Veritabanı Yönetim Sistemleri (335)

Dr. Öğr. Üyesi Ahmet Arif AYDIN

L17-

Dosya Yapıları ve İndexleme (storage & indexing)

GÜZ -2022

Ders Takvimi

Lecture Notes (2021)

L16- PostgreSQL Veri Tipleri (pdf & mp4)
L15- SQL-4 (pdf & mp4)
L14- SQL-3 (pdf & mp4)
L13- SQL-2 (pdf & mp4)
L12- SQL-1 (pdf & mp4)
L11- İlişkisel Hesap (pdf & mp4)
L10- İlişkisel Cebir (pdf & mp4)
L9- İlişkisel Veri Modeli (pdf & mp4)
L8- UML (pdf & mp4)
L7- ER Modeli-2 (pdf & mp4)
L6- ER Modeli-1 (pdf & mp4)
L5-Veritabanı Kullanıcıları ve Tasarım (pdf & mp4)
L4- Dosya Sistemleri ve Problemler (pdf & mp4)
L3-Veritabani Çeşitleri (pdf & mp4)
L2-Introduction to DBMS (pdf & mp4)
L1- General Overview (pdf)

http://aaaydin.github.io/teaching/

Konular
Veritabanı Tasarımı
ER-Model
UML
Relational Model
İlişkisel Cebir
İlişkisel Hesap
SQL (1-2-3-4)
Dosya Yapıları ve İndexleme
Sorgu Optimizasyonu
Hareket Yönetimi
Kilitlenmeler
Concurrency
Veritabanı Kurtarma Teknikleri

Veritabanı Yönetim Sistemleri ve Sağladıkları

Veritabanı yönetimi sistemleri verinin

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

sağlamaktadır.

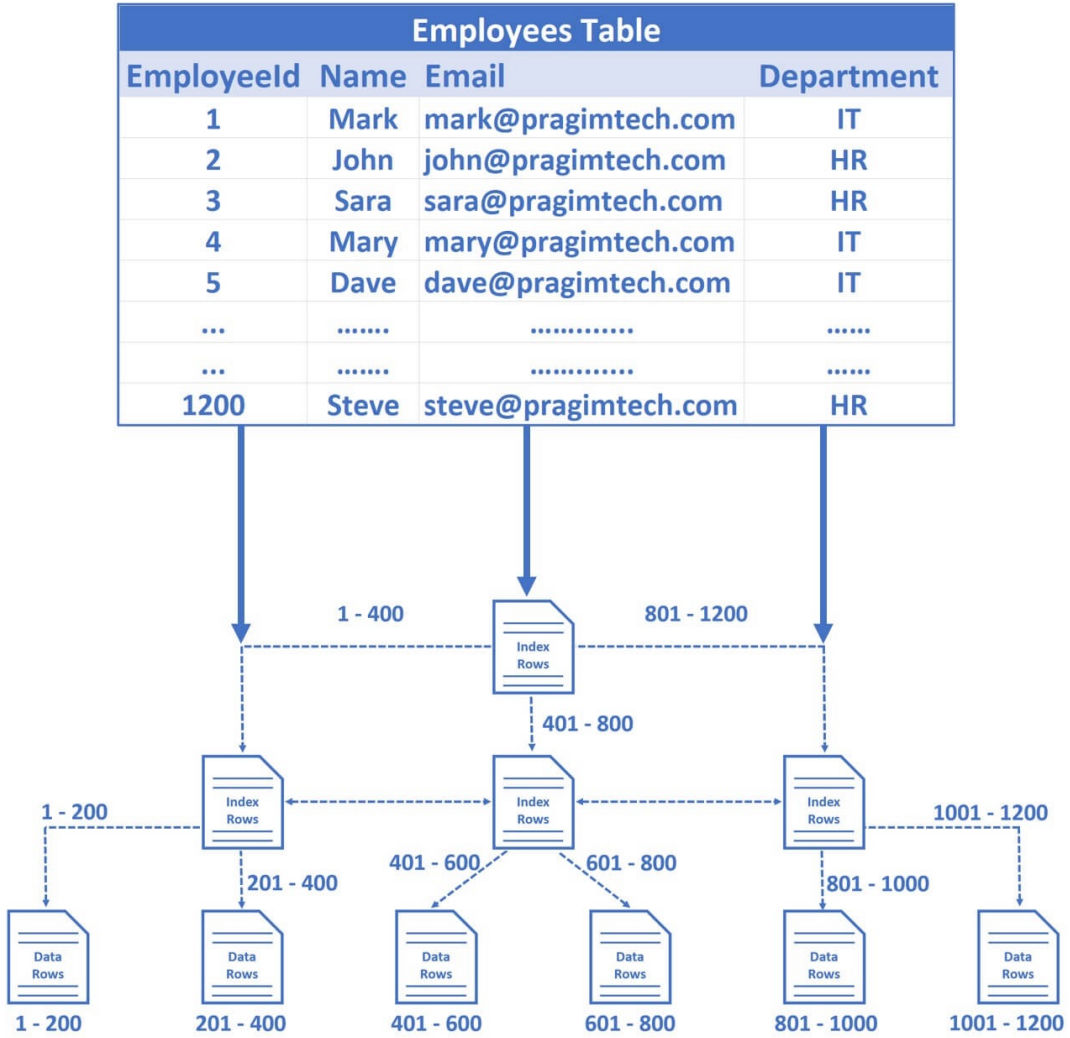
Sorular

Veritabanı yönetim sistemleri

- Tablolara eklenen(*insert*) veriyi disk üzerinde nasıl kaydetmektedir?
- Disk üzerinde kayıtlı veri nasıl yönetilmektedir (management)?
- Veriye erişimi (*access*) nasıl sağlamaktadır?

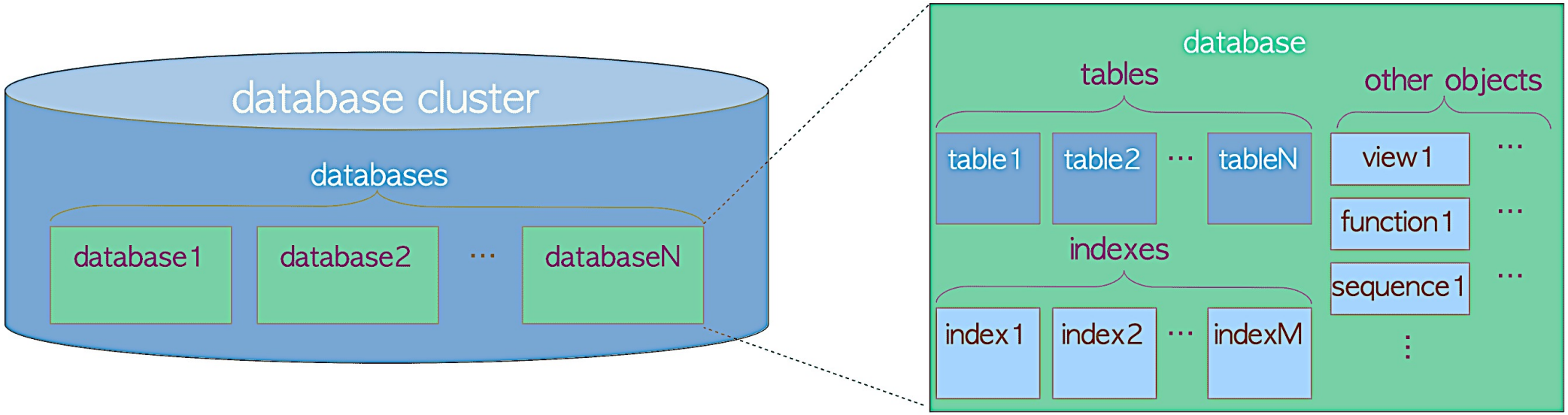
Verinin Diske Kaydedilmesi

- Veritabanı yönetim sistemleri farklı formatlardaki çok çeşitli veriyi hard disk üzerine kaydetmektedir.
- Veritabanı parametresi **sayfa (page)** dır.
- 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.



https://www.pragimtech.com/blog/sql-optimization/how-is-data-stored-in-sql-database/

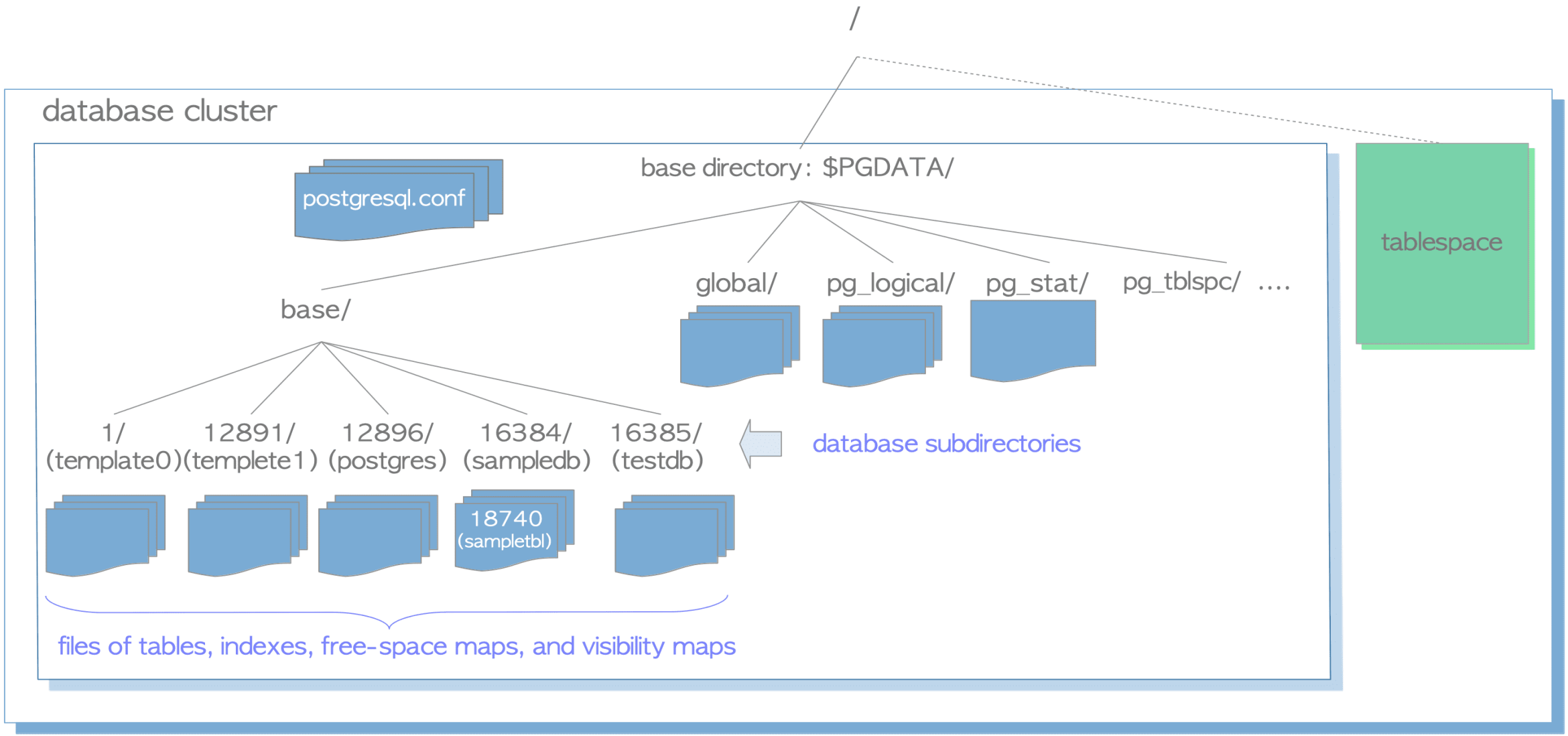
Verinin Diske Kaydedilmesi



- PostgreSQL Server tarafından *database cluster* yönetilir.
- Database *cluster tek bir server üzerine kuruludur*
- Veritabanında bulunana bütün nesneler *object identifiers* (OIDs- 4byte integer) ile belirlenir.

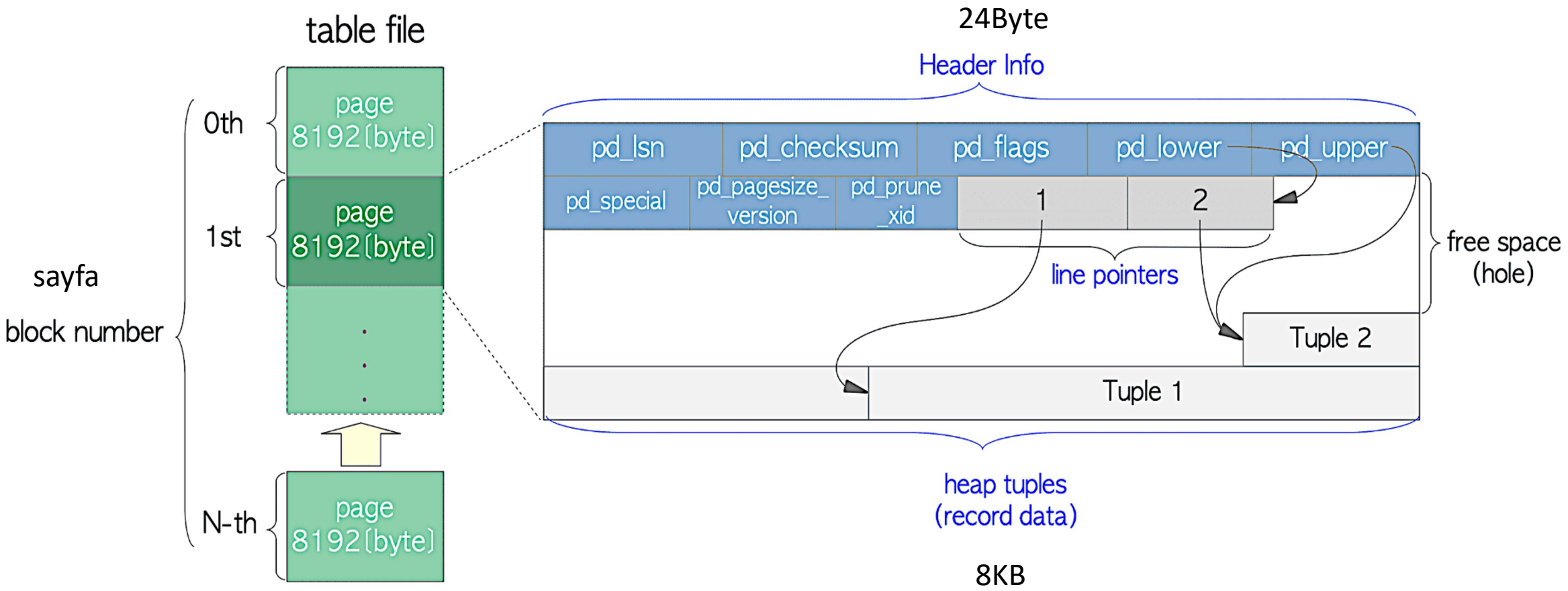
https://www.interdb.jp/pg/pgsql01.html

Verinin Diske Kaydedilmesi



https://www.interdb.jp/pg/pgsql01.html

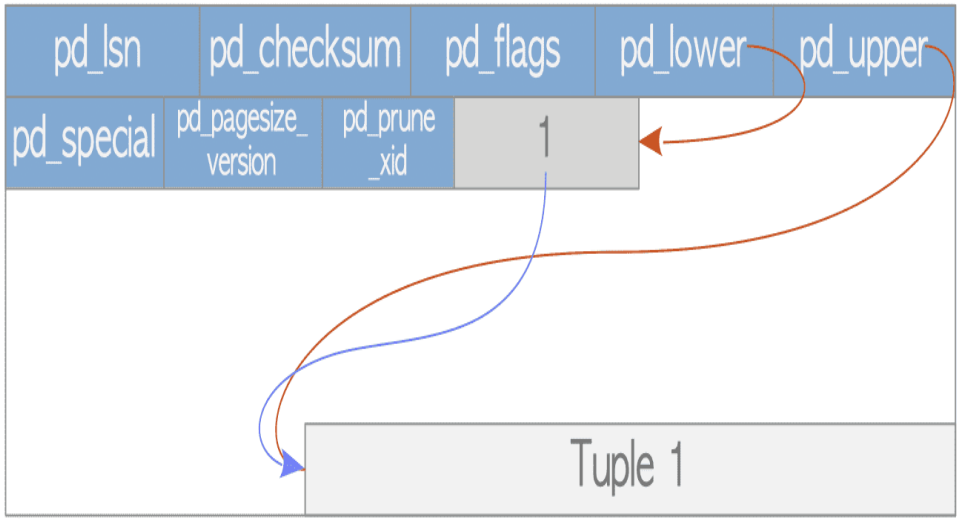
Verinin Diske Kaydedilmesi



https://www.interdb.jp/pg/pgsql01.html

Verinin Diske Kaydedilmesi

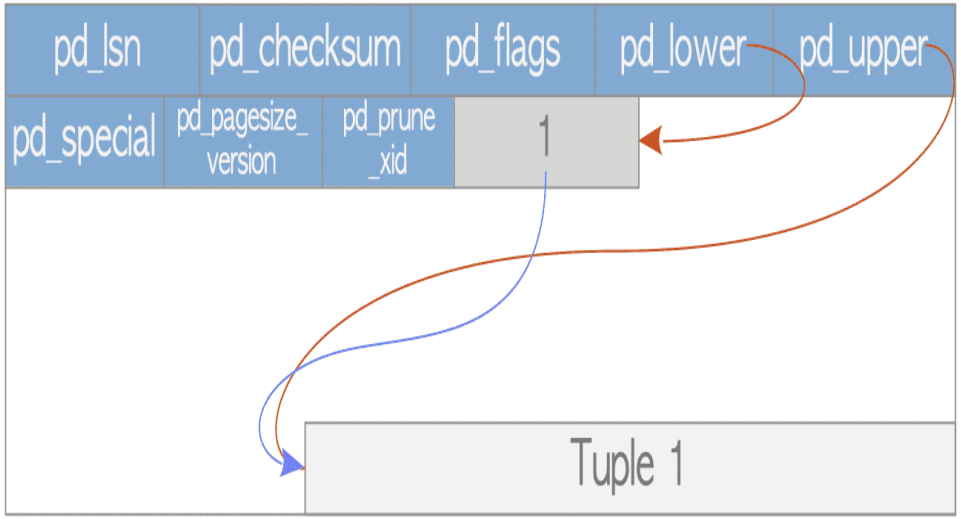
(a) Before insertion of Tuple 2



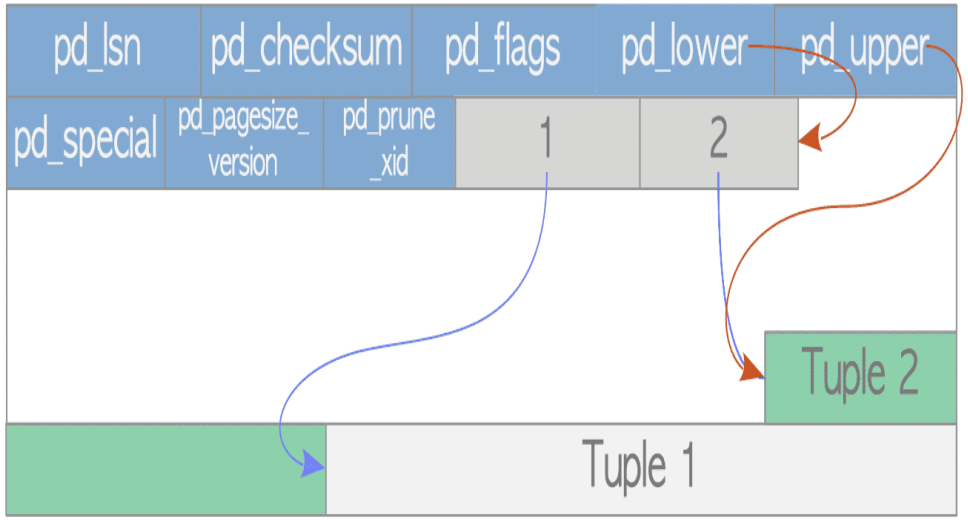
https://www.interdb.jp/pg/pgsql01.html

Verinin Diske Kaydedilmesi

(a) Before insertion of Tuple 2



(b) After insertion of Tuple 2

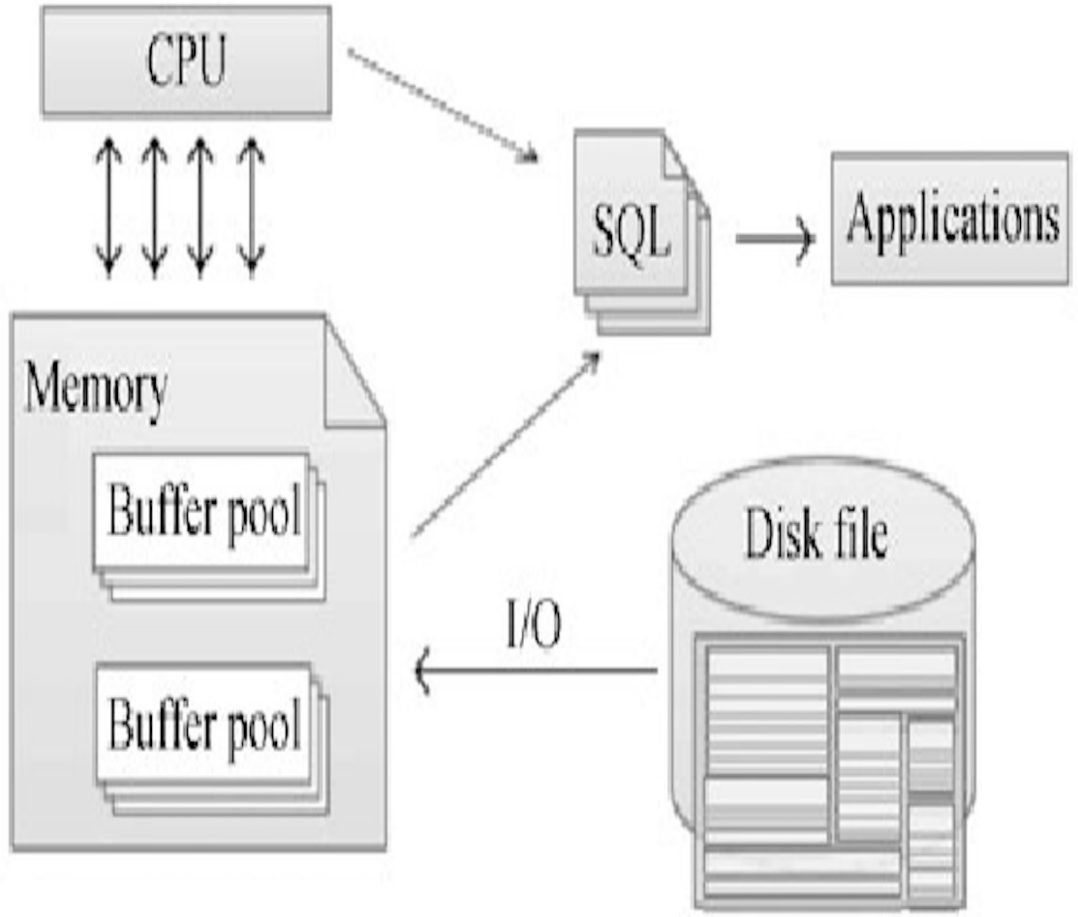


https://www.interdb.jp/pg/pgsql01.html

Page Input – Output (page I/O)

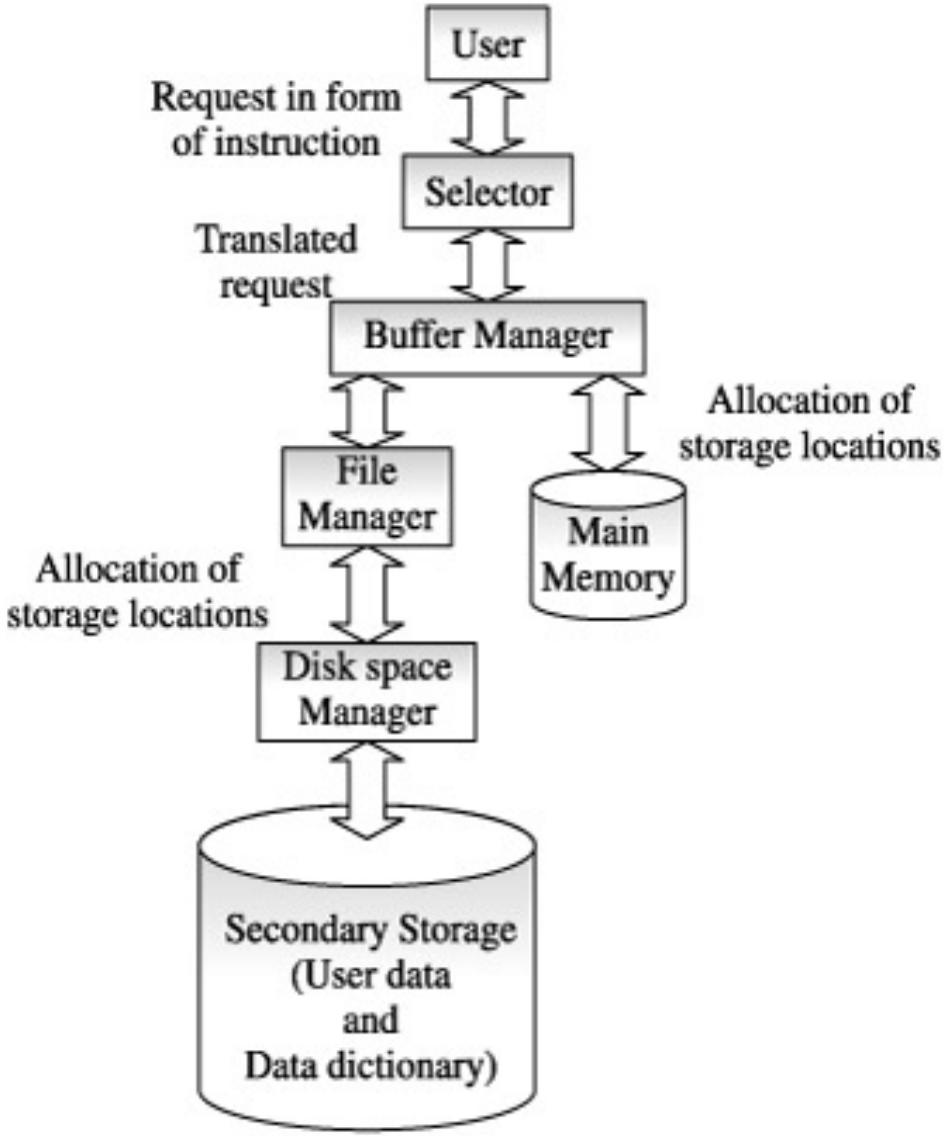
(Sayfa Giriş / Çıkış)

- Veritabanında bulunan kayıtlara erişimi sağlamaktadır (diskten hafızaya - hafızadan 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.

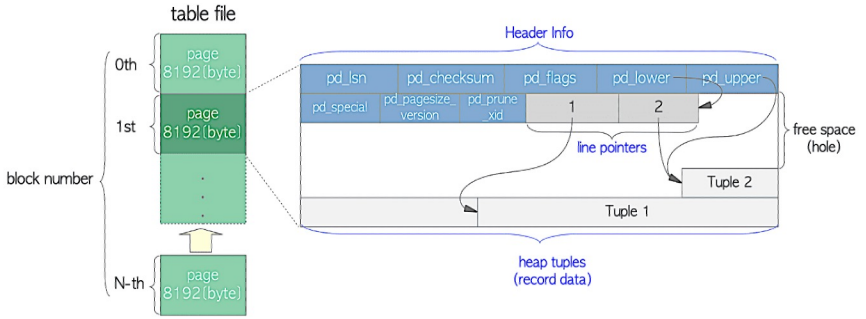


Verinin Diske Kaydedilmesi

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
24 bytes: bir integer ve text
4 bytes: pointer on page to tuple

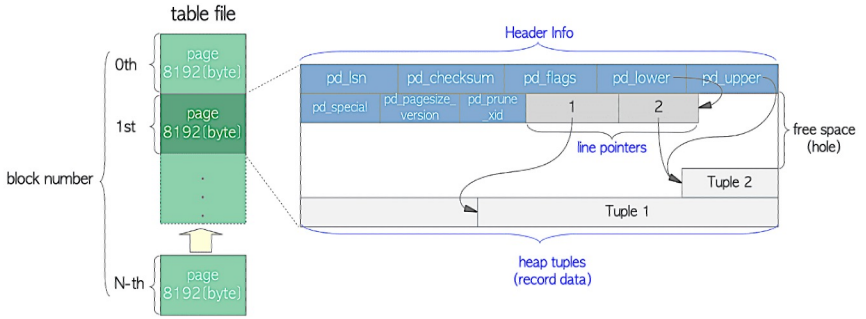
---

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
24 bytes: bir integer ve text
4 bytes: pointer on page to tuple

---

52 bytes (1satır)

Bir satırın kapladığı alan

8192 bytes
/
52 bytes

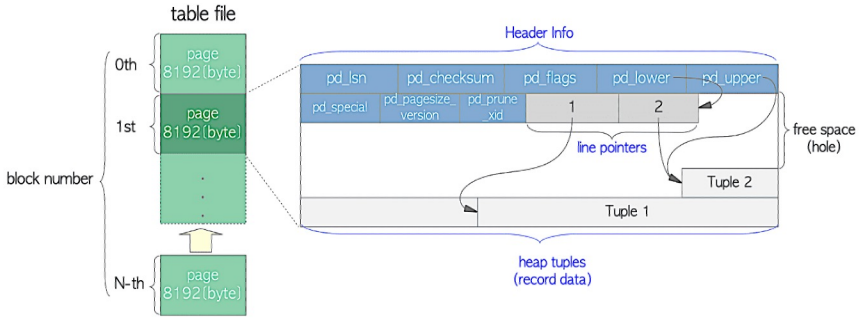
---

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
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

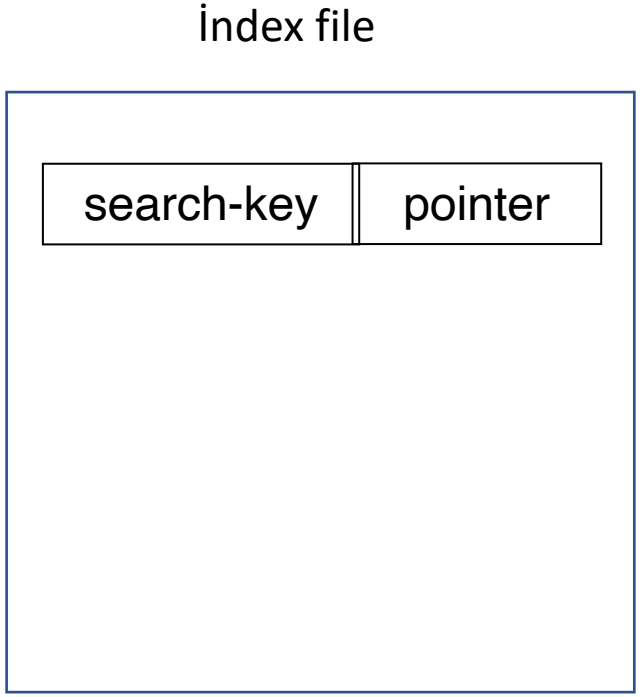
100.000
/
158

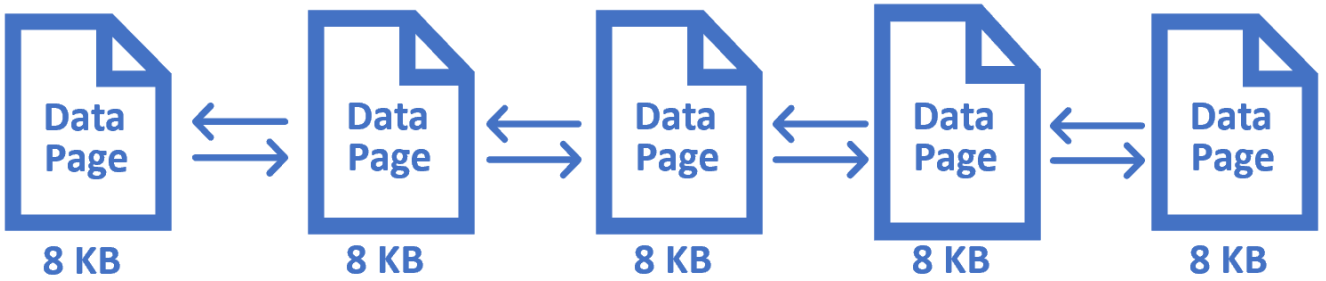
633

Page (sayfa) sayısı

İndeks nedir?

- Kayıtlı olan verilere erişmek için **index** kullanılır.
- İndex bir veri yapısı olarak düşünülebilir.
- Page(sayfa) içerisinde kayıtlı olan veriye efektif ve hızlı bir biçimde erişim imkanı index yardımıyla sağlar.
- Disk üzerinde kayıtlı olan dosyaları üzerindeki gerçekleştirilecek CRUD işlemlerini gerçekleştirmek için index kullanılır





İndeks çeşitleri

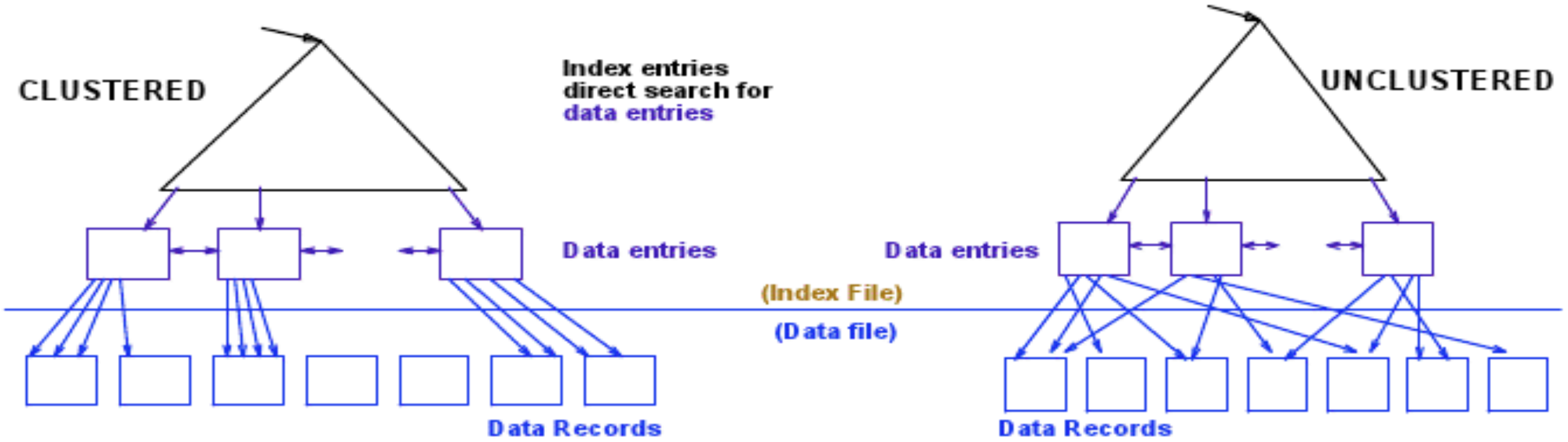
İki ana başlık altında değerlendirilir:

- Sıralı (ordered) indexler
- Hash indexler

İndeks çeşitleri: Sıralı index

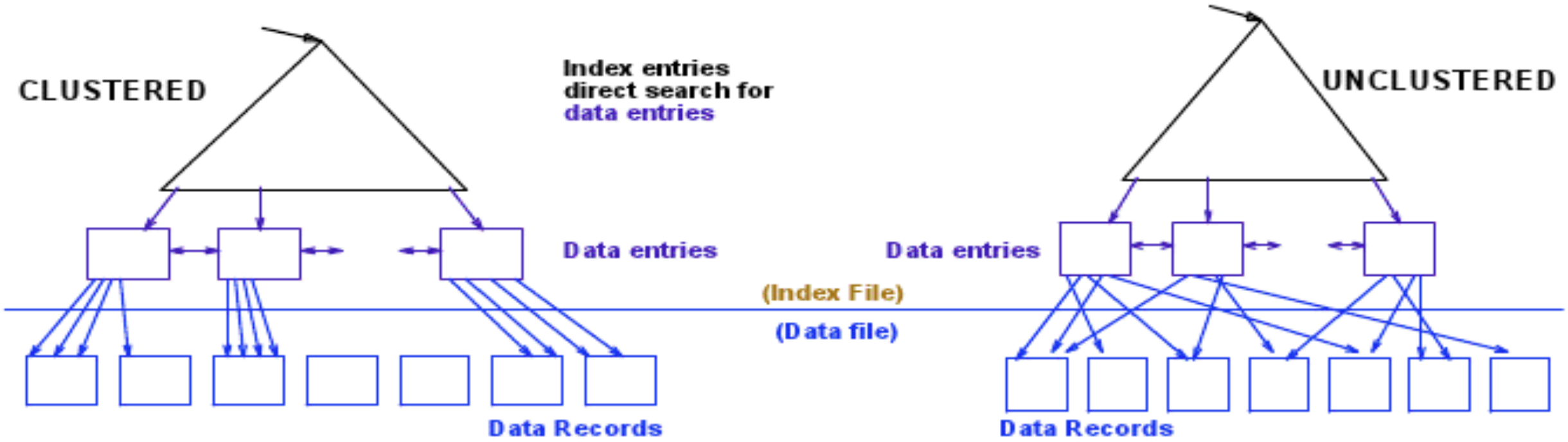
- Primary Index
  - Indexler sıralı bir biçimde kaydedilir.
  - Index genellikle primary key dir
  - Clustering (clustered) index olarak da adlandırılır.
- Secondary index
  - Sıralıdır fakat sıralamada primary key kullanılmamış
  - Unclustered (unclustering) veya secondary index olarak adlandırılır.

İndeks çeşitleri: Sıralı index



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

İndeks çeşitleri: Sıralı index



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

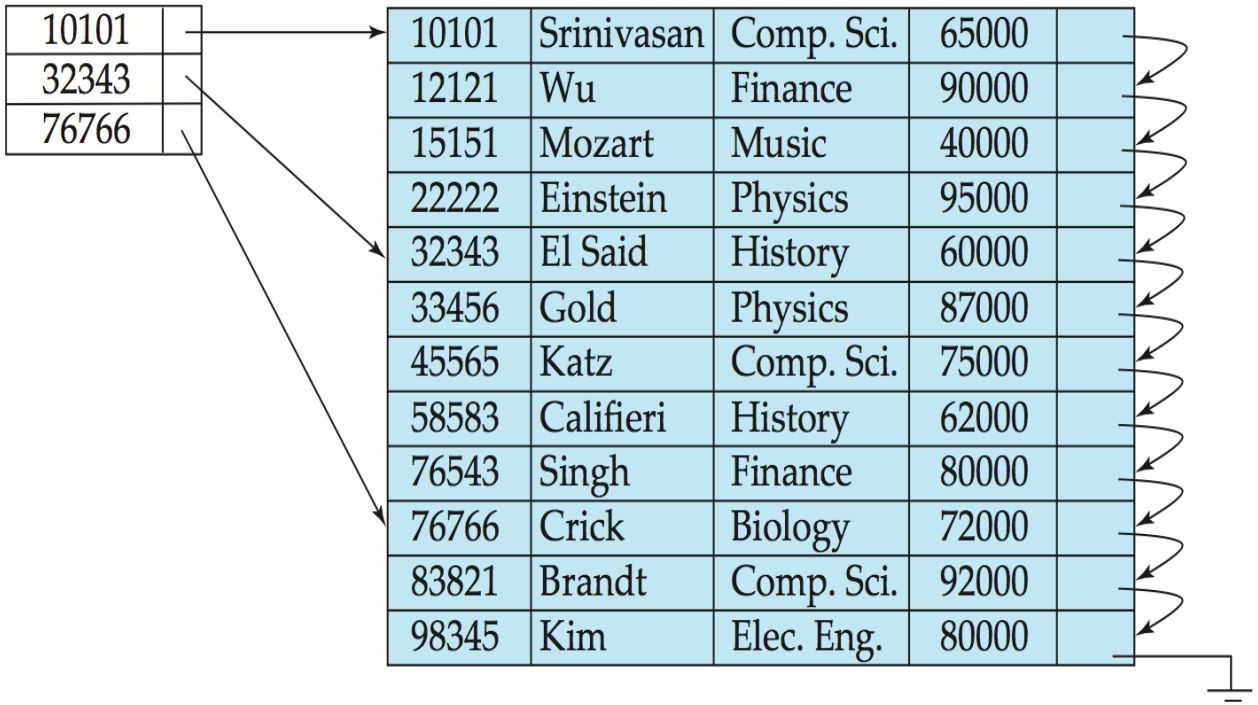
Unclustered olunca bir değerin aranması için bütün kayıtların incelenmesi gerekmektedir.

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

Sıralı İndeks çeşitleri: Sparse index – Dense Index

Sparse Index

>

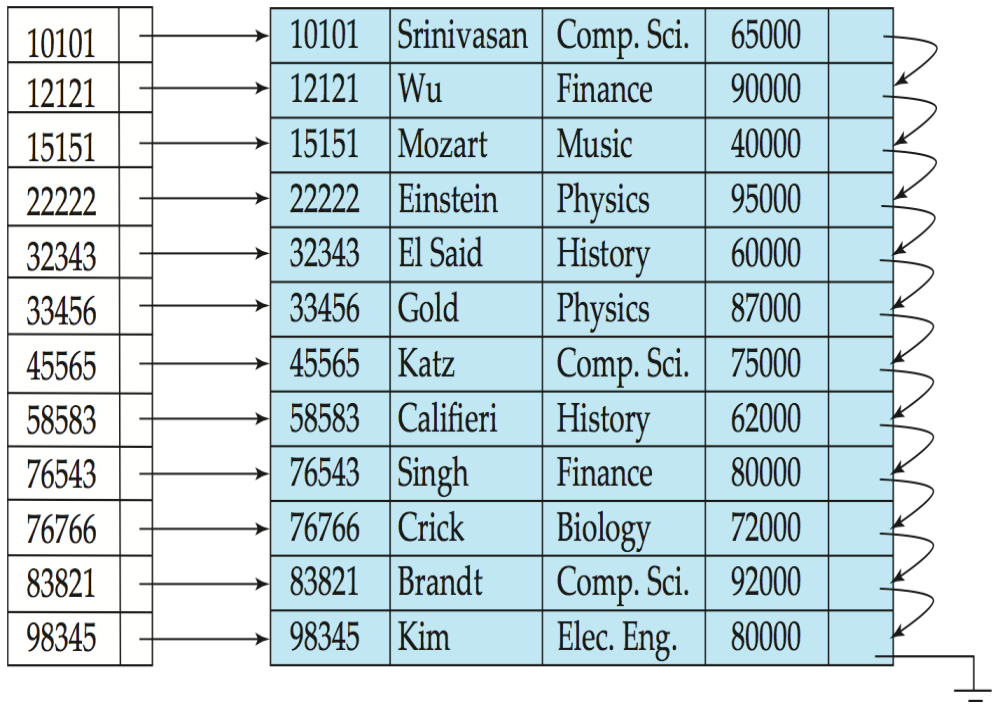


10101	
32343	
76766	

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

Dense Index

=

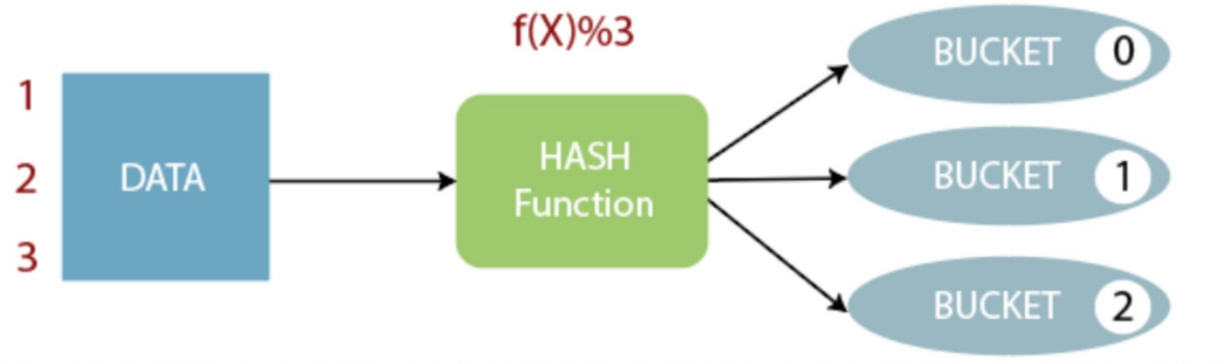


10101	
12121	
15151	
22222	
32343	
33456	
45565	
58583	
76543	
76766	
83821	
98345	

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

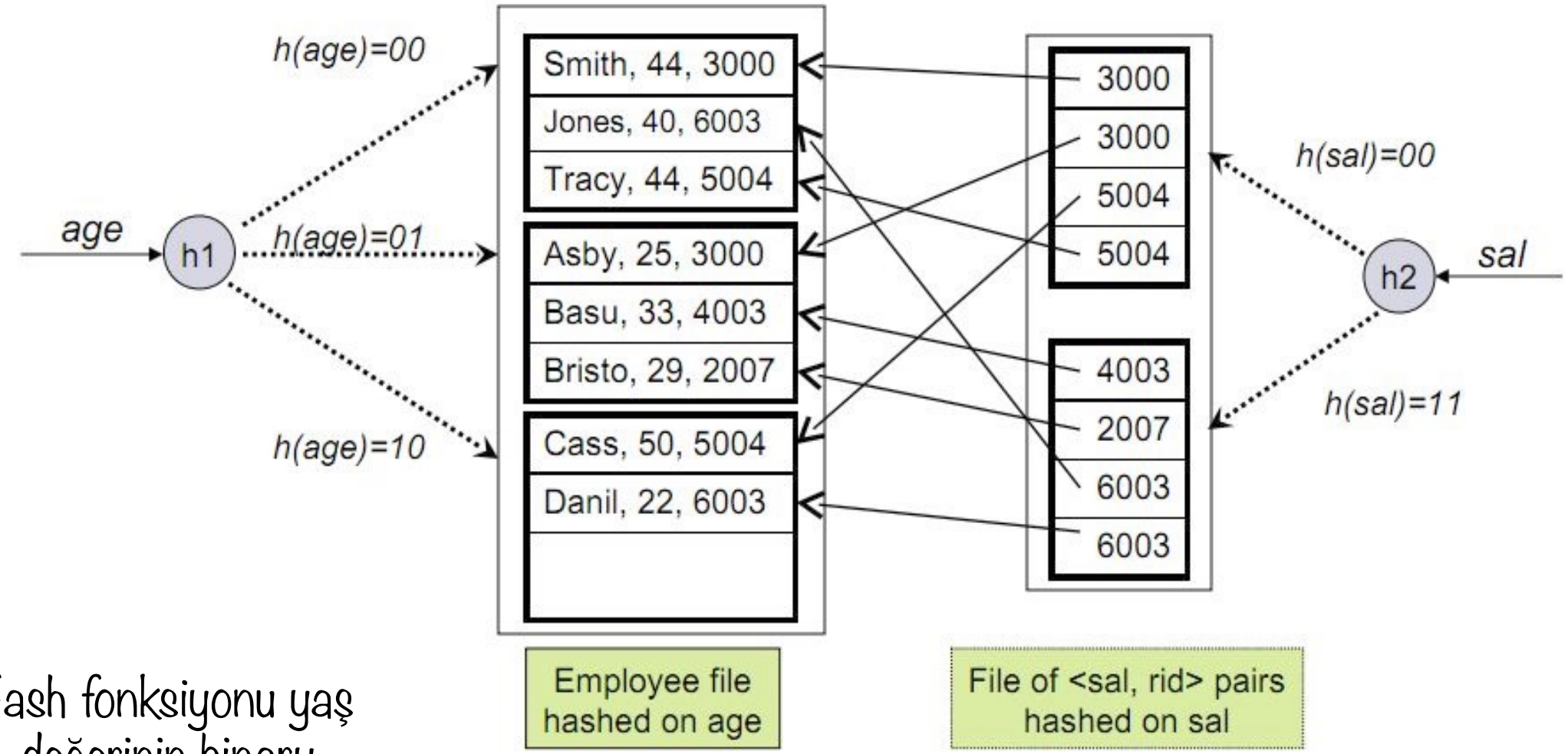
İndeks çeşitleri: Hash index

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



https://cwiki.apache.org/confluence/display/HUDI/RFC+-+29%3A+Hash+Index

İndeks çeşitleri: Hash index



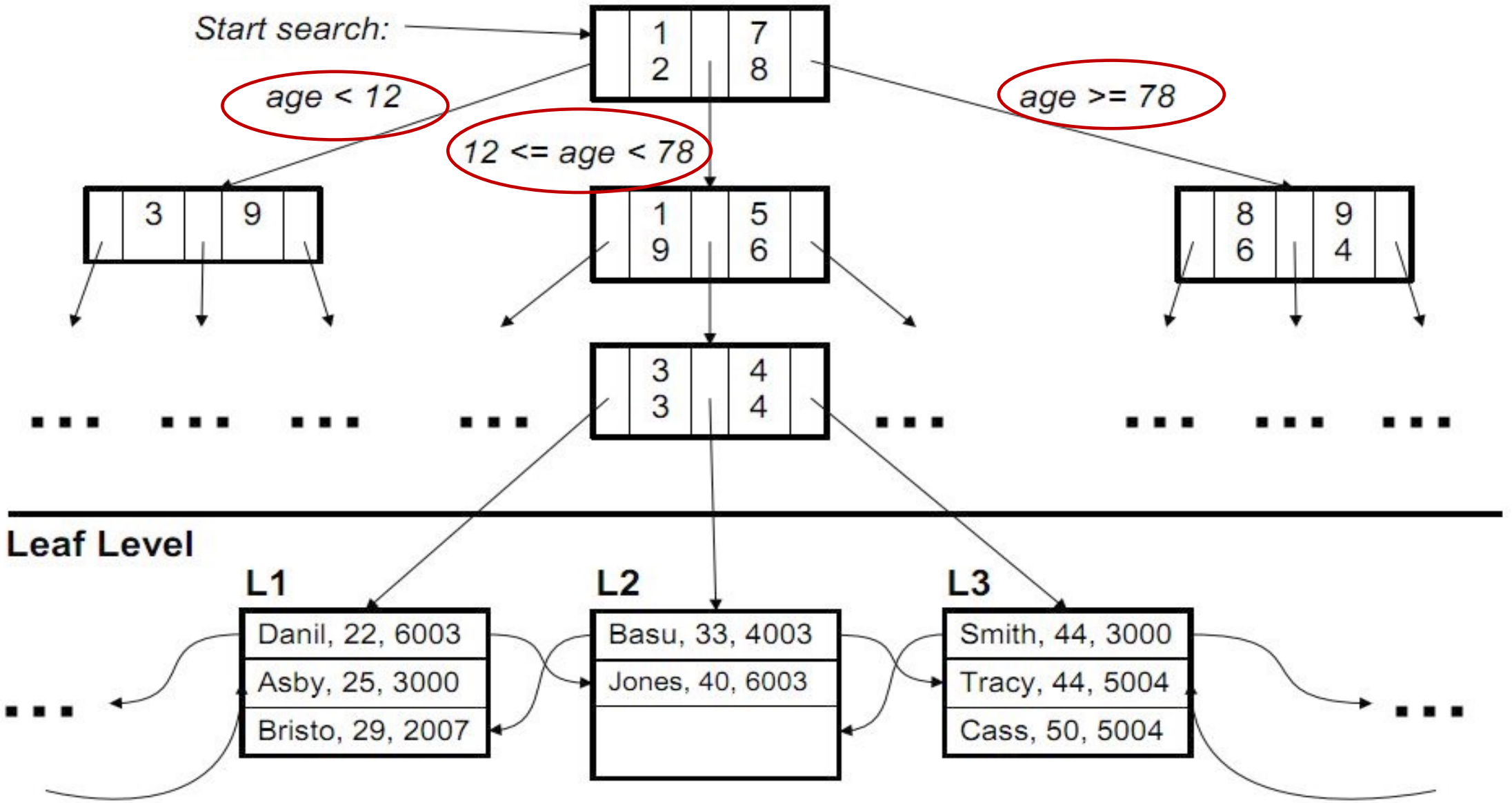
Hash fonksiyonu yaş değerinin binary son iki bitine göre işlem gerçekleştiriyor

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

Ağaç Yapısı tabanlı (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.

Ağaç Yapısı tabanlı (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 Karşılaştırılma Kriterleri

- Desteklenen bağlantı tipi
  - anahtar değer (key-value)
  - aralık (range)
- Bağlantı zamanı (access time)
- Veri ekleme süresi (insertion time)
- Veri silme süresi (deletion time)
- Space overhead

İndeks Yapılarının Karşılaştırılması

Sıralı olmayan dosya (file unsorted)

- Hızlı kayıt ekleme
- Arama ve silme işlemlerinde yavaş
- Hızlı tarama (scan)

İndeks Yapılarının Karşılaştırılması

Sıralı dosya (sorted file)

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

İndeks Yapılarının Karşılaştırılması

Clustered file

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

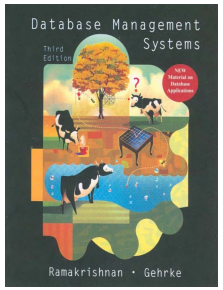
İndeks Yapılarının Karşılaştırılması

Hash ve Tree yapılarında

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

İndeks Yapılarının Karşılaştırılması

	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) Delete
(1) Heap	BD	0.5BD	BD	2D	Search +D
(2) Sorted	BD	Dlog 2B	D(log 2 B + # pgs with match recs)	Search + BD	Search +BD
(3) Clustered	1.5BD	Dlog F 1.5B	D(log F 1.5B + # pgs w. match recs)	Search + D	Search +D
(4) Unclust. Tree index	BD(R+0.15)	D(1 + log F 0.15B)	D(log F 0.15B + # pgs w. match recs)	Search + 2D	Search + 2D
(5) Unclust. Hash index	BD(R+0.125)	2D	BD	Search + 2D	Search + 2D



B: sayfa sayısı , D: okuma yazma zamanı R: Safyada bulunan kayıt sayısı

Dinlediğiniz için

Teşekkürler...

İyi çalışmalar...