Veritabanı Yönetim Sistemleri

Dr. Öğr. Üyesi Ahmet Arif AYDIN

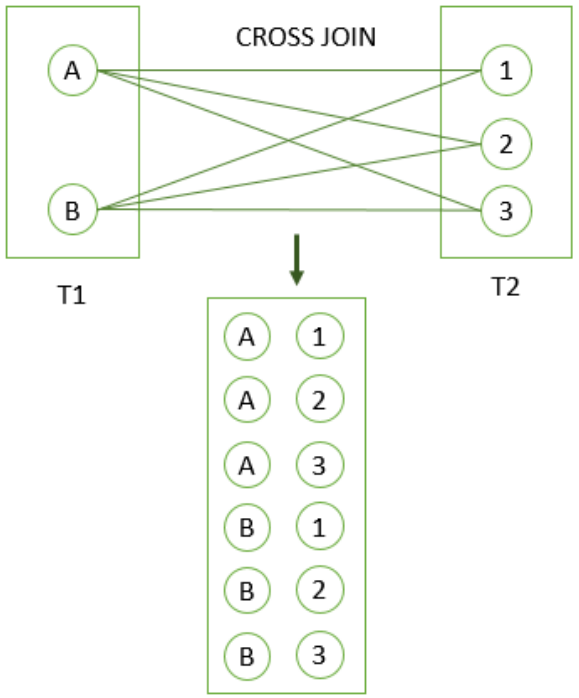
Dosya Yapıları ve İndexleme (storage & indexing)

Sorular

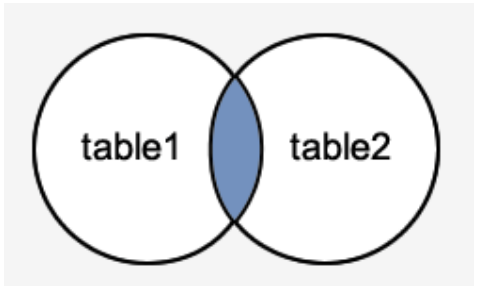
SQL-3

- Group By - Having
- Kısıtlamalar
  - Check, Unique, Not NULL
- Join
  - Inner, Left , Right
  - Natural

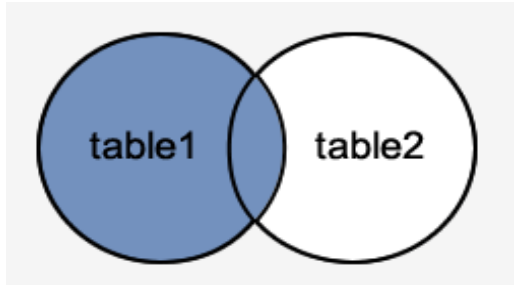
Özet



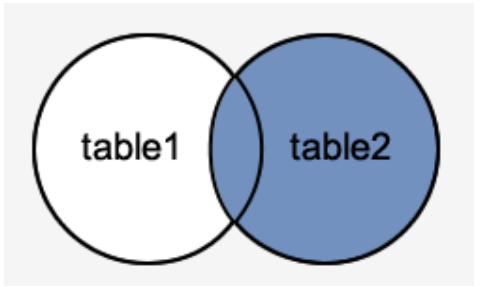
inner



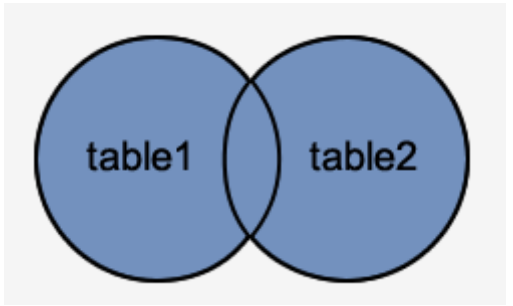
left



right



Full outer



SELECT * FROM T1 NATURAL [**INNER**, LEFT, RIGHT] JOIN T2;

Ders Takvimi

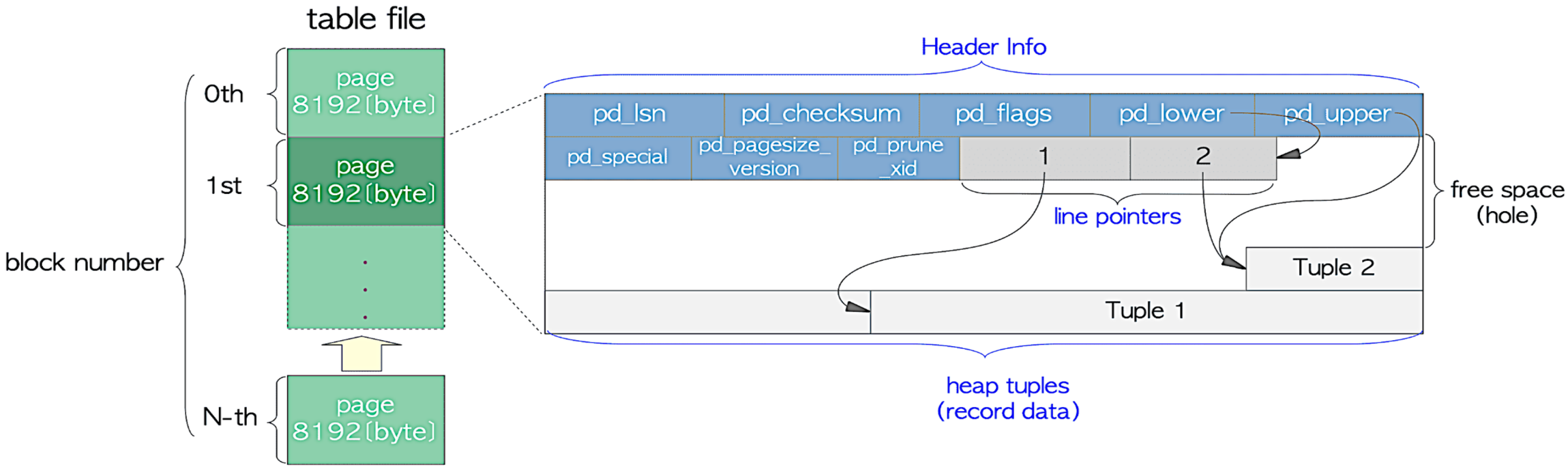
Konular
Veritabanı Tasarımı
ER-Model
UML
Relational Model
İlişkisel Cebir
İlişkisel Hesap
SQL (1-2-3)
Dosya Yapıları ve İndexleme
Sorgu Optimizasyonu
Hareket Yönetimi
Kilitlenmeler
Concurrency
Veritabanı Kurtarma Teknikleri

Veritabanı yönetimi sistemleri verinin

- bütünlüğü (consistency)
- bağımsızlığı (data independence)
- etkili bir biçimde erişimi (access)
- yönetimi (management)
- güvenliğini (security)

sağlamaktadır.

Verinin Diske Kaydedilmesi



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

Verinin Diske Kaydedilmesi

Her bir satırında text ve integer alanları bulunan ve 100.000 satırdan oluşan bir veri koleksiyonunu PostgreSQL de kaydedelim.

Verinin Diske Kaydedilmesi

Her bir satırında text ve integer alanları bulunan ve 100.000 satırdan oluşan bir veri koleksiyonunu 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

Verinin Diske Kaydedilmesi

Her bir satırında text ve integer alanları bulunan ve 100.000 satırdan oluşan bir veri koleksiyonunu 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 sayfada depolanacak olan Satır sayısı

Verinin Diske Kaydedilmesi

Her bir satırında text ve integer alanları bulunan ve 100.000 satırdan oluşan bir veri koleksiyonunu 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 sayfada depolanacak olan Satır sayısı

100.000
/
158

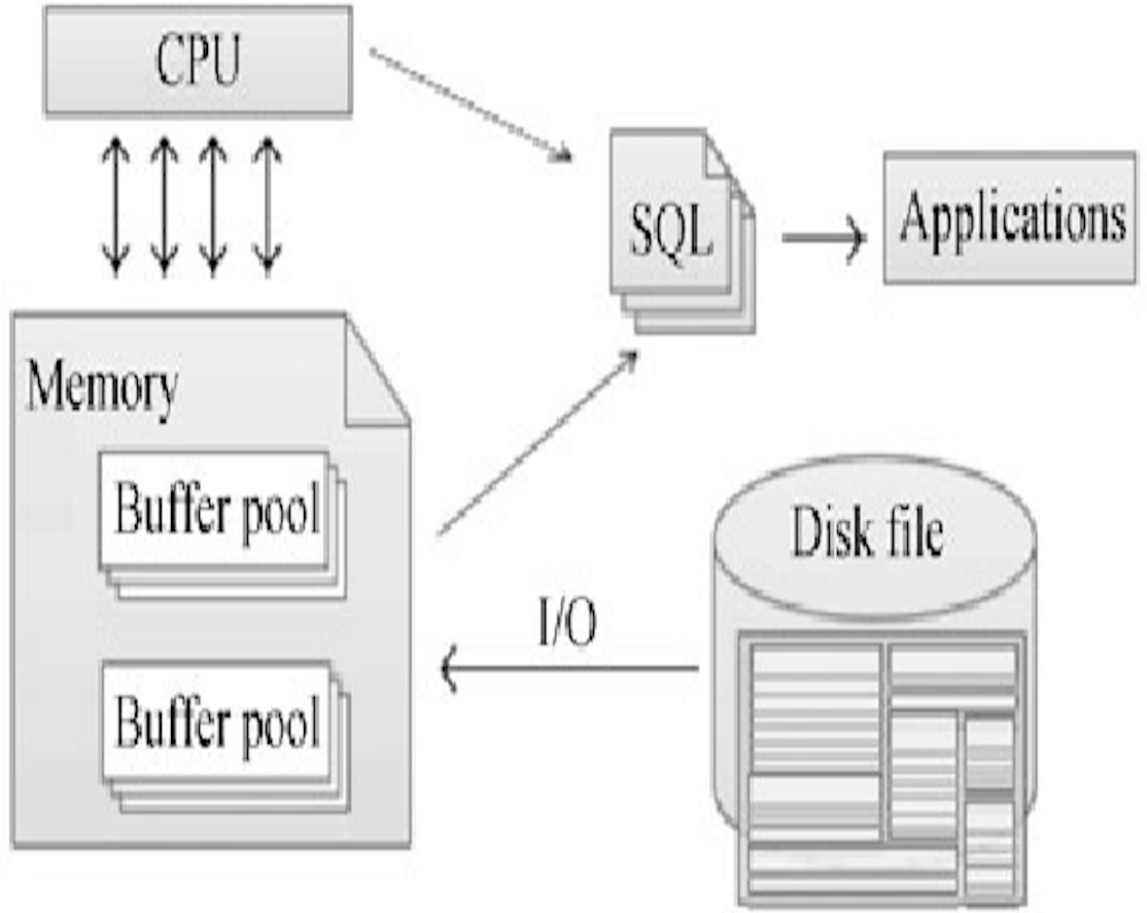
633

Page (sayfa) sayısı

Page Input – Output. (page I/O)

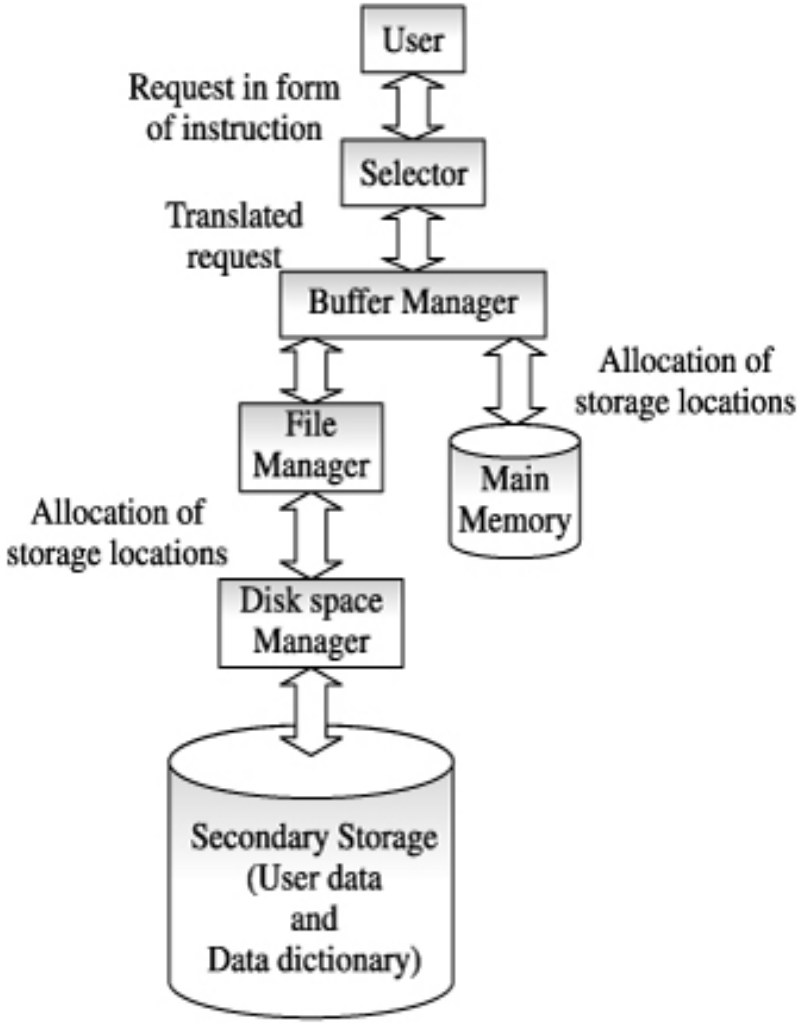
(Sayfa Giriş / Çıkış)

- Veritabaninda bulunan kayıtlara erişimi sağlamaktadır (diskten hafizaya - hafizadan diske)
- Gerçekleştirilecek sorgulama işlemlerinin maliyetini birinci dereceden etkilemektedir.
- Page I /O nun işlem maliyetlerini düşürmek için optimizasyon çalışmaları yapılmaktadır.



Ara Bellek Yöneticisi (Buffer Manager)

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



Dosya Organizasyonları ve İndexleme

- 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 İndeksleme

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

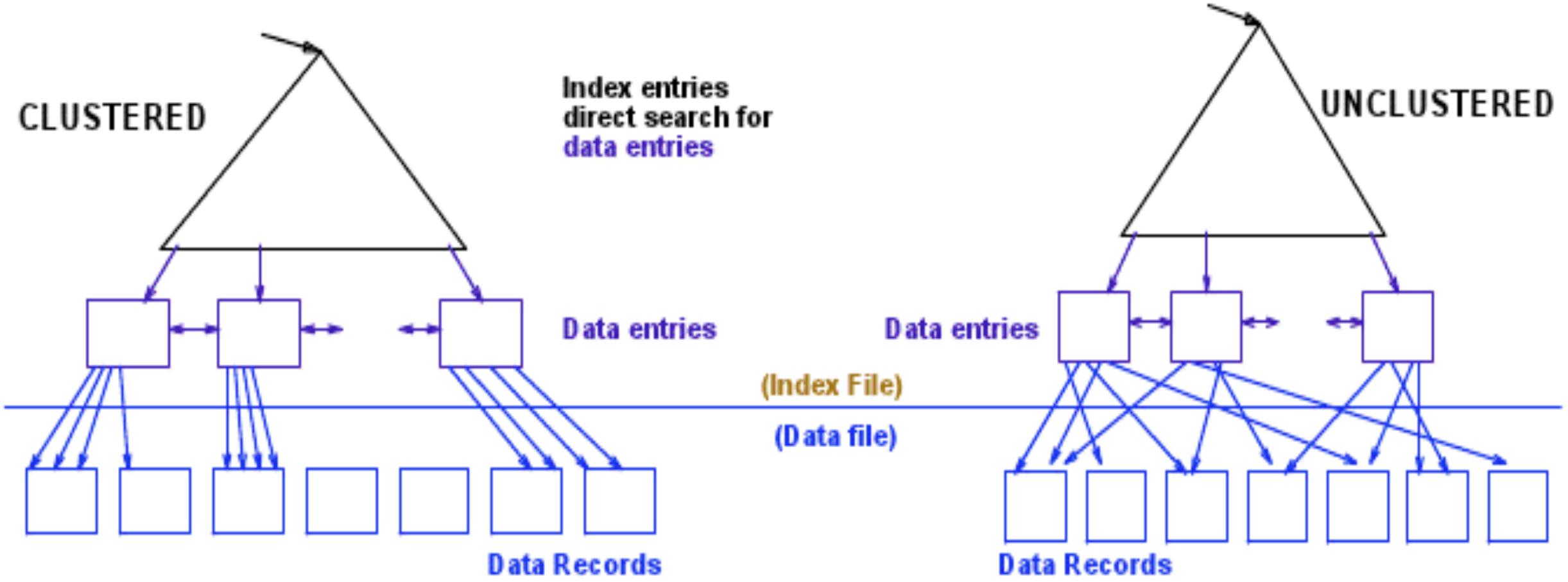
1. **key* (arama anahtari ve veri)**
2. (key , rid)
3. (key , rid-list)

Dosya Organizasyonları ve İndeksleme

- Index yapısı **clustered(kümelenmiş) index** olduğu zaman istenilen cevap birbiri ardınca sıralanan sayfa(lar) icerisindedir.

- **Unclustered** olunca bütün kayıtların incelenmesi gerekmektedir.

Dosya Organizasyonları ve İndeksleme



https://www.quora.com/How-are-clustered-and-non-clustered-indexing-implemented-internally

Dosya Organizasyonları ve İndeksleme

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

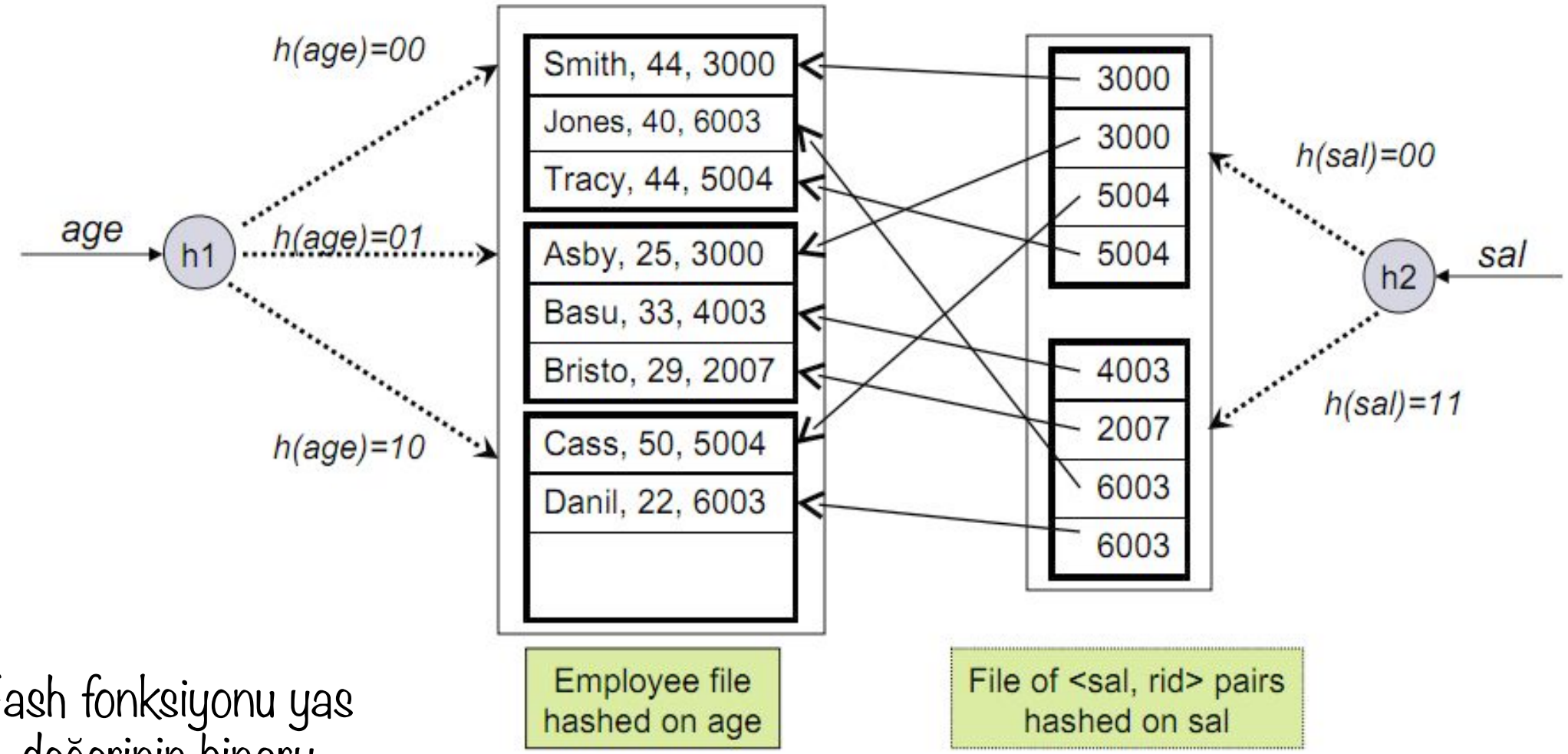
İndeks Yapısı

1. Hash data entries
2. Tree structure

İndeks Yapısı (Hash Tabanlı İndexleme)

- Kayıtlar anahtar değer (key-value) göre gruplandırılır
- Her bir alt grup **bucket** (kova) olarak isimlendirilir.
- Hash fonksiyonunun sonucuna göre verinin gruplara ayırma işlemi gerçekleşir.
- Yeni değerler eklenirken hash fonksiyonunun sonucuna göre bucket belirlenir.

İndeks Yapısı (Hash Tabanlı İndexleme)



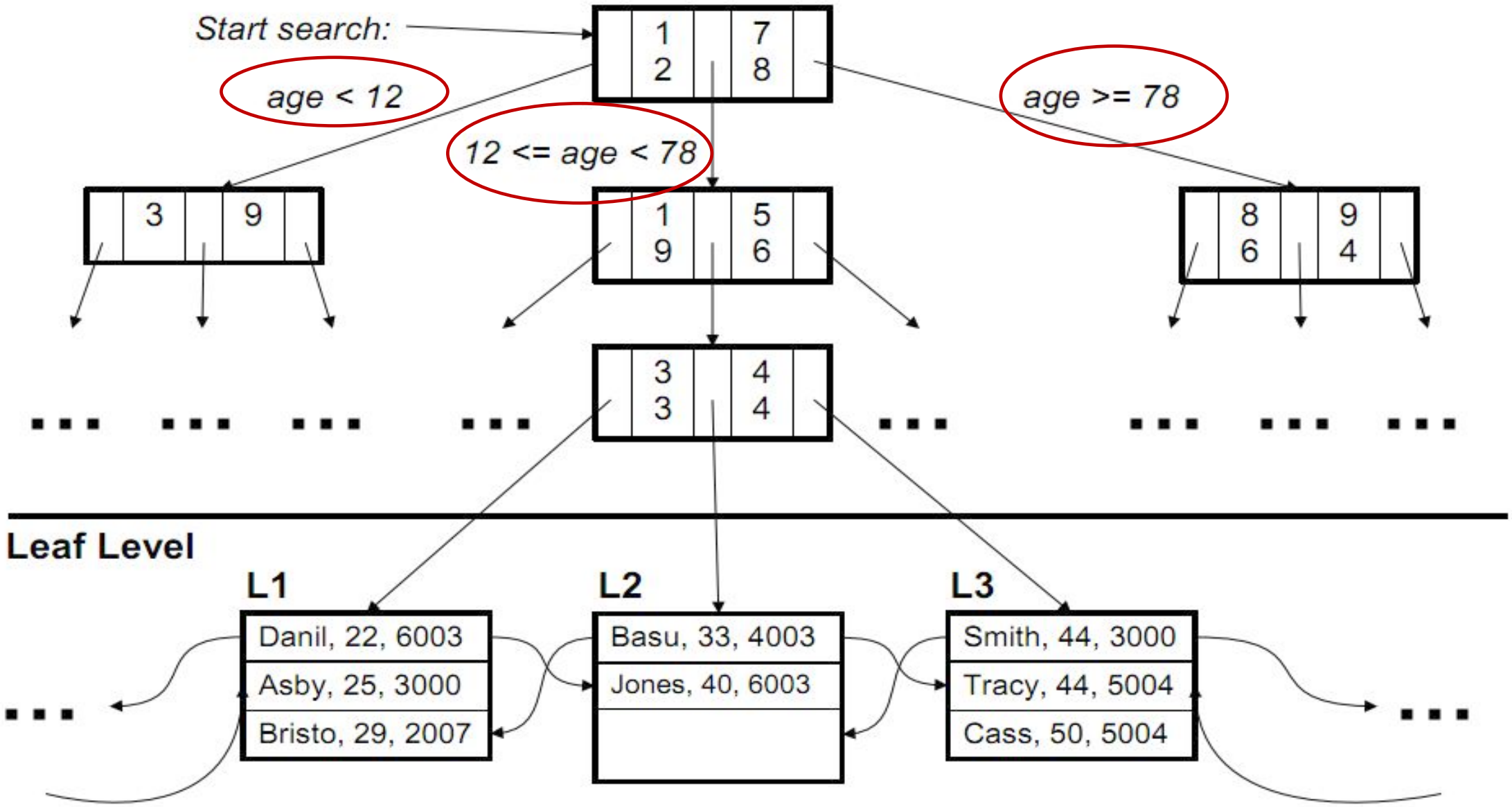
Hash fonksiyonu yas değerinin binary son iki bitine göre işlem gerçekleştiriyor

http://bosbluebluesky.blogspot.com.tr/2012/01/database-concept.html

İndeks Yapısı (Tree Based İndexleme)

- Tree based (ağaç tabanlı) indexleme hash yapısının alternatifidir.
- Belirlenen alana göre sıralama ve gruplandırma ile ağaç yapısı oluşturulur.
- Bu yapı ile beraber aramalar **belirlenen aralıkta** gerçekleştirilir.

İndeks Yapısı (Tree Based İndexleme)



PostgreSQL İndeks 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);

http://patshaughnessy.net/2014/11/11/discovering-the-computer-science-behind-postgres-indexes

İndeks Yapılarının Karşılaştırılması

Sıralı olmayan dosya (file unsorted)

- Hızlı tarama
- Hızlı kayıt ekleme
- Arama ve silme işlemlerinde yavaş

İndeks Yapılarının Karşılaştırılması

Sıralı dosya (sorted file)

- kayıt ekleme ve kayıt silme yavaş
- Arama(search) sıralı olmayan file dan hızlı

İndeks Yapılarının Karşılaştırılması

Clustered file

- kayıt ekleme ve kayıt silmede etkili
- Aramalar sıralı olan dosyalardan hızlı

İndeks Yapılarının Karşılaştırılması

Hash ve Tree yapılarında

- kayıt ekleme ve kayıt silme etkili
- Aramalar hızlı
- Tarama ve aralık bulmada yavaş

İndeks Yapılarının Karşılaştırılması

File Type	Scan	Equality Search	Range Search	Insert	Delete
Heap	BD	$0.5BD$	BD	$2D$	$Search + D$
Sorted	BD	$Dlog_2B$	$Dlog_2B + \#$ matching pages	$Search + BD$	$Search + BD$
Clustered	$1.5BD$	$DlogF1.5B$	$Dlo9F1.5B + \#$ matching pages	$Search + D$	$Search + D$
Unclustered tree index	$BD(R + 0.15)$	$D(1 + logFO.15B)$	$D(lo9FO.15B + \#$ matching records$)$	$D(3 + logFO.15B)$	$Search + 2D$
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 fonksıyonu,

PostgreSQL: Cascade

```
CREATE TABLE products (
    product_no integer PRIMARY KEY,
    name text,
    price numeric );
```

```
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    shipping_address text,
    ... );
```

the child data is either deleted or updated when the parent data is deleted or updated

PostgreSQL: Cascade

```
CREATE TABLE products (
    product_no integer PRIMARY KEY,
    name text,
    price numeric );
```

```
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    shipping_address text,
    ... );
```

the child data is either deleted or updated when the parent data is deleted or updated

```
CREATE TABLE order_items (
    product_no integer REFERENCES products(product_no) ON DELETE RESTRICT,
    order_id integer REFERENCES orders(order_id ) ON DELETE CASCADE,
    quantity integer,
    PRIMARY KEY (product_no, order_id)
);
```

PostgreSQL: Cascade

```
CREATE TABLE products (
    product_no integer PRIMARY KEY,
    name text,
    price numeric );
```

```
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    shipping_address text,
    ... );
```

the child data is either deleted or updated when the parent data is deleted or updated

```
CREATE TABLE order_items (
    product_no integer REFERENCES products(product_no) ON DELETE RESTRICT,
    order_id integer REFERENCES orders(order_id ) ON DELETE CASCADE,
    quantity integer,
    PRIMARY KEY (product_no, order_id)
);
```

Restrict default olarak primary key alanına bagli foreign key'ler mevcut oldugunda slime işlemini engeller.

Cascade products tablosundan bir ürün silindiğinde ilişkili tablolardan da atomatik olarak silinmesini sağlar

NO ACTION
ON UPDATE CASCADE
ON UPDATE RESTRICT

PostgreSQL: Trigger

- Bir veritabanında tanımlanan özel bir olay gerçekleştiğinde otomatik olarak veritabanında bulunan bir işlemin geröekleşmesini sağlayan prosedurdur.
- Belirli bir tabloya bir satır eklenmesi
- Bir tabloda bulunan sütunları güncellenmesi
- Bir trigger'ın (tetikleyici) çalışmasını sağlar.

```
CREATE TRIGGER trigger_name
    {BEFORE | AFTER | INSTEAD OF} {event [OR ...]}
     ON table_name
     [FOR [EACH] {ROW | STATEMENT}]
     EXECUTE PROCEDURE trigger_function
```

Dinlediğiniz İçin Teşekkürler....