emery@cs.umass.edu

SQL – Structured Query Language

Container classes

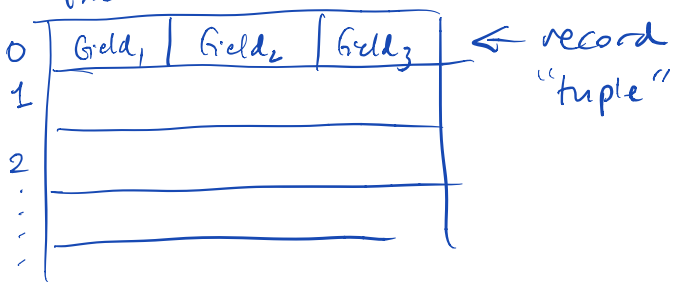
set<T> vector<T>

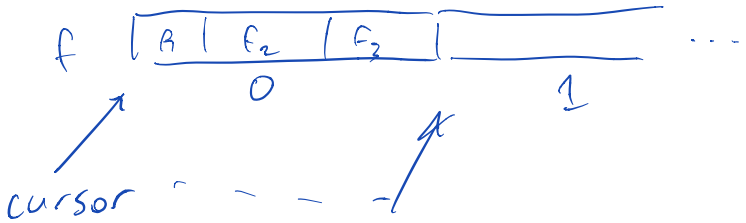
list<T>

⋮

```
vector<int> x;
x[14] = 12;
cout << x[14] << endl;
```

file f





begin
nextField
nextRecord

Salary database Name DNI Salary

```
d = dBopen("salary.db");
cursorBegin(d);
while (cursorPosition(d) != cursorLen(d)) {
  s = readField(d, 3);
  if (s < 10,000) {
  } }
```

1M

procedural

imperative PLs

(Java, C++, Python, PHP, Pascal)

1M entries

sorted by salary ?

```
d = dBopen("salary.db");
cursorBegin(d);
while (cursorPosition(d) != cursorLen(d)) {
  s = readField(d, 3);
  if (s < 10,000) {
    if sorted(d)
  } } else { break; }
```

100 records

inflexible
hard to optimize
breaks abstraction

low selectivity

city = 'Oviedo' ← near 0

and

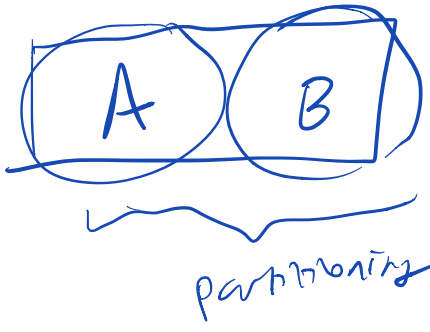
salary > 20.000 Error 100%

Census
EU 300M
each record 1MB

statistics about DB
→ big perf benefit

in-memory DBMS
but expensive
- only for medium workloads (e.g. NOT TB)
- "volatile"
  - not persistent
  - not durable

- indexing



partitioning

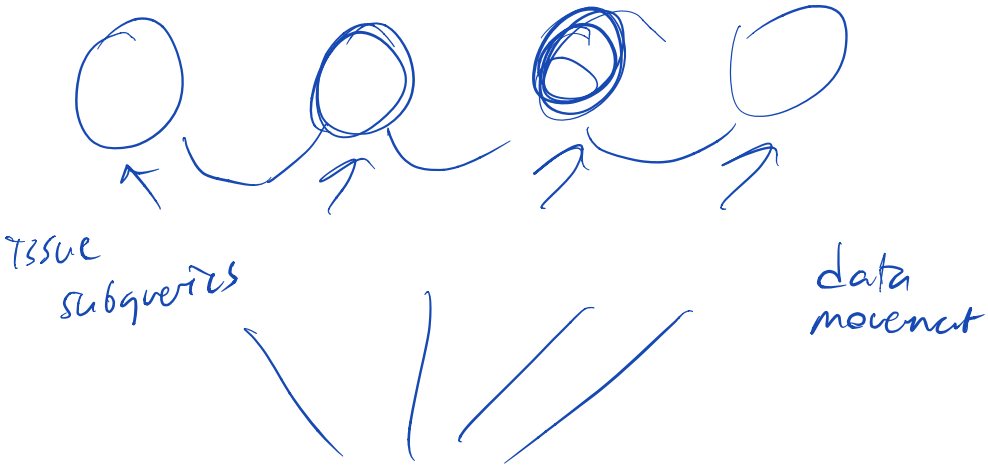
SQL — declarative

```
SELECT emp.name, emp.salary
FROM   emp
WHERE  emp.salary < 20000
```

syntactic sugar
implicit parallelism

query optimizer

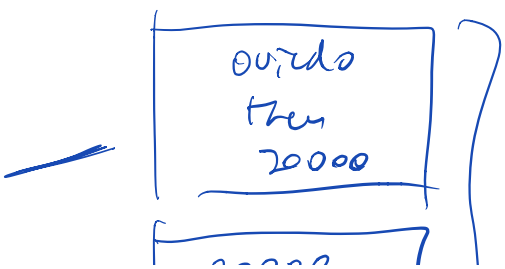
System R IBM (SQL) (DB2)



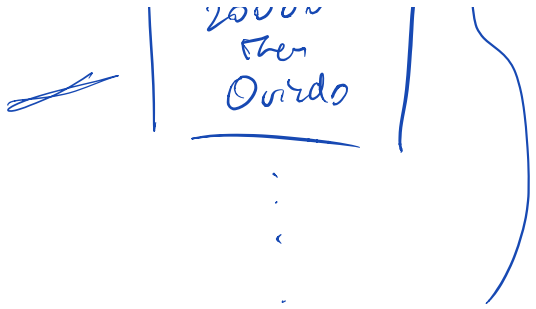
query plans

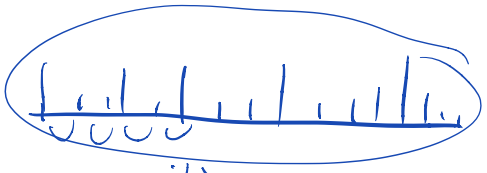
SELECT

WHERE salary < 20000
AND city = 'Ourdo'



histogram by city





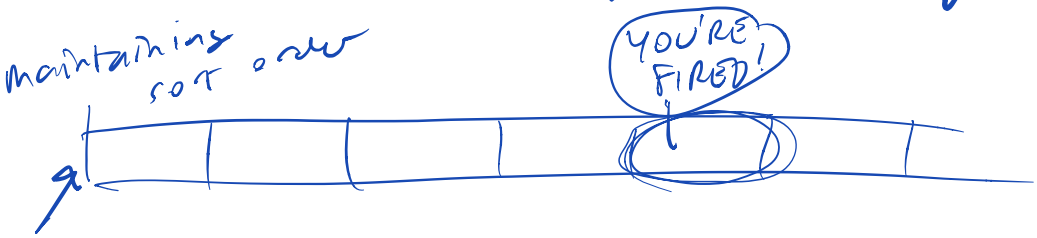
avg # of trips for each city

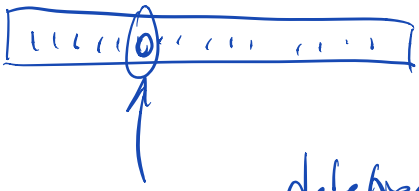
selectivity

cache queries

auto partitioning

leverage indexing



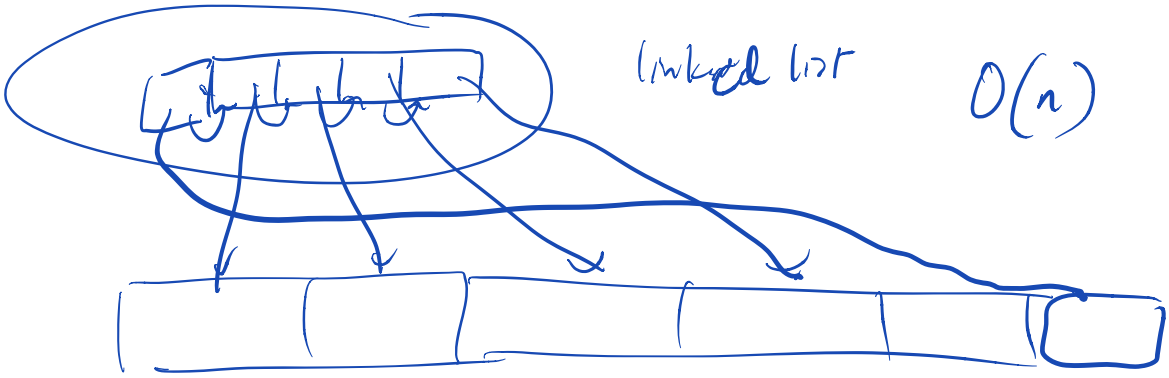


bitmap

deletion "OK" → garbage collection

insertion!

"all problems in CS
can be solved by adding
one level of indirection" =

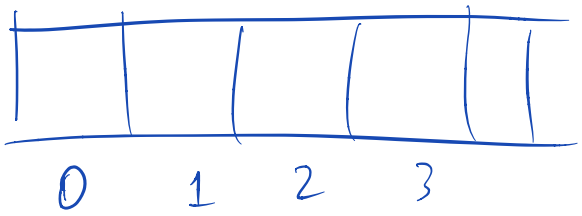


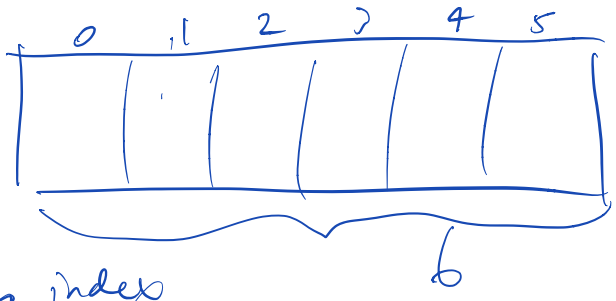
O(log n) — scalable

O(1) — even better

hash table HashMap

hash(key)





DNI

key → index

h(dni) → 0..5

"12345678Q" 1959 ← "nonce"

%

modulo

1231 ← 1 if even, 0 if odd — parity

1+2+3 = 6

Check bit

253 0 X

10

100 % 3 = 1

99 100 101 102

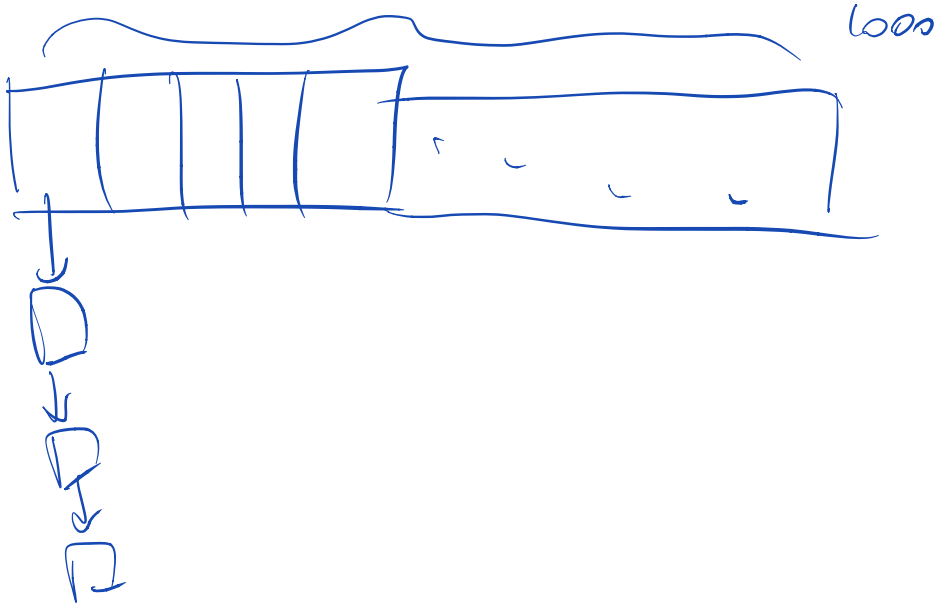
0 1 2 0

1236 — checksum

2530

3934 X

integrity



partitioning — goal: even distribution

salary

name
DNI
salary
city
~~rand~~

rand

how do I ensure "good" partition?

- access pattern?

(OLTP: read-mostly queries)

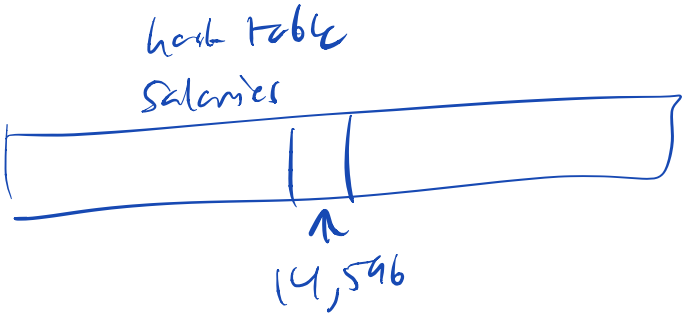
hash partitioning

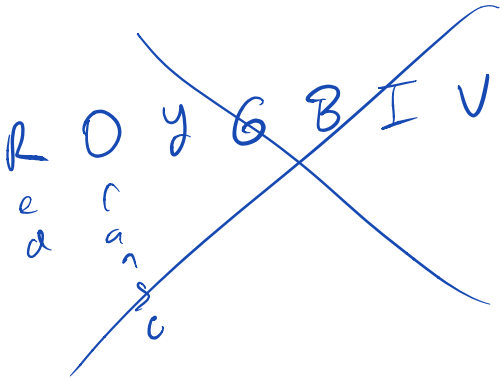
hashCode trivia

Sorting

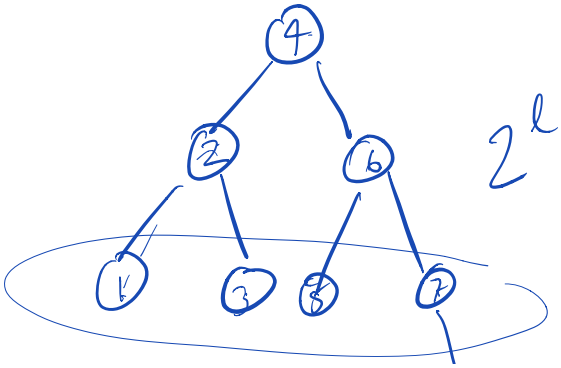
linked lists — SORTED ORDER ~ no moving but O(n)

hash tables — lookups great! Sorting not so much





$O(\log n)$



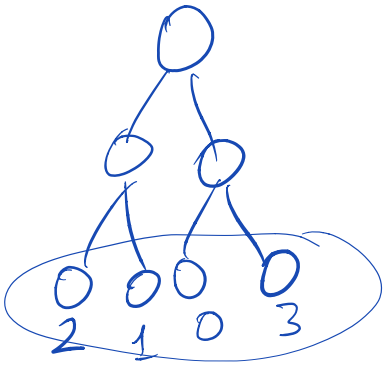
$2^l - 1$

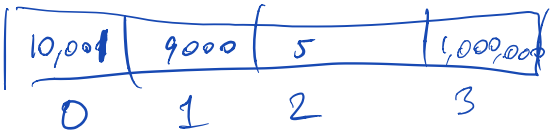
2^{l-1}

balanced (binary) tree

Red Black tree

AUL tree

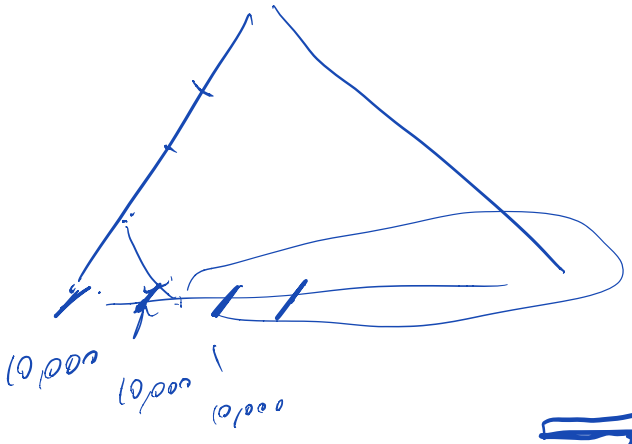


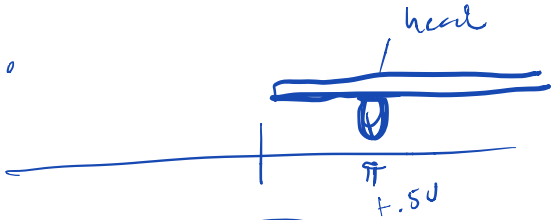


"spinning media"

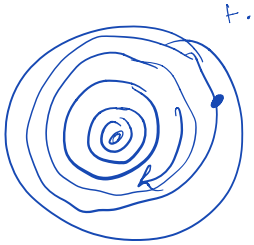
HDD — conventional disc

SSD — Flash





Seek



10,000 RPM

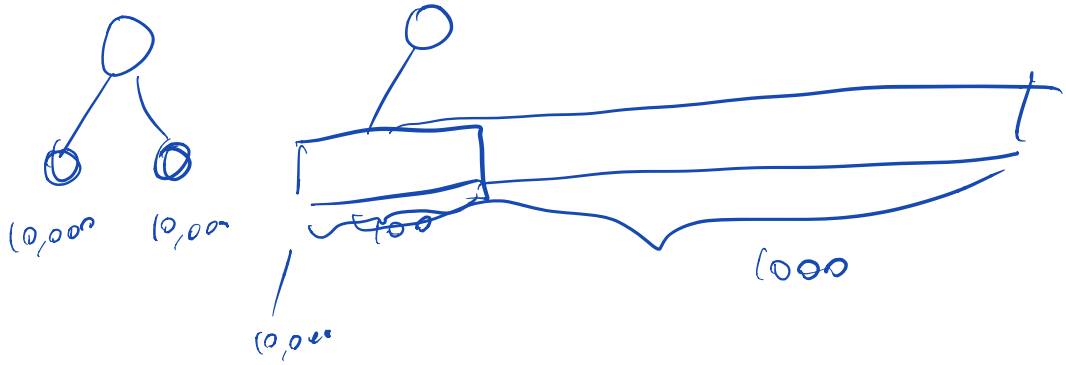
seek latency >> rotation latency ~> 0

disk bandwidth

SSD: ~0 seek latency

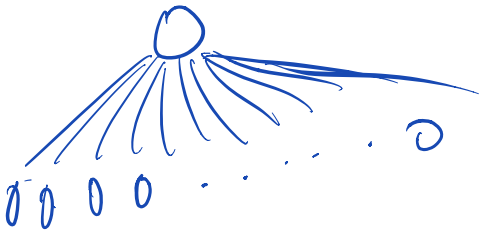
bw < disk bw

cost ~ 10x



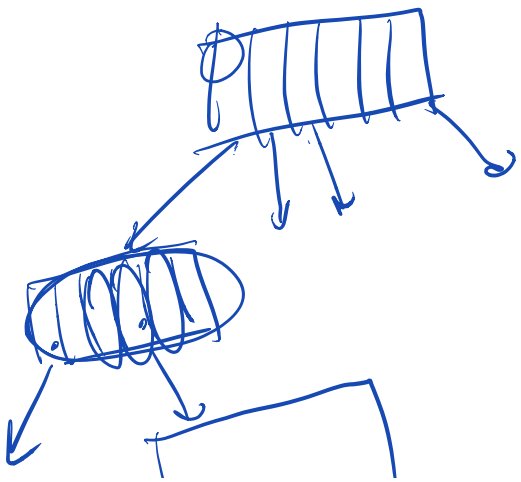
10,000 × 1000

10,000 × 1

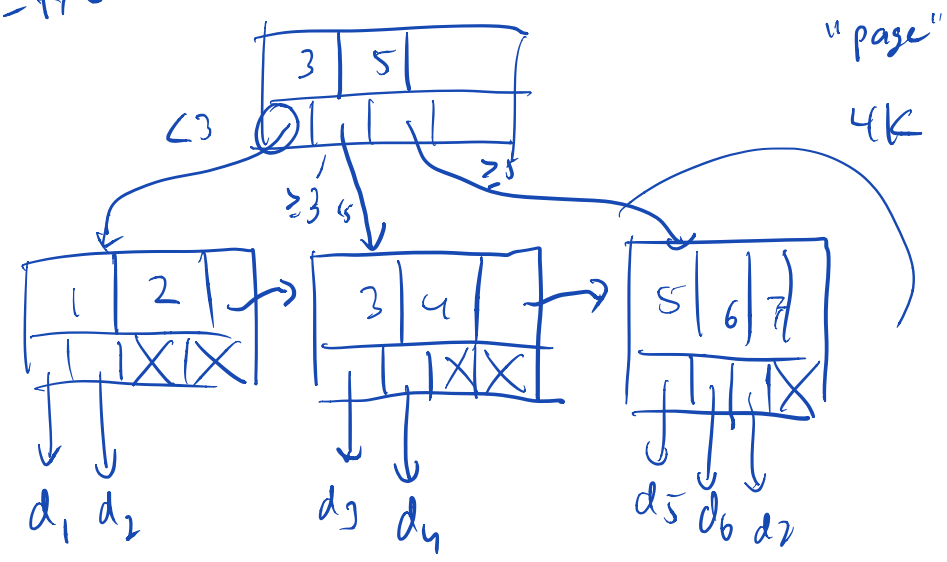


B-tree

B+-tree



B+-tree



"page"
4K

---

index all fields?

- storage
- update linearly w/ # of indexes

city, name

↑ primary key

↑ secondary key

SELECT
:
GROUP BY city
"SORTED" BY city, name

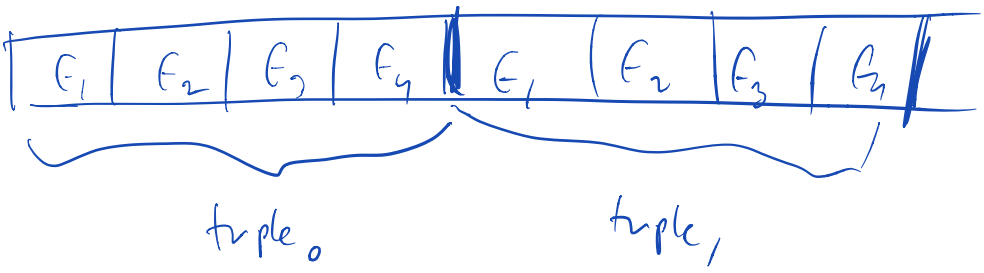
4 fields ~ 4 indexes

pairs? $\binom{4}{2}$ $\frac{4!}{2! \cdot 2!}$ $\frac{4 \cdot 3 \cdot 2 \cdot 1}{2 \cdot 1 \cdot 2 \cdot 1}$ ~ 6

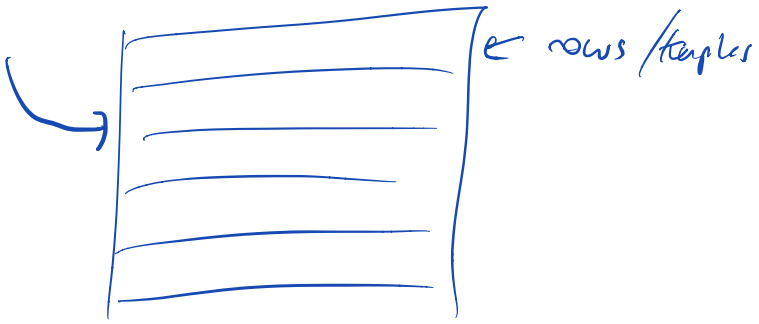
100 fields — 100 indices

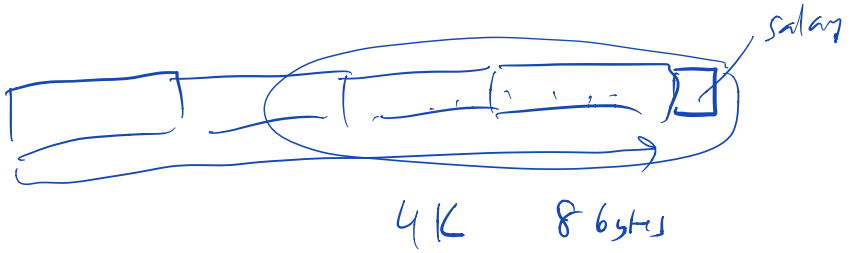
$\binom{100}{2}$ $\frac{100!}{2!\,98!}$ $\frac{100 \cdot 99}{2} \approx 5{,}000$

DB administrator



row-based layout



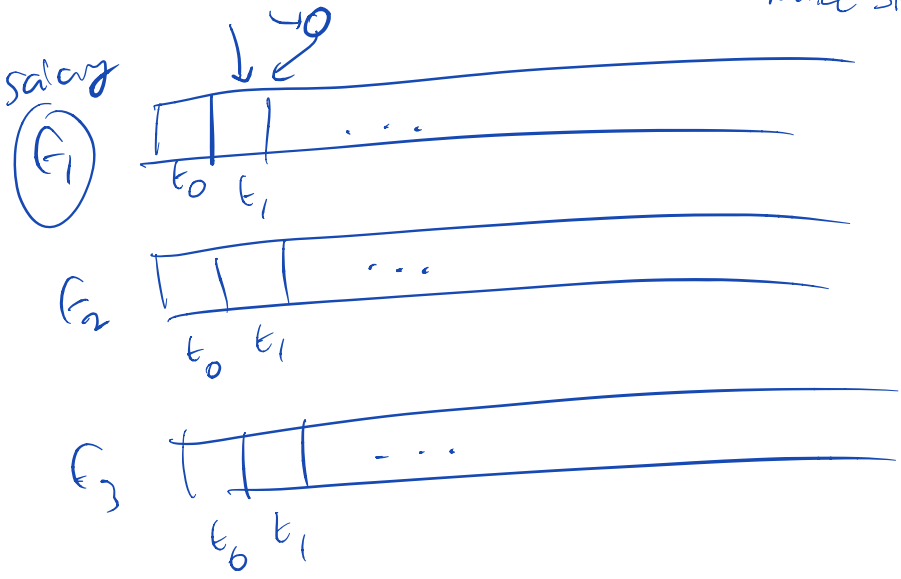


Column-based layout

Vertica

Mike Stonebraker

MIT



SIGMOD

VLDB