

Veritabanı Yönetim Sistemleri (335)

Yrd.Doç.Dr. Ahmet Arif AYDIN

Dosya Yapıları ve indexleme

(storage and indexing)

Giriş

- Veritabanı yönetimi sistemleri
 - Veri bütünlüğü
 - Veri bağımsızlığı
 - verilere etkili bir biçimde erişimi
 - verinin yönetimini
 - Veri güvenliğini sağlamaktadır.

Giriş

- Bu derste işlenecek konular
 - Veritabanı içerisinde depolanan kayıtların yönetimi
 - Index yapıları

Örnek

- Bir dosya içerisinde çalışanların yaş , isim , ve maaş bilgileri bulunsun.
- Çalışan bilgisini alabilmek için bilgiler yas alanına göre artan sıralama yapılabilir.
- Eğer dosya içeriği
 - sıklıkla değişiyorsa
 - yeni veriler ekleniyorsa istenilen sıralamayı sağlamak maliyetli olur.
- yaş alanından başka diğer alanlardaki bilgileri almak istendiğinde istenilen bilgiye ulaşılması için **bütün kayıtların taranması** gerekmektedir.
 - Örneğin maaş 'ı 2000 ve üzeri olan çalışanların listesi

Kayıt koleksiyonlarına bulunan birden fazla alanlara erişimi sağlayan indexing tekniğidir.

PostgreSQL limitler

Limit	Value
Maximum Database Size	Sınırsız
Maximum Table Size	32 TB
Maximum Row Size	1.6 TB
Maximum Field Size	1 GB
Maximum Rows per Table	Sınırsız
Maximum Columns per Table	250 - 1600 kolonun tipine bağlı
Maximum Indexes per Table	Sınırsız

Verinin Diske Kaydedilmesi

- VTYS çok çeşitli veriyi disk üzerine kaydetmektedir.
- Üzerinde işlem yapılması gereken veri hafızaya alınmaktadır.
- Verinin **diske yazıldığı** ve sonrasında ihtiyaç halinde okunduğu **her bir birim sayfa (page)** olarak adlandırılır.
- Veritabanının parametresi Sayfa (page) dır.
- PostgreSQL 'in sayfa kapasitesi **8192 bytes (8 KB)** dır.

Verinin Diske Kaydedilmesi

Her bir satırında text ve integer alanı bulunan ve 100.000 satırdan oluşan veri koleksiyonu PostgreSQL de kaydedelim

24 bytes: herbir satır başlığı (ortalama)

24 bytes: bir integer ve text

4 bytes: pointer on page to tuple

52 bytes (1satır)

**Bir satırın
kapladığı
alan**

8192 bytes

/

52 bytes

158 satır

**Bir sayfanın
depolayacağı
Satır sayısı**

100.000

/

158

633

**Page (sayfa)
sayısı**

Verinin Diske Kaydedilmesi

Page I/O (Sayfa Giriş / Çıkış)

- Veritabanında bulunan kayıtlara erişimi sağlamaktadır. (diskten hafızaya - hafızadan diske)
- Gerçekleştirilecek olan sorguların ve işlemlerinin maliyetini birinci dereceden etkilemektedir.
- Page I /O nun maliyetini düşürmek için optimizasyon çalışmaları yapılmaktadır.

Verinin Diske Kaydedilmesi

- Veritabanı dosya yapısıyla beraber kaydedilen verinin istenilen sayfasını sabit maliyetle okumaya imkan sağlar
- Belirli bir sıra ile kaydedilen sayfalara erişmek, karışık olan sayfalara erişip okumaktan daha az maliyetlidir.
- Disk' de kayıtlı sayfaların her birini diğerinden ayırt etmeyi sağlayan **recordid (rid)** 'si bulunmaktadır.
- **rid** sayfanın kayıtlı olduğu adres bilgisini içermektedir.

Buffer Manager

- Diskte kayıtlı olan veriyi hafıza getiren ve diske kayıt işlemi gerçekleştiren yazılım katmanı **buffer manager** (ara bellek yöneticisi) olarak isimlendirilir.
- **File layer** sayfalara erişim isteğini **buffer manager**'dan ister.
- Hard Disk üzerindeki alan **disk space manager** tarafından yönetilmektedir.
- Yeni bir sayfa (page)' in kaydedilmesi için File layer **disk space manager** dan alan ister.
- File layer da bir page silindiğinde **disk space manager** bırakılan alanı kullanıma açık alana ekler.

Dosya Organizasyonları ve indexleme

- Veritabanında bulunan kayıtlar sayfalar (page) halinde dosya (file) içerisinde saklanır.
- Bir dosya sayfalardan oluşur ve insert, delete , scan işlemlerini destekler
- File layer sayfa yönetimini sağlar.

Dosya Organizasyonları ve indexleme

İndex

- Disk üzerinde kayıtlı olan dosyaları üzerindeki gerçekleştirilecek okuma işlemlerini optimize eder.
- Arama kriterlerine uyan sonuçlar index yardımıyla alınır.
- İndex aşağıdaki biçimde tanımlanabilir
 1. key^* (arama anahtarı ve veri)
 2. (Key , rid)
 3. (Key , rid-list)

Dosya Organizasyonları ve indexleme

Clustered Indexing

Bir dosyada bulunan kayıtların sıralanması index yapısındaki sıralamasına yakınsa bu durum clustered index olarak ifade edilir.

1. **key*** (arama anahtarı ve veri) **Clustered**
2. (Key , rid) **Clustered veya random**
3. (Key , rid-list)

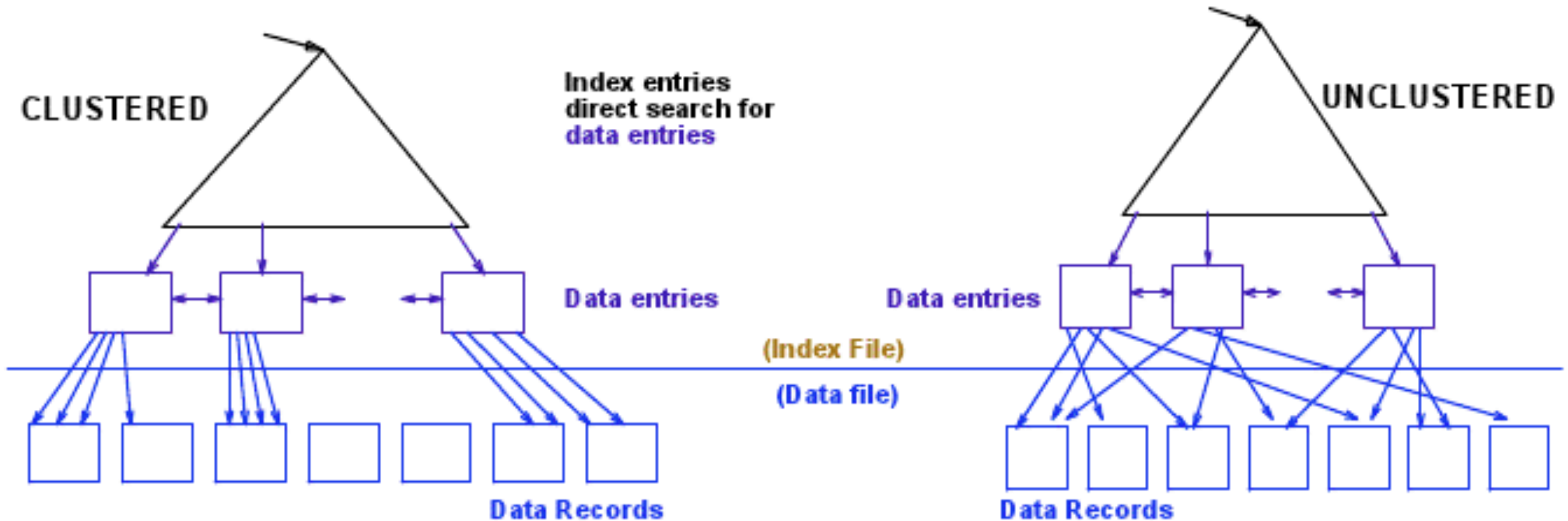
Pratikte 1 clustered 2 ve 3 unclustered olarak tanımlanır

Dosya Organizasyonları ve indexleme

Clustered Indexing

- Index yapısı clustered index olduğu zaman istenilen cevap birbiri ardınca sıralanan sayfa(lar) içerisindedir.
- Unclustered olunca bütün kayıtların incelenmesi gerekmektedir.

Dosya Organizasyonları ve indexleme



Dosya Organizasyonları ve indexleme

Primary and Secondary Indexes

- Primary index
 - Tablo oluştururken tanımladığımız Primary key olan alandır
 - Unique değerler içerir
- Secondary index
 - Foreign Key
 - Unique değerler içermez

index Veri Yapısı

İki yöntemle tanımlanabilir

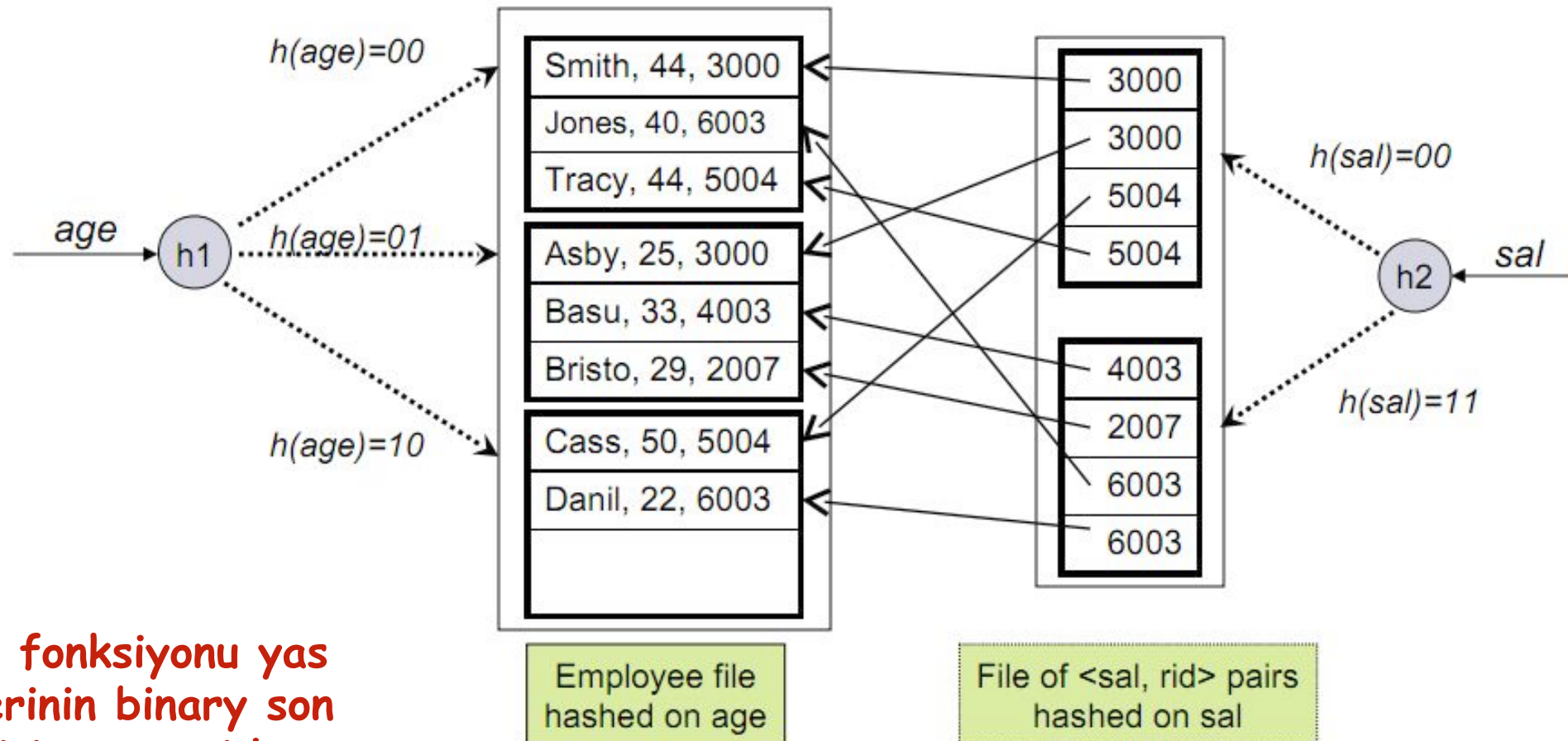
1. Hash data entries
2. Tree structure

index Veri Yapısı

Hash Data Entries

- Kayıtlar anahtar değere göre gruplandırılır
- Her bir alt grup bucket (kova) olarak isimlendirilir.
- Hash fonksiyonunun değerine göre guruplara ayırma işlemi gerçekleşir.
- Yeni değerler eklenirken hash fonksiyonuna göre bucket belirlenir.

Hash Based Indexing



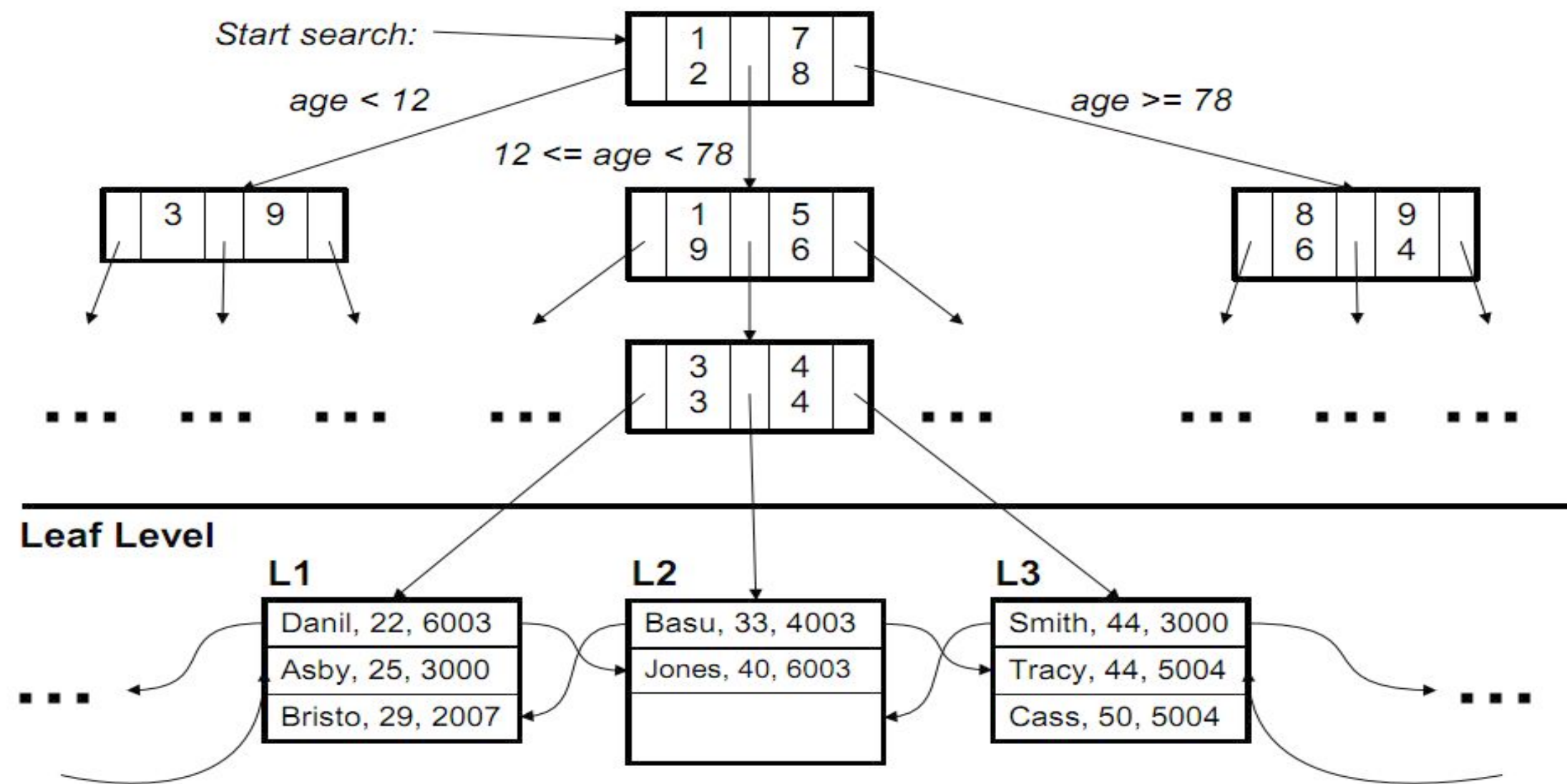
Hash fonksiyonu yas
değerinin binary son
iki bitine göre işlem
gerçekleştiriyor

index Veri Yapısı

Tree based indexing

- Hash yapısının alternatifidir.
- Belirlenen alana göre sıralama ve gruplandırma ile ağaç yapısı oluşturulur.
- Bu yapı ile beraber aramalar belirlenen aralıkta gerçekleştirilir.

Tree based indexing



PostgreSQL Index Yapıları

- B+ Tree (Balanced) Default olarak oluşturulan index yapısıdır
- Hash `CREATE INDEX name ON table USING hash (column);`
- GIST `CREATE INDEX name ON table USING gist(column);`
- GIN `CREATE INDEX name ON table USING gin(column);`

Index Yapılarının Karşılaştırılması

File of randomly ordered employee records, or heap file.

File of employee records sorted on *(age, sal)*.

Clustered B+ tree file with search key *(age, sal)*.

Heap file with an unclustered B+ tree index on *(age, sal)*.

Heap file with an unclustered hash index on *(age, sal)*.

Index Yapılarının Karşılaştırılması

- Sıralı olmayan dosya (file unsorted)
 - Depolama etkili
 - Etkili ve hızlı kayıt
 - Hızlı tarama
 - Arama ve silme işlemlerinde yavaş

Index Yapılarının Karşılaştırılması

- Sıralı dosya (sorted file)
 - Etkili depolama
 - kayıt ekleme ve kayıt silme yavaş
 - Arama sıralı olmayan dosyadan dan hızlı

Index Yapılarının Karşılaştırılması

- **Clustered file**
 - Sıralanmış dosyaların avanlajları bulunmaktadır
 - kayıt ekleme ve kayıt silme etkili
 - Aramalar sıralı olan dosyalardan hızlı

Index Yapılarının Karşılaştırılması

- **Unclustered Hash ve Tree yapılarında**
 - kayıt ekleme ve kayıt silme etkili
 - Aramalar hızlı
 - Tarama ve aralık bulmada yavaş

Index Yapılarının Karşılaştırılması

<i>File Type</i>	<i>Scan</i>	<i>Equality Search</i>	<i>Range Search</i>	<i>Insert</i>	<i>Delete</i>
Heap	BD	$0.5BD$	BD	$2D$	$Search + D$
Sorted	BD	$D \log_2 B$	$D \log_2 B + \# \text{ matching pages}$	$Search + BD$	$Search + BD$
Clustered	$1.5BD$	$D \log F 1.5B$	$D \log F 1.5B + \# \text{ matching pages}$	$Search + D$	$Search + D$
Unclustered tree index	$BD(R + 0.15)$	$D(1 + \log F 0.15B)$	$D(\log F 0.15B + \# \text{ matching records})$	$D(3 + \log F 0.15B)$	$Search + 2D$
Unclustered hash index	$BD(R + 0.125)$	$2D$	BD	$4D$	$Search + 2D$

B: sayfa sayısı , D: okuma yazma zamanı. C: kayıt işleme zamanı H: hash fonksiyonu,