week 10

Data Storage and Indexes

Spatial control → performance (store pages sequentially)
Temporal " → Locking (buffer manager in DBMS)

Spatial Control

Raw → skips OS file management
- → organizes disk space as it wishes ⊕
- → performance ↑ ⊕
- → OS cannot deal with that space ⊖
- → not portable ⊖

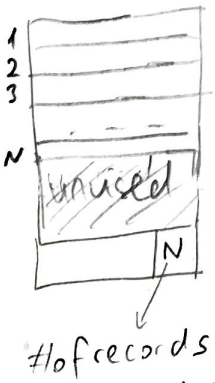
Using OS → DBMS creates 1 huge file.
- → good for physical locality ⊕
- → OS can limit size ⊖
- → # of open fd ⊖

Commercial systems uses both

*when a page is requested, we can predict next pages and prefetch
then (less disk access ⊕)

Records

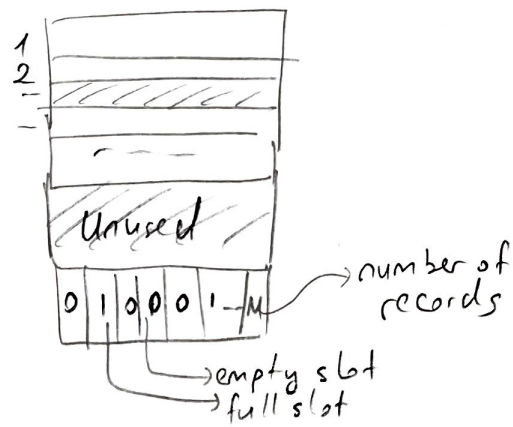
Fixed length



Packed version

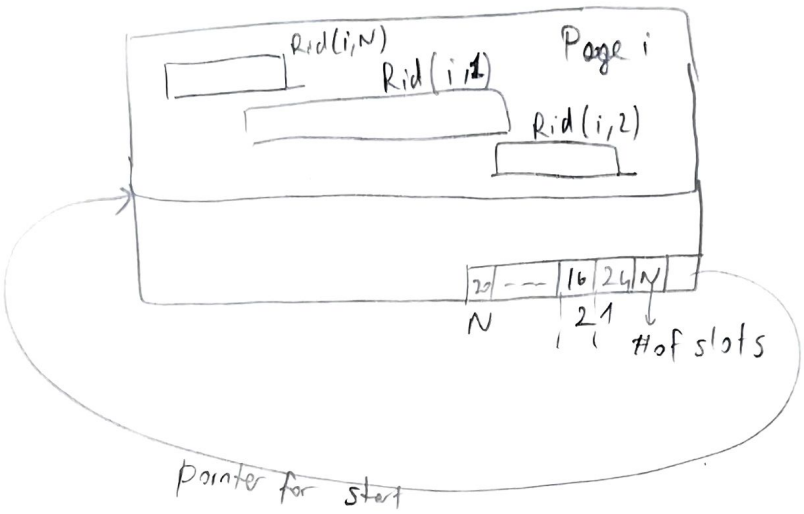
if you delete, you shift.

Unpacked



Recordid <pageid, slot#>

Variable Length



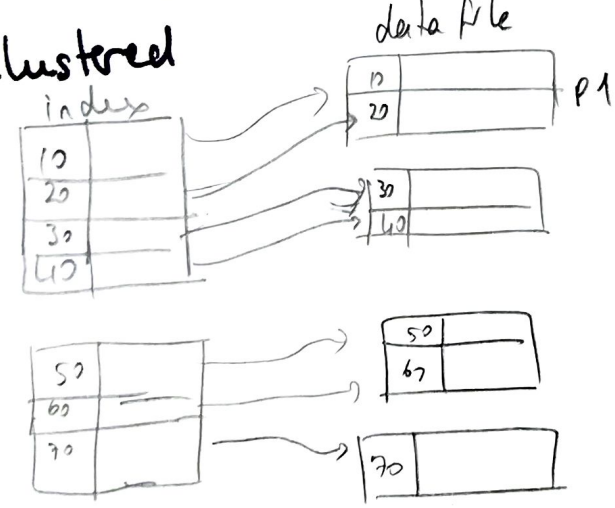
Can move records on page w.o. changing (shifting).

DB file types → Heap → unsorted
→ Sequential → Sort according to a key (not necessarily P.K.)

Index

- Key value pairs
- Commonly a seperate file

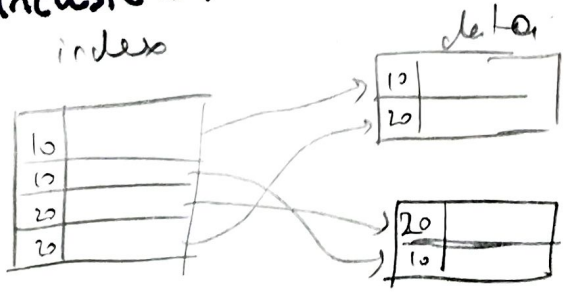
Clustered



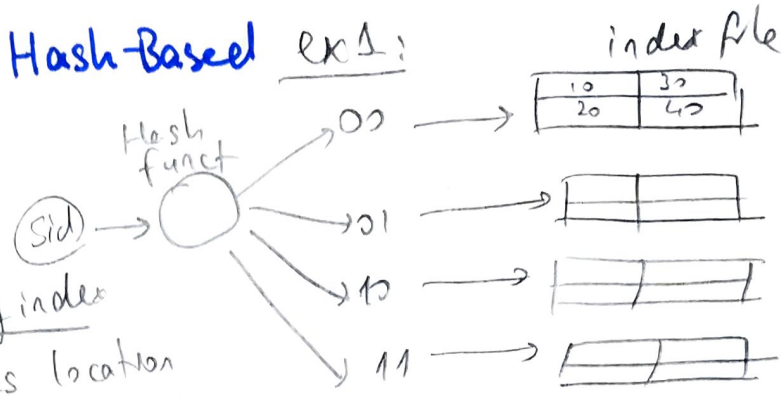
Adv: 10'u okuma istegince P1' memory'e geldi. 20 de gelmis oldu ✓.

Records close in index are close in data.

Unclustered



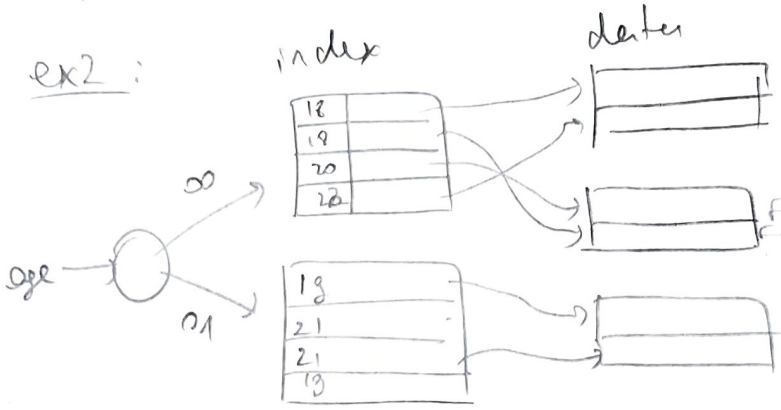
Hash-Based ex1:



No seperate data file (hash on sid)

clustered

ex2:



Secondary index

Data entries (key, record id)

which record

Unclustered

Clustered → hash, find, fetch → 1 access

Unclustered → hash, fetch index, find data (record id), fetch data

↳ 2 access

Searching B+ tree

index on age

age = 25 (exact field)

⇒

S name

F P.

W age = 25

---

index on (name, zipcode)

① S *
F P
W name = '...'
and zip = '...'

✓

② S *
F P
W name = '...'

✓

③ S *
F P
W zip = '...'

✗

useless (name, zipcode) index for this query

B+tree → good for exact match (pname='gizmo')
→ good for range (50<price and price<100)
→ not good (less effective) multirange (50<price<100 ✓ and 2<q<5 ?)

Creating Index

Create Table V (M, N, P ...);

Create Index V1 on V(N);
... V2 on V(P,M);
→ Default is B+tree if you want hash use "USING HASH"

CLUSTER V USING V2;
→ only one index can be clustered. others are unclustered. Default is generally P.K.

Operations

Union (∪)
Set difference (−)
Selection ($\sigma_{condition}(S)$)
Projection ($\pi_{list\ of\ attr}(S)$)
Cross product (X)
Intersection (∩)
Join ($R \bowtie_\theta S = \sigma_\theta (R \times S)$)

$\rho_{B1,...,Bn}(S)$ → rename

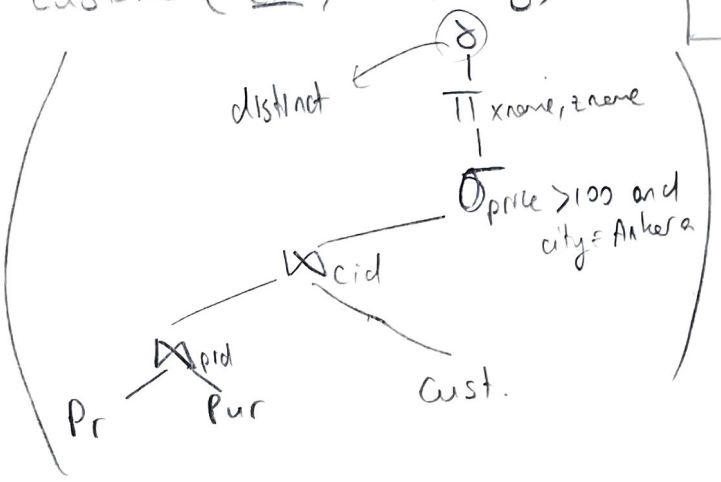
$R1 \cap R2 = R1 - (R1 - R2)$

δ → duplicate eliminator
γ → group by, aggregation (min, max, avg, count)
τ → sorting

Relational Algebra

Product (pid , name , price)
Purchase (pid , cid , store)
Customer (cid , name , city)

S Dist. x.name, z.name
F Pr x, Pur y, Cus z
W x.pid = y.pid and y.cid = z.cid
and x.price > 100 and z.city = Ankara



⇒ Logical plan

B+ tree is good for equality & range queries

select *
F
W N > ? and N < ?

} Fast in B+ tree

Outer Join

Patient

age	zip	disease
54	1	h
20	2	f
33	2	L

Job

job	age	zip
L	54	1
C	20	2

(P ⟕ J) → outer relation, it will include all tuples

age	zip	disease	job
54	1	h	L
20	2	f	C
33	2	L	NULL

Semi Join

Used in distributed databases.

Emp

ssn	name

Dependents

ssn	dname	age

ayrı bilgisayarda

Emp ⋈ssn (σage>71 (Dep))

R = Emp ⋈ (T)

T ← Πssn (σage>71 (Dep))

↓

R = Emp ⋈ T

→ Answer = R ⋈ Dep