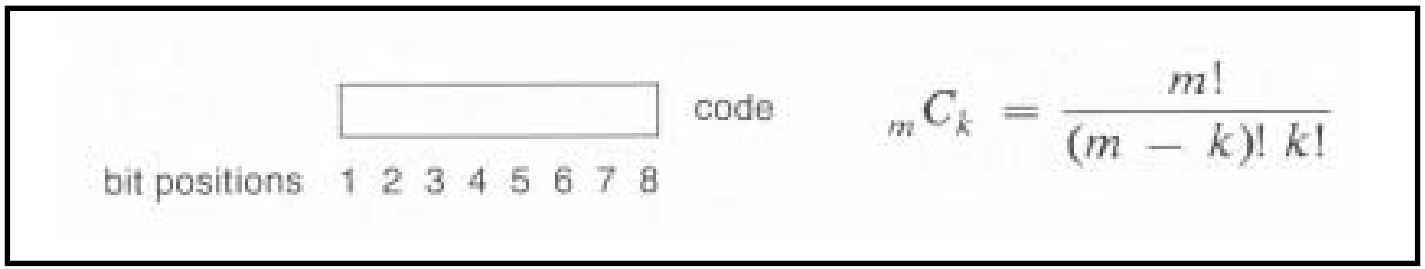
- Böyle bir yapı sayesinde, bilgi daha kolay ve hızlı bir şekilde işlenebilir.

Desserts	Baking powder	Blueberries	Brown sugar	Butter	Chocolate	Cinnamon	Eggs	Flour	Ice cream	Lemon juice	Milk	Numeg	Salt	Sugar	Vanilla	Water	Special ingredients
Banana sundaes	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	Bananas
Berry crumble	0	1	1	1	0	1	0	1	1	1	0	0	1	0	0	0	Oats
Chocolate toffee bars	1	0	1	1	1	0	1	1	0	0	0	0	1	0	1	0	Nuts
Custard	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	0	
Fresh apple pie	0	0	0	1	0	1	0	1	0	0	0	1	1	1	0	0	Apples
Glazed pound cake	1	0	1	0	0	0	1	1	0	0	1	0	1	1	1	0	Graham crackers
Peanut-fudge pudding cake	1	0	1	0	1	0	0	1	1	0	1	0	1	1	1	1	Peanut butter, vegetable oil
Southern blueberry pie	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	1	Cornmeal, cornstarch

Source: *Betty Crocker's Microwave Cookbook*, Random House, New York, Copyright © 1981. Adapted with permission.

Bu tabloda, her bir kayıt 16 bit ile ifade edilmektedir. Kolaylık ve işlem hızını arttırmak açısından her bir kaydı daha az sayıda bit ile temsil etmek mümkündür. Bu da superimposed-coding ile gerçekleştirilir.

Bu tabloda, her bir kaydı 8 bit ile temsil etmek istersek kullanacağımız yapı;



şeklinde olacaktır. Burada;

m= toplam bit sayıdır (8).
k = toplam bit sayısı içerisinde kullanılacak 1 sayısıdır.
(Örn: k=1 için toplam farklı kod sayısı 8, k=2 için 28'dir.

Bu yaklaşım her bir kayda uygulandığında bileşenler için oluşturulan kodlar;

Apples	00100100	Ice cream	01000001
Bananas	00100010	Lemon juice	00100001
Blueberries	10000001	Nutmeg	00010001
Brown sugar	01000010	Nuts	01000100
Chocolate	10000100	Oats	00010010
Cinnamon	00011000	Peanut butter	01001000
Cornmeal	00101000	Vanilla	10000010
Graham crackers	00010100		

Genellikle tüm kayıtlarda kullanılan tüm ortak özellikler bu kodlama içerisinde yer almayabilir. Böylelikle retrieval süresi azaltılmış olur.

- Bir kaydın sayısal değeri her bir özelliğinin kodlarının OR işleminden geçirilmesi ile elde edilir.

Brown sugar	01000010
Chocolate	10000100
Ice cream	01000001
Vanilla	10000010
Peanut butter	01001000
Peanut-fudge pudding cake	11001111

- Bir bileşimin bileşenlerini bulmak için, bileşen kodlarında 1 olan pozisyonlara bakılır.

- Bir bileşenin bulunduğu bileşim kodlarını bulmak için de bileşimde 1 olan pozisyonlara bakılır.

Örn: İçerisinde Chocolate olan bileşimlere bakılırsa, Chocolate Toffee Bars, Glazed Bound Cake ve Peanut-Fudge Pudding Cake olduğu görülür. Halbuki orijinal tabloya bakıldığında ise, Glazed Bound Cake'in Chocolate içermediği görülür.

	Ingredients																
Desserts	Baking powder	Blueberries	Brown sugar	Butter	Chocolate	Cinnamon	Eggs	Flour	Cream	Ice	Lemon juice	Milk	Nutmeg	Salt	Sugar	Vanilla	Water	Special ingredients
Banana sundaes	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0		Bananas
Berry crumble	0	1	1	1	0	1	0	1	1	1	0	0	1	0	0	0		Oats
Chocolate toffee bars	1	0	1	1	1	0	1	1	0	0	0	0	1	0	1	0		Nuts
Custard	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	0		
Fresh apple pie	0	0	0	1	0	1	0	1	0	0	0	1	1	1	0	0		Apples
Glazed pound cake	1	0	1	0	0	0	1	1	0	0	1	0	1	1	1	0		Graham crackers
Peanut-fudge pudding cake	1	0	1	0	1	0	0	1	1	0	1	0	1	1	1	1		Peanut butter, vegetable oil
Southern blueberry pie	0	1	0	0	0	1	0	0	0	1	0	0	1	1	0	1		Cornmeal, cornstarch

Source: Betty Crocker's Microwave Cookbook, Random House, New York, Copyright © 1981. Adapted with permission.

Apples	00100100	Ice cream	01000001
Bananas	00100010	Lemon juice	00100001
Blueberries	10000001	Nutmeg	00010001
Brown sugar	01000010	Nuts	01000100
Chocolate	10000100	Oats	00010010
Cinnamon	00011000	Peanut butter	01001000
Cornmeal	00101000	Vanilla	10000010
Graham crackers	00010100		

- Bu kodlama sonucunda, kullanılan bit sayısına ve kodlamaya bağlı olarak bilgi kaybı (false drop) olabilmektedir. Ancak bu yapının en önemli avantajı retrieval süresini oldukça kısaltmasıdır.

- Her kayıt için 8 bitlik bir kodlama yeterlidir.

TABLE 5.3 SUPERIMPOSED RECIPE CODES

Banana sundaes	01111011	Fresh apple pie	00111101
Berry crumble	11111011	Glazed pound cake	11010110
Chocolate toffee bars	11000110	Peanut-fudge pudding cake	11001111
Custard	10010011	Southern blueberry pie	10111001

Genel Olarak

- Superimposed coding yüzlerce bitlik bilgiyi daha kısa bir şekilde ifade eder.
- Bilginin retrieval süresini kısaltır.
- False Drop olan kayıtların tespit edilebilmesi için orijinal kodların da saklanması gerekir.
- False Drop sayısı kullanılan bit sayısı arttırılarak azaltmak mümkündür.
- False Dropları azaltmak için k değerini arttırmak etkisizdir. Aksine false dropları arttırabilir.

Metin Üzerinde Arama

- Naive metin arama algoritması aranan desen ile aranılacak string arasında baştan sona kadar bir karşılaştırma yapar. Örn:

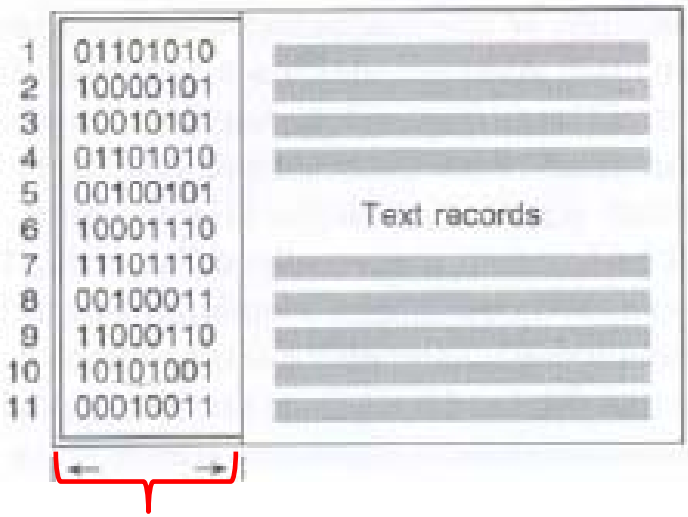
```
string   Problem solving is a common paradigm of computer science

pattern  computer
```

- Böyle bir yapıda toplamda 50 karşılaştırma yapılmaktadır. Arama yapan çeşitli algoritmalar mevcuttur.

```
Problem solving is a common paradigm of computer science.      Number of comparisons

computer                                                              1
 computer                                                             2
  computer                                                            3
   .  .  .

                    computer                                          22 - 24
                     computer                                         25
                      computer                                        26
                       .  .  .

                                          computer                    43 - 50

     [The _ represents the pattern symbol being compared.]
```

İmzalar (Signatures)

- Arama işleminde, aramayı metnin tümünde yapmak yerine belli bir kısım üzerinde gerçekleştirmek hızı arttıracaktır.
- Bunun için metinde yer alan satırlar ve paragraflar için birer imza oluşturulur.
- Öncelikle imza bilgisine göre aranan string'in olup olmadığı belirlenir ve daha sonra arama algoritmaları ile metin üzerinde arama yapılır.



- k adet gruplanmış yanyana sembolün hash fonksiyonuyla m boyutundaki signature içerisinde ilgili pozisyona aktarılabilir.

string Problem solving is a common paradigm of computer science

if $k = 2$ then all 2-symbol pairs are hashed, that is,

h(Pr), h(ro), h(ob), h(bl), h(le), h(em), h(m▒), . . . , h(e.)

where h is the hashing function and ▒ represents a blank symbol.

- Örn:

0 0 1 0 1 . . . 0 0 1 0 0 1	Text signature

- Hash (10010101) =3 olursa tabldan sadece 3 .satıra bakılmalıdır.