BOSTON UNIVERSITY

CS 561: Data Systems Architectures

class 3

Relational Recap & Column-Stores Basics

Prof. Manos Athanassoulis

https://bu-disc.github.io/CS561/

what to do now?

A) read the syllabus and the website
B) register to Piazza + Gradescope
C) finish project 0 (due 1/31)
D) start working on project 1 (due 2/14)
E) register for the presentation (week 2-3)
F) start reading papers & prepare for tech. questions (week 3)
G) go over the class project (end of next week will be available)
H) start working on the proposal (week 3)

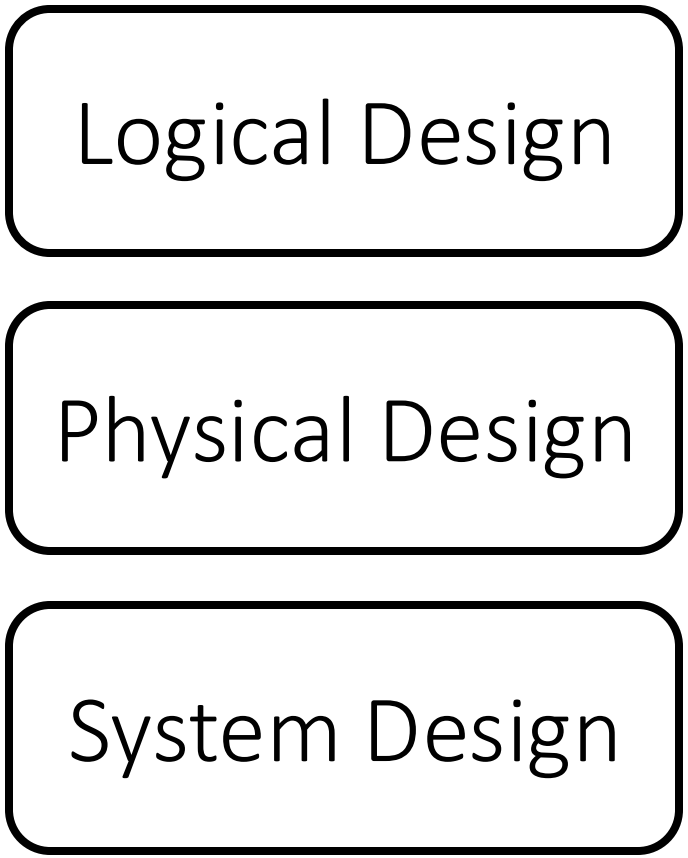
How can I prepare?

1) Read background research material

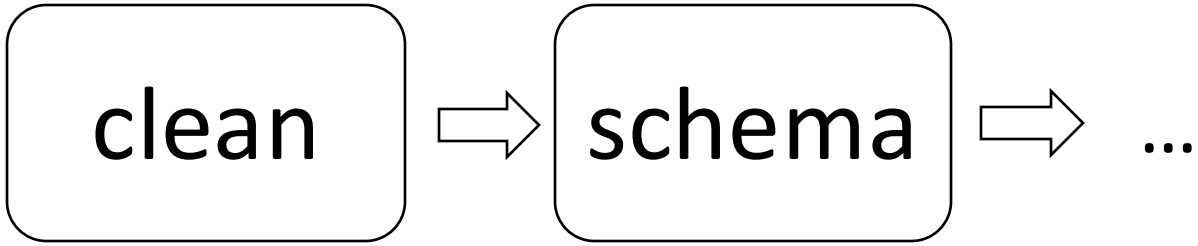
- **Architecture of a Database System**.
  By J. Hellerstein, M. Stonebraker and J. Hamilton.
  Foundations and Trends in Databases, 2007
- **The Design and Implementation of Modern Column-store Database Systems**.
  By D. Abadi, P. Boncz, S. Harizopoulos, S. Idreos, S. Madden.
  Foundations and Trends in Databases, 2013
- **Data Structures for Data-Intensive Applications: Tradeoffs and Design Guidelines.**
  By M. Athanassoulis, S. Idreos, D. Shasha.
  Foundations and Trends in Databases, 2024

2) Start going over the papers

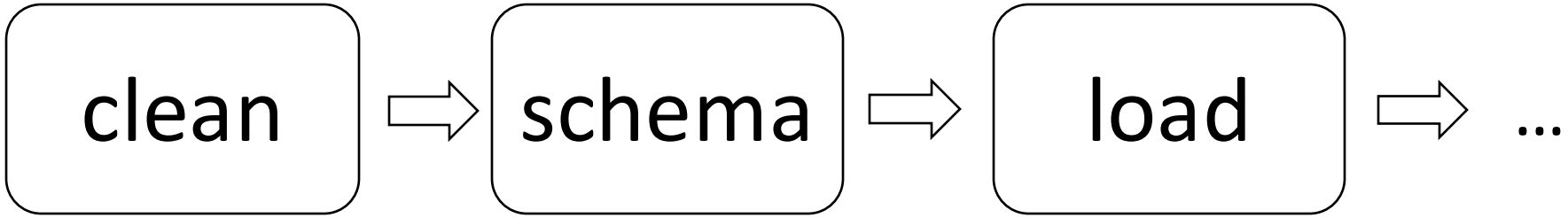
Database Design Abstraction Levels



Data can be messy!

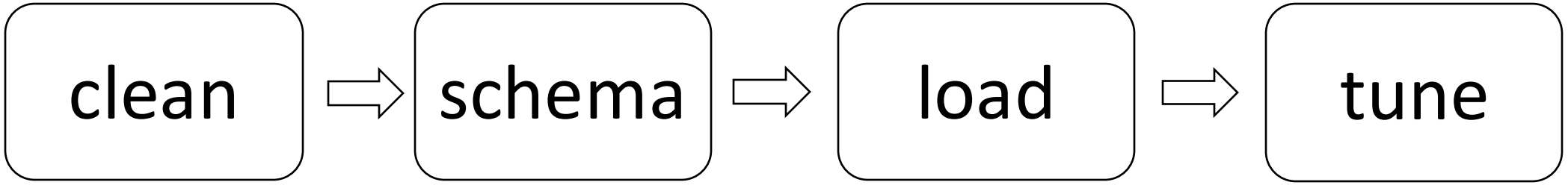


Data can be messy!

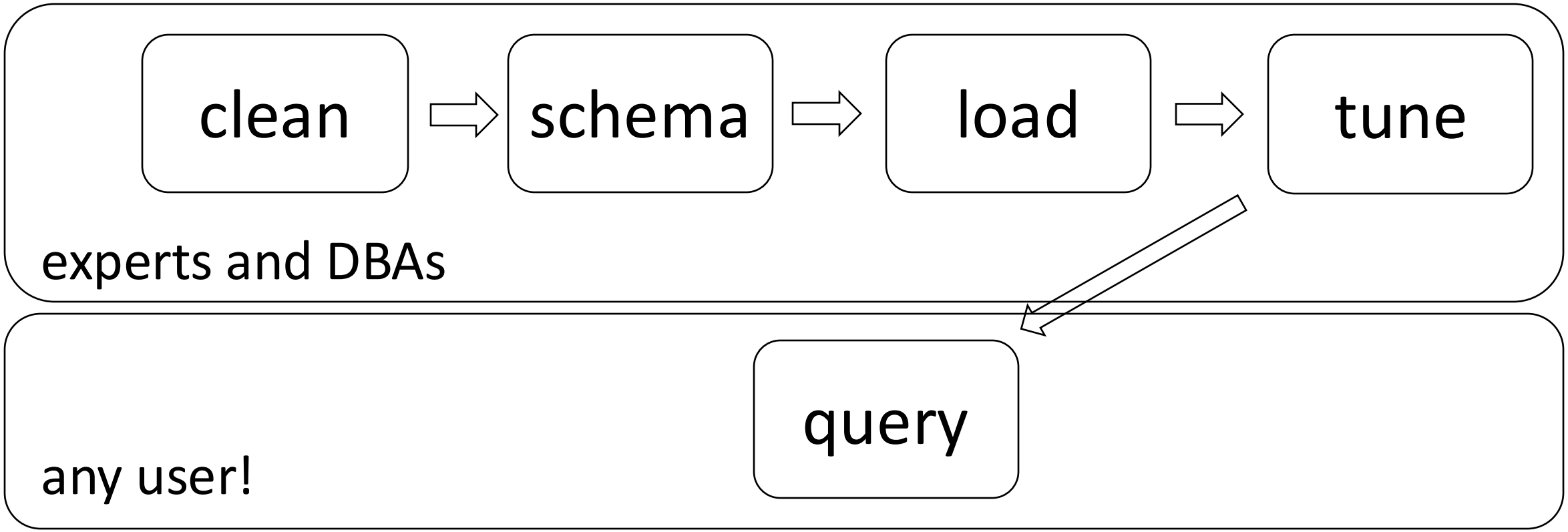


Data can be messy!

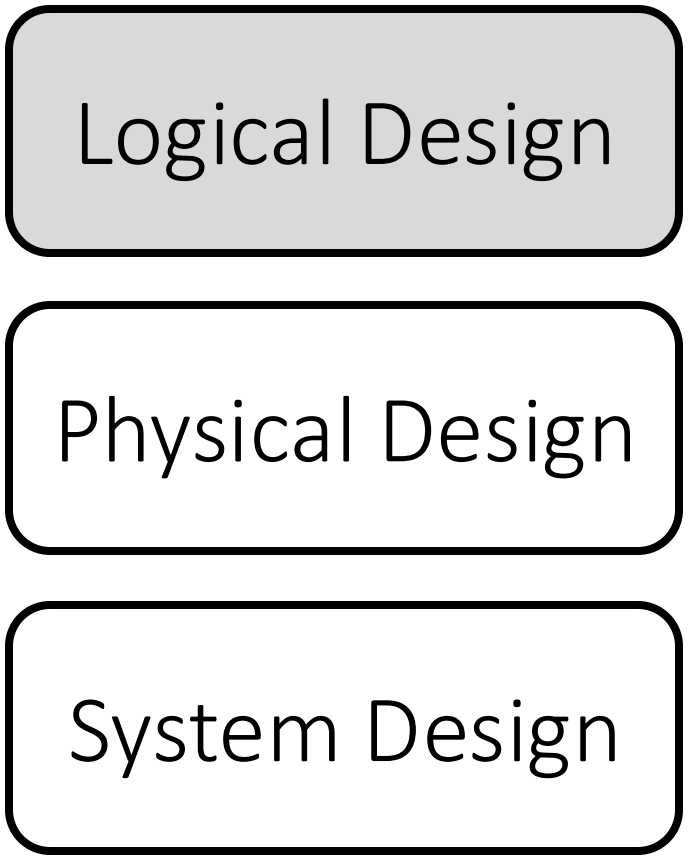
what kind of indexes
size of memory buffer
how many threads to use
…



Data can be messy!



Database Design Abstraction Levels



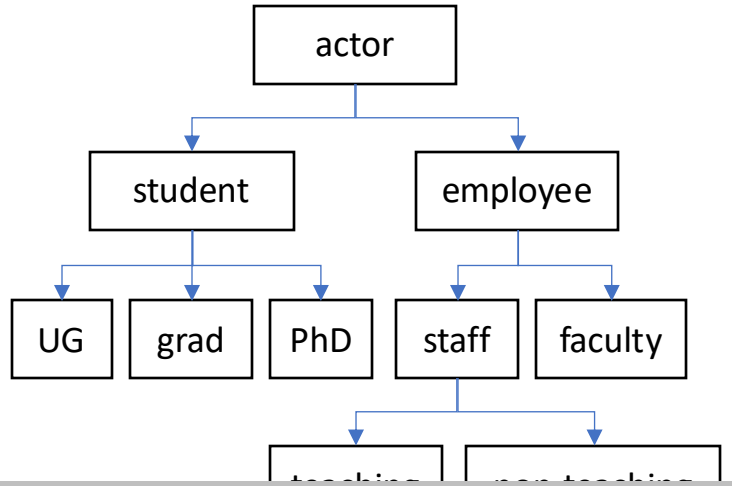
Logical design

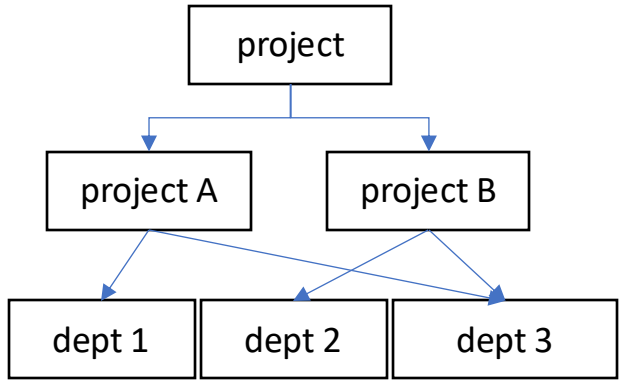
What is our data? How to model them?

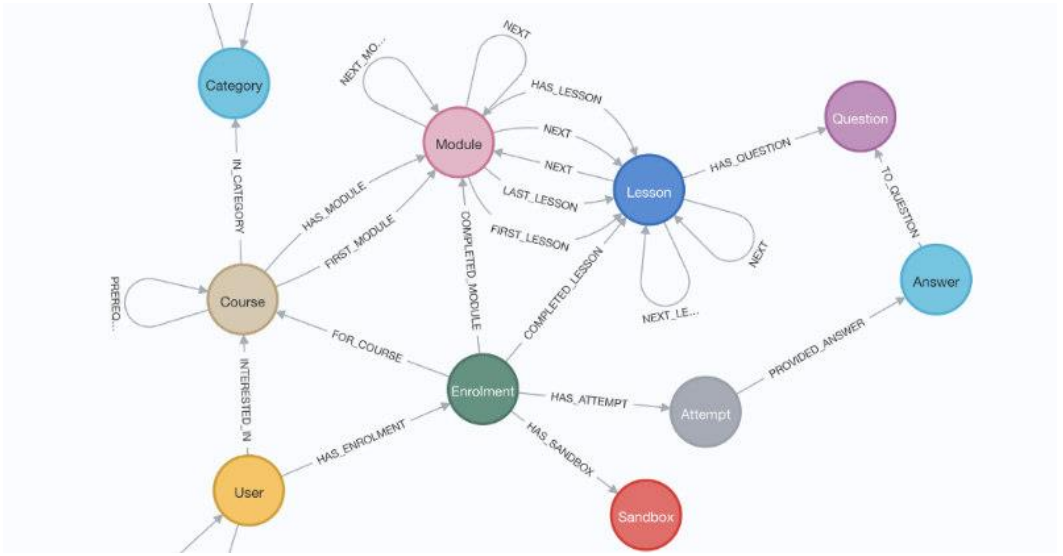
Hierarchical? Network? Object-oriented? Flat?

Logical design

What is our data? How to model them?







relational data model

key-value data model

Logical design

What is our data? How to model them?

Hierarchical? Network? Object-oriented? Flat?

Relational & Key-value

A collection of **tables**, each being a collection of **rows and columns**

[schema: describes the columns of each table]

Logical Schema of "University" Database

Students

***sid**: string, **name**: string, **login**: string, **year_birth**: integer, **gpa**: real*

Courses

***cid**: string, **cname**: string, **credits**: integer*

Enrolled

***sid**: string, **cid**: string, **grade**: string*

attributes for *Enrolled* ?

BOSTON UNIVERSITY

Relational Model and SQL

relations

keys

Students

***sid**: string, **name**: string, **login**: string, **year_birth**: integer, **gpa**: real*

Courses

***cid**: string, **cname**: string, **credits**: integer*

Enrolled

***sid**: string, **cid**: string, **grade**: string*

PK for *Enrolled* ?

Relational Model and SQL

how to create the table students?

create table *students (sid:char(10), name:char(40), login:char(8), age:integer, …)*

Students

sid*: string,* ***name****: string,* ***login****: string,* ***year_birth****: integer,* ***gpa****: real*

how to add a new student?

insert into *students (U1398217312, John Doe, john19, 19, …)*

Courses

cid*: string,* ***cname****: string,* ***credits****: integer*

bring me the names of all students

select *name* ***from*** *students* ***where*** *GPA > 3.5*

Enrolled

sid*: string,* ***cid****: string,* ***grade****: string*

Relational Model and SQL

student

(sid1, name1, login1, year1, gpa1)
(sid2, name2, login2, year2, gpa2)
(sid3, name3, login3, year3, gpa3)
(sid4, name4, login4, year4, gpa4)
(sid5, name5, login5, year5, gpa5)
(sid6, name6, login6, year6, gpa6)
(sid7, name7, login7, year7, gpa7)
(sid8, name8, login8, year8, gpa8)
(sid9, name9, login9, year9, gpa9)

cardinality: 9

Relational Model and SQL

student

(sid1, name1, login1, year1, gpa1)
(sid2, name2, login2, year2, gpa2)
(sid3, name3, login3, year3, gpa3)
(sid4, name4, login4, year4, gpa4)
(sid5, name5, login5, year5, gpa5)
(sid6, name6, login6, year6, gpa6)
(sid7, name7, login7, year7, gpa7)
(sid8, name8, login8, year8, gpa8)
(sid9, name9, **login9**, year9, gpa9)

cardinality: 9

what if a student does not have a login ID yet?

Relational Model and SQL

student

(sid1, name1, login1, year1, gpa1)
(sid2, name2, login2, year2, gpa2)
(sid3, name3, login3, year3, gpa3)
(sid4, name4, login4, year4, gpa4)
(sid5, name5, login5, year5, gpa5)
(sid6, name6, login6, year6, gpa6)
(sid7, name7, login7, year7, gpa7)
(sid8, name8, login8, year8, gpa8)
(sid9, name9, **NULL**, year9, gpa9)

cardinality: 9

what if a student does not have a login ID yet?

Relational Model and SQL

how to show all enrollments in CS561?

keys

Students

***sid**: string, **name**: string, **login**: string, **year_birth**: integer, **gpa**: real*

Courses

***cid**: string, **cname**: string, **credits**: integer*

Enrolled

***sid**: string, **cid**: string, **grade**: string*

Relational Model and SQL

how to show all enrollments in DSA?

Students

***sid**: string, **name**: string, **login**: string, **year_birth**: integer, **gpa**: real*

Courses

***cid**: string, **cname**: string, **credits**: integer*

Enrolled

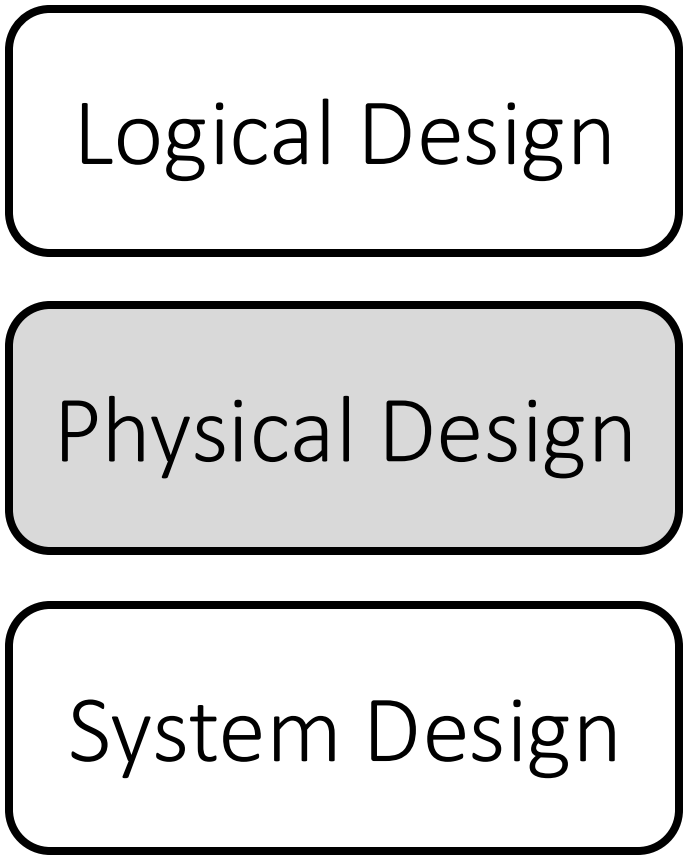
***sid**: string, **cid**: string, **grade**: string*

foreign keys

using foreign keys we can join information of all three tables

select student.name
from students, courses, enrolled
where course.cname="DSA"
and course.cid=enrolled.cid
and student.sid=enrolled.sid

Database Design Abstraction Levels



Physical Design

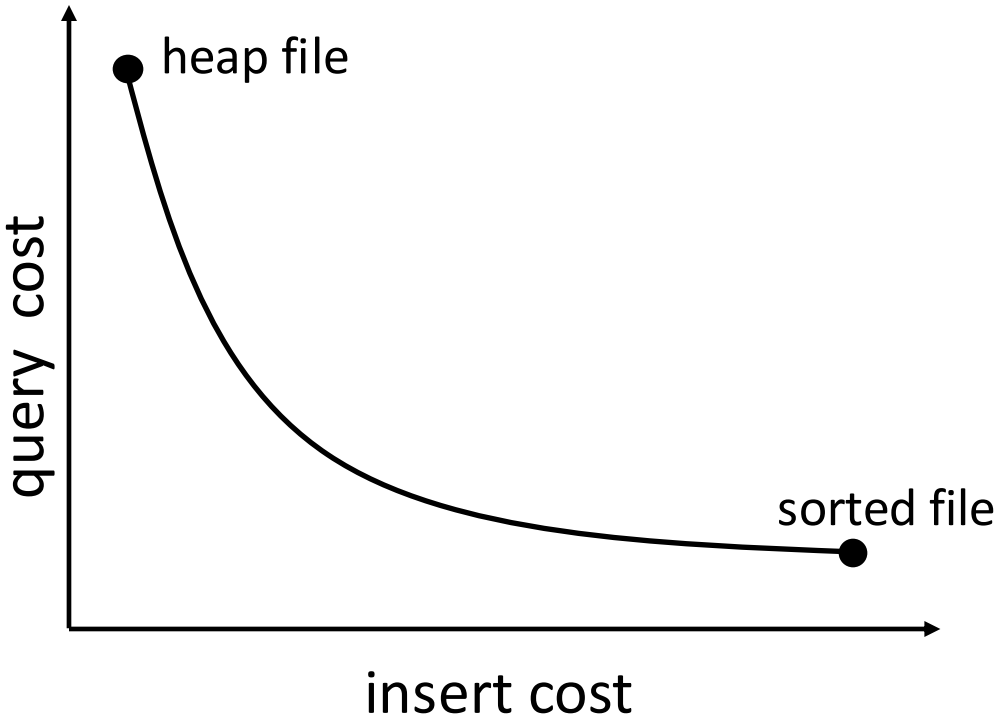
File Organization

heap files

sorted files

clustered files

more …



Physical Design

File Organization

heap files

sorted files

clustered files

more …

Indexes

should I build an index?

on which attributes/tables?

what index structure?

B-Tree Trie

Hash Bitmap

Zonemap

Physical Design

File Organization

heap files

sorted files

clustered files

more …

Indexes

should I build an index?

on which attributes/tables?

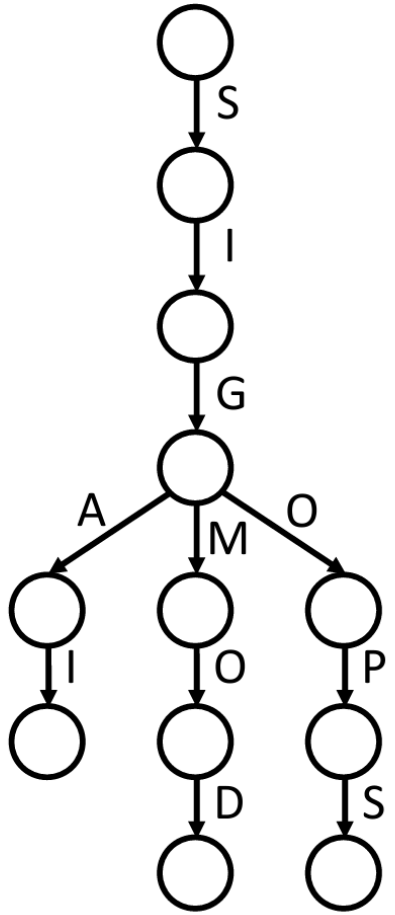
what index structure?

B-Tree Trie

Hash Bitmap

Zonemap

trie?



k-ary prefix tree

Physical Design

File Organization

heap files

sorted files

clustered files

more …

Indexes

should I build an index?

on which attributes/tables?

what index structure?

B-Tree Trie

Hash Bitmap

Zonemap

bitmap?

rid	Column
1	30
2	20
3	30
4	10
5	20
6	10
7	30
8	20

data

rid	10	20	30
1	0	0	1
2	0	1	0
3	0	0	1
4	1	0	0
5	0	1	0
6	1	0	0
7	0	0	1
8	0	1	0

bitmap

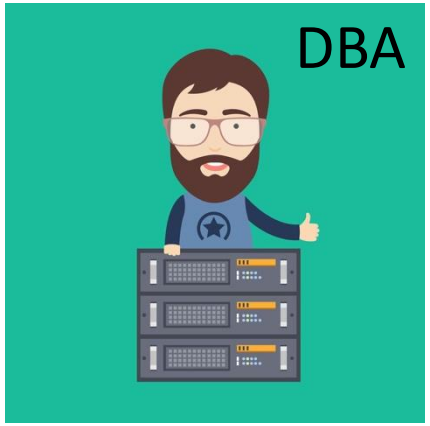
works great for columns with few distinct values

Data systems are declarative!

ask ***what*** you want

data system

system decides ***how***
to store & access

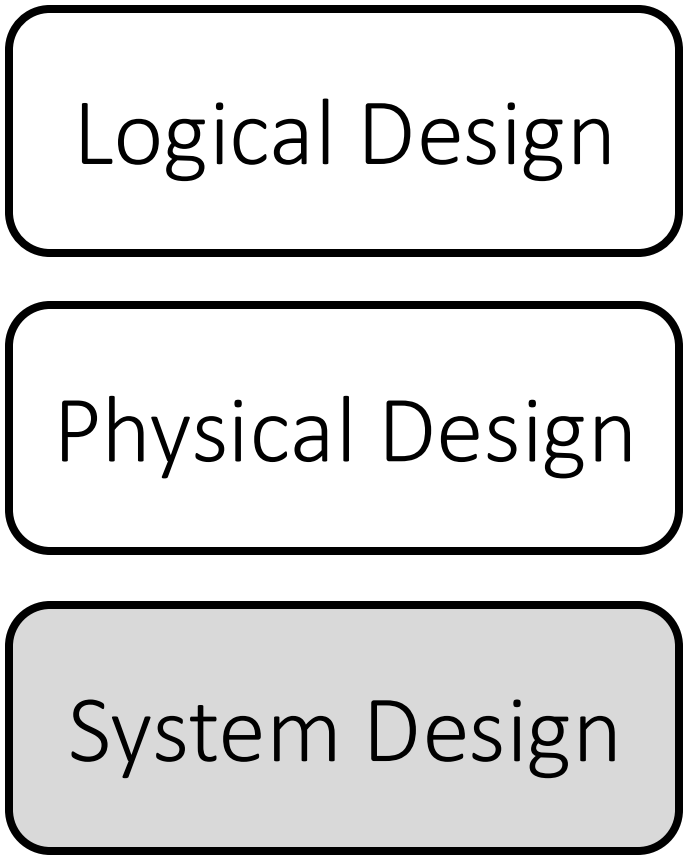


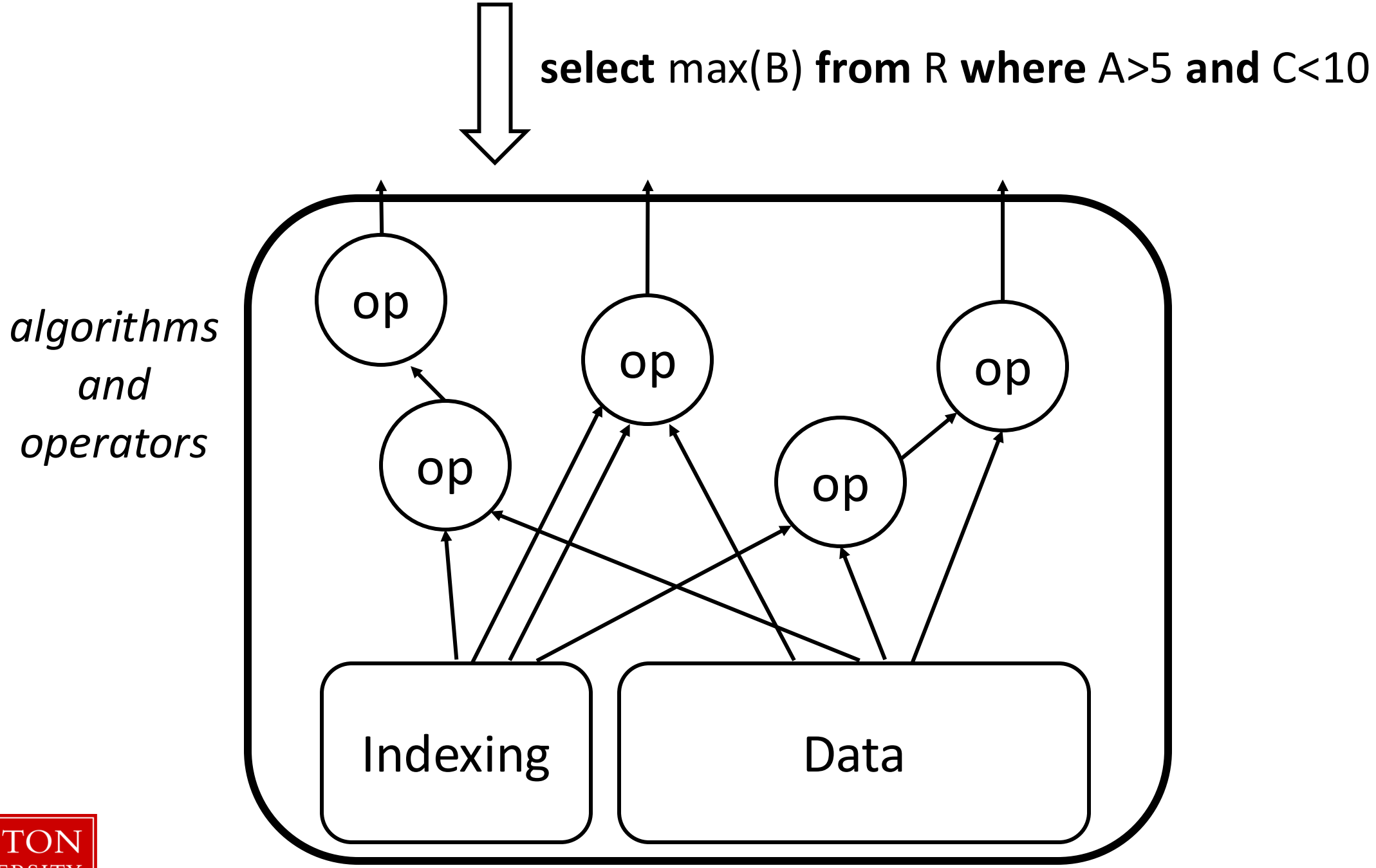
design decisions, physical design
indexing, tuning knobs

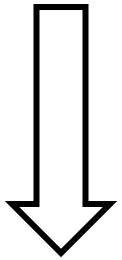
research to automate!

adaptivity **autotuning**

Database Design Abstraction Levels







select max(B) **from** R **where** A>5 **and** C<10

modules

Parser

Optimizer

Evaluation

Storage

registers/CPU

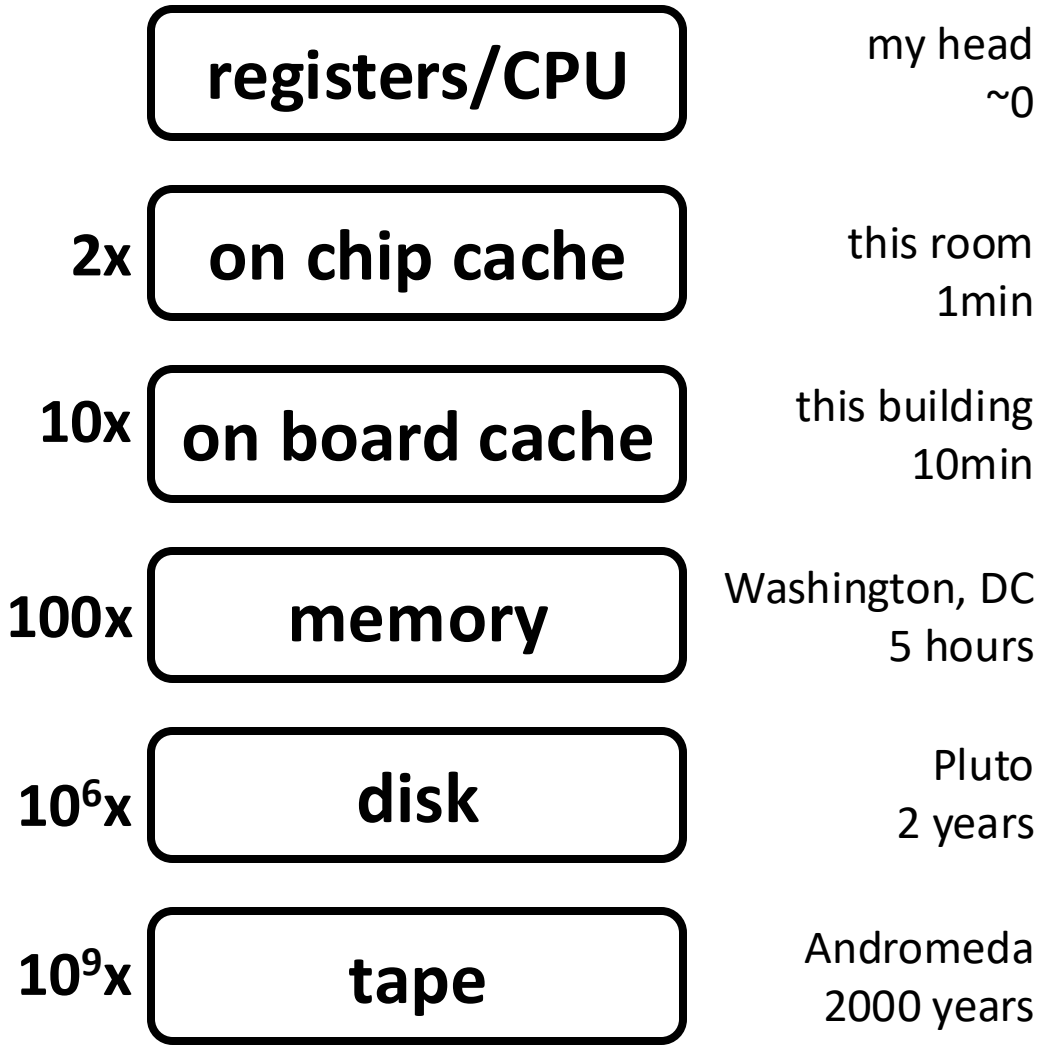
on chip cache

on board cache

memory

disk

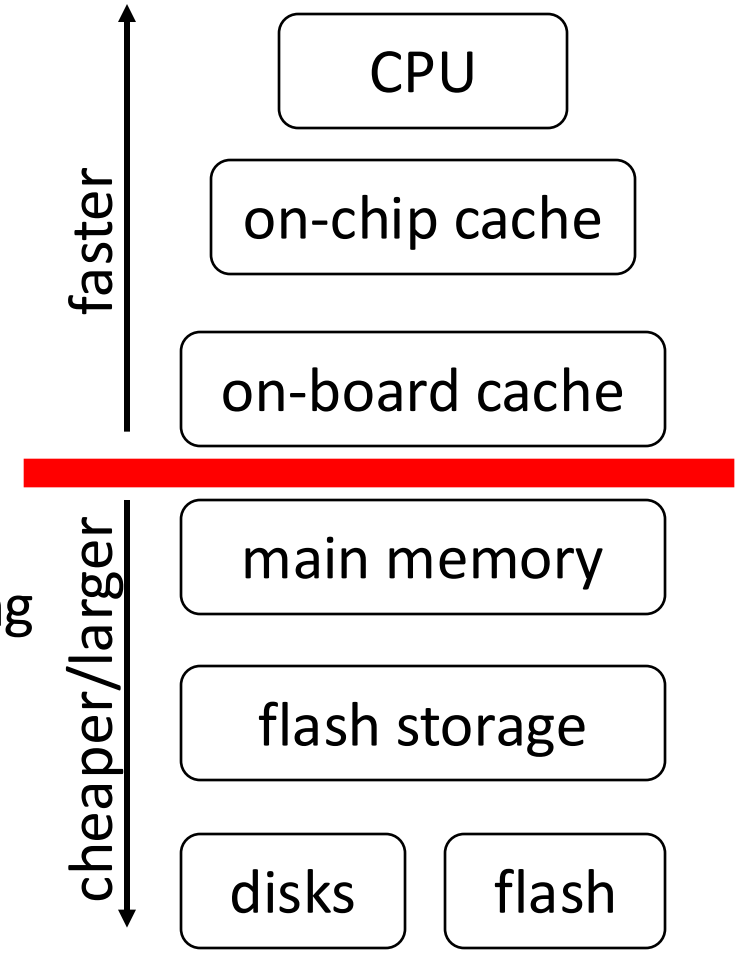
tape

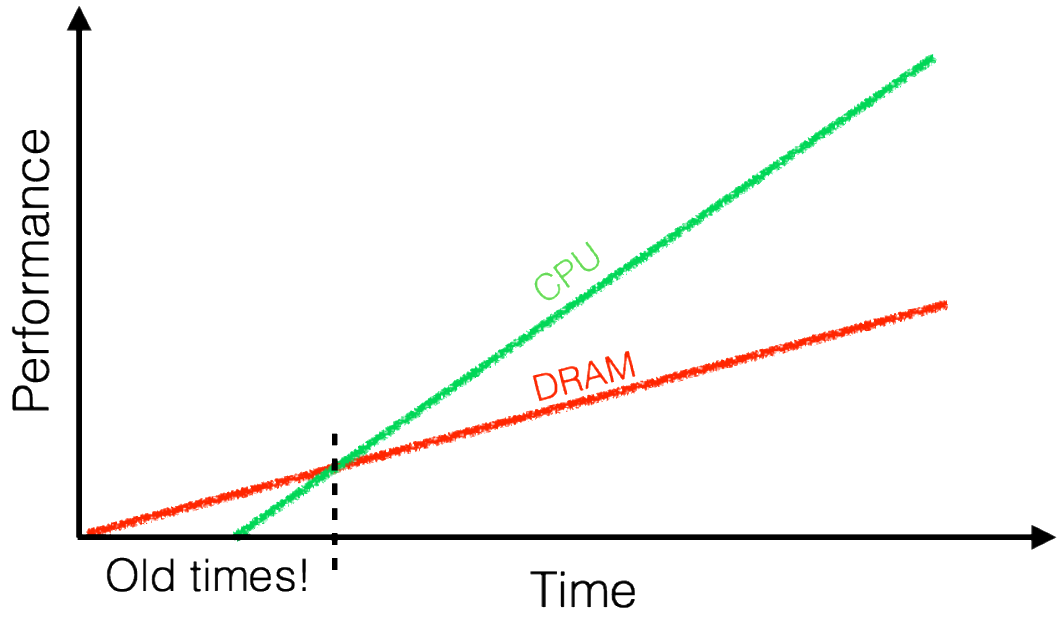


memory wall

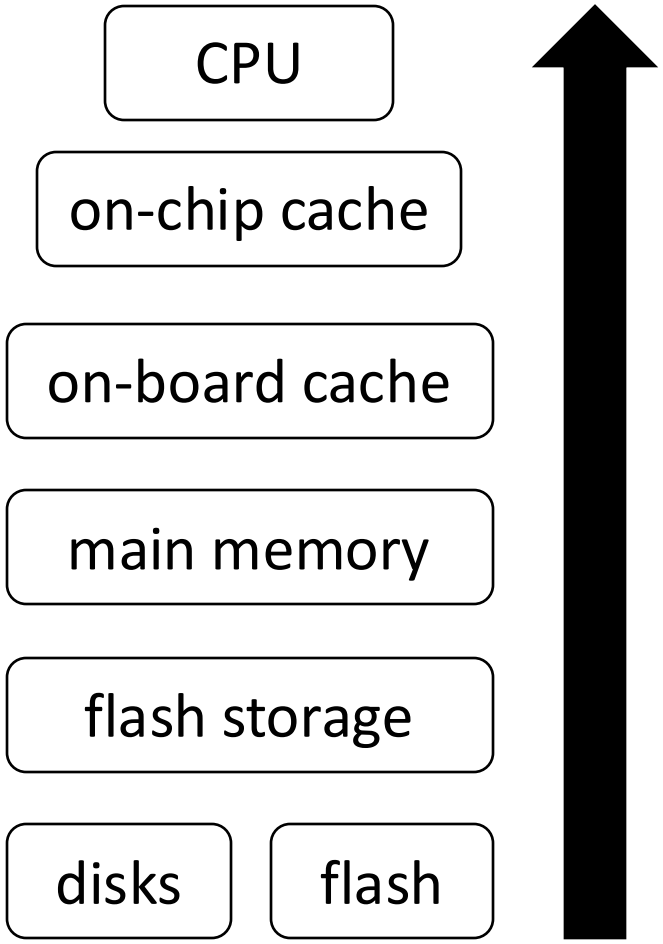
cache miss: looking for something that is not in the cache

memory miss: looking for something that is not in memory





data movement & page-based access

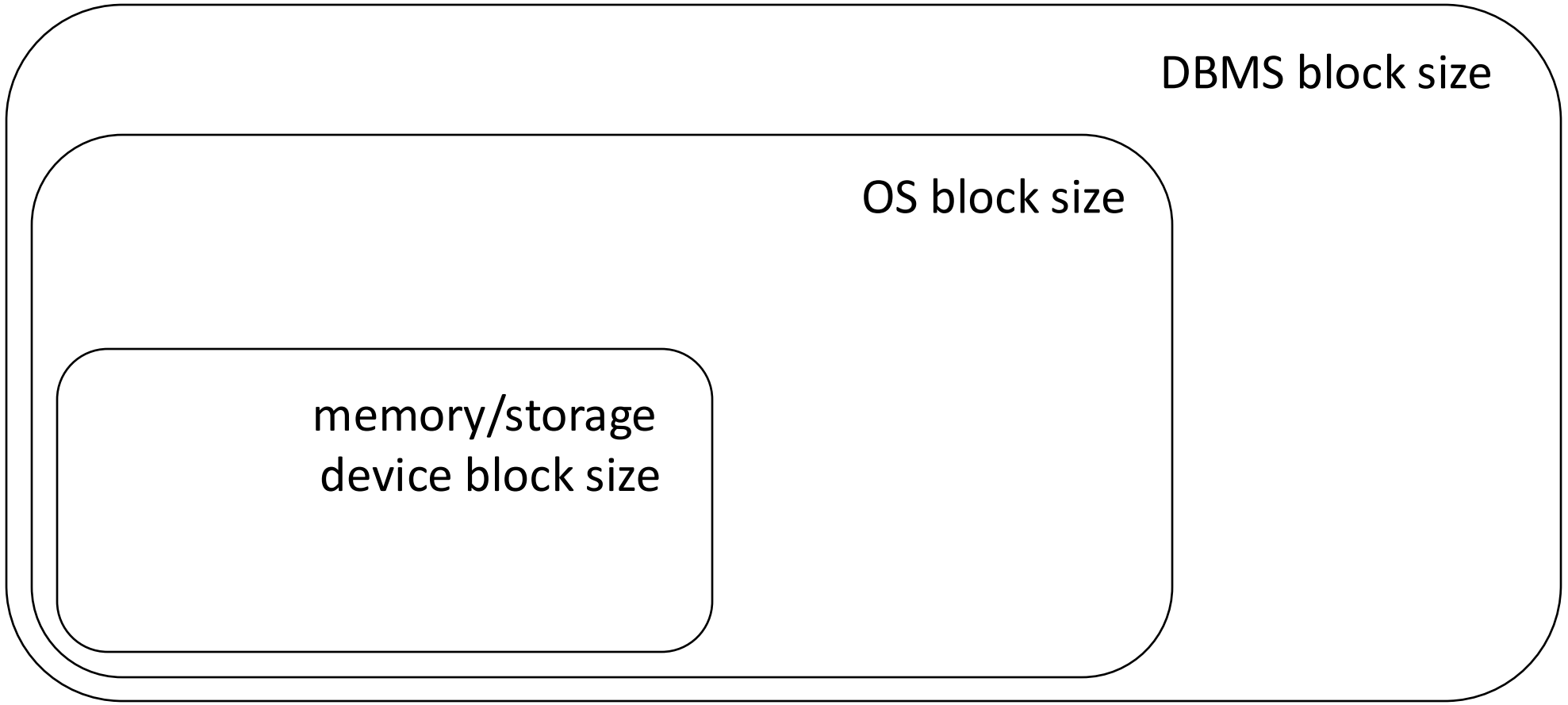


data go through
all necessary levels

also read
unnecessary data

need to read only X
read the whole page

access granularity



file system and DBMS “pages”

understanding data placement

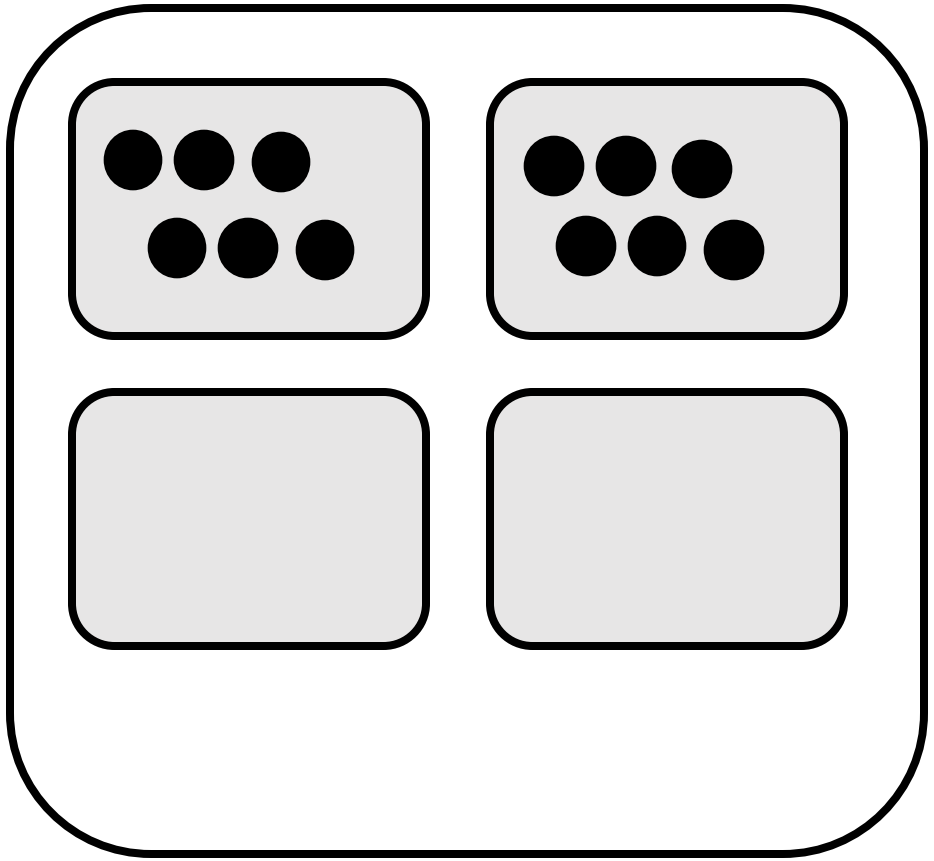
data storage

how to physically place data?

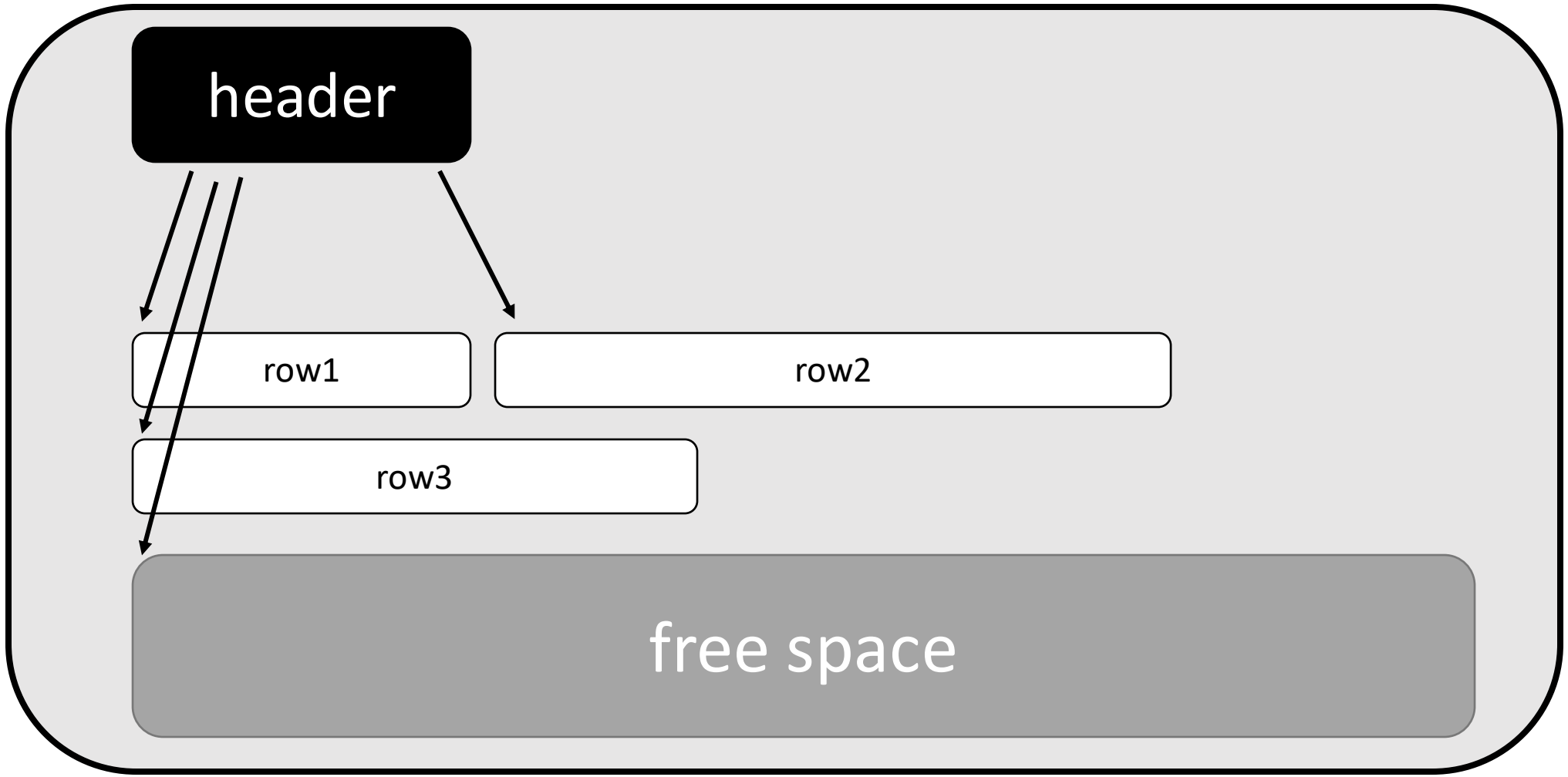
*Student (**sid**: string, **name**: string, **login**: string, **year_birth**: integer, **gpa**: real)*

student

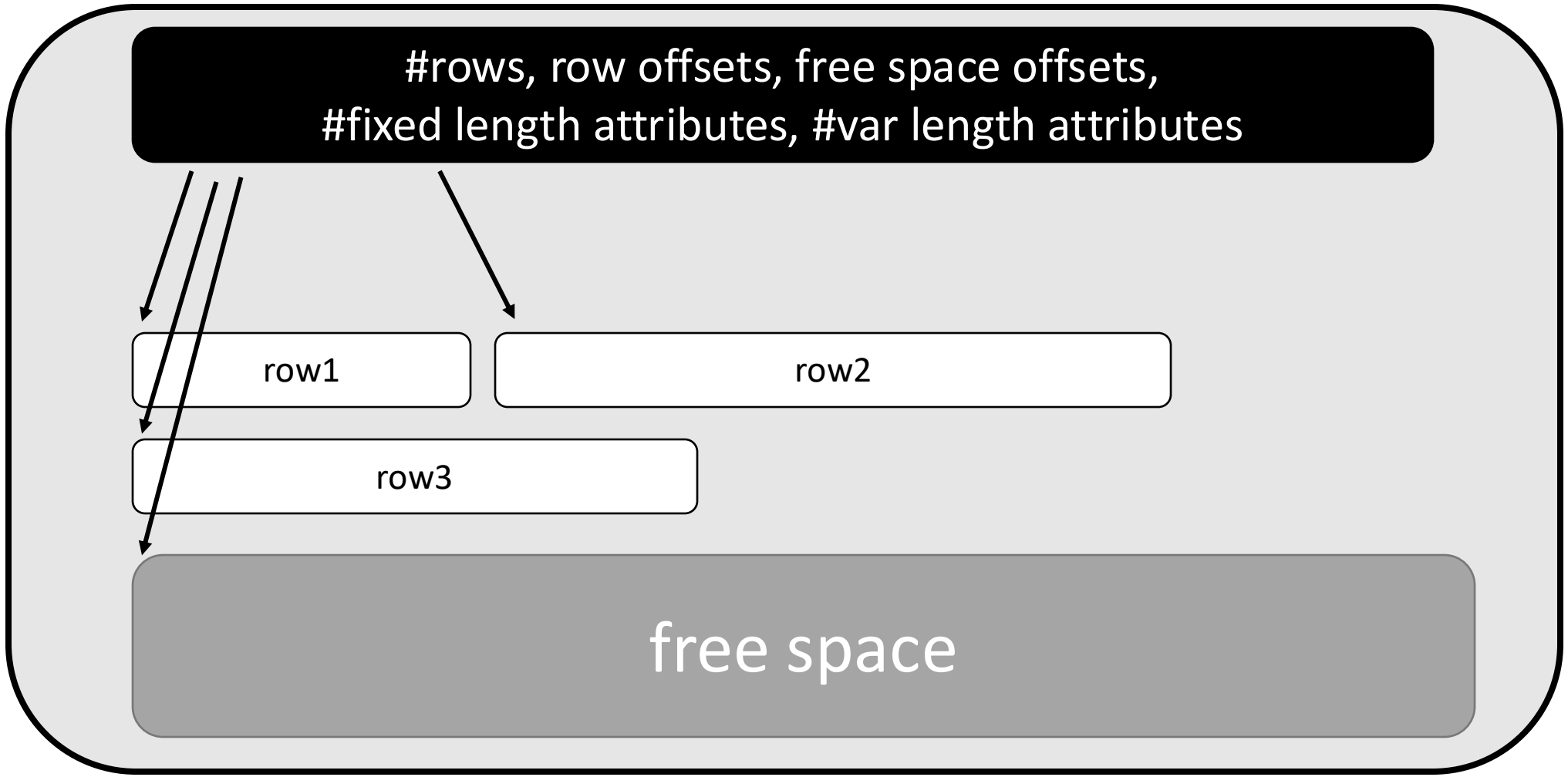
(sid1, name1, login1, year1, gpa1)
(sid2, name2, login2, year2, gpa2)
(sid3, name3, login3, year3, gpa3)
(sid4, name4, login4, year4, gpa4)
(sid5, name5, login5, year5, gpa5)
(sid6, name6, login6, year6, gpa6)
(sid7, name7, login7, year7, gpa7)
(sid8, name8, login8, year8, gpa8)
(sid9, name9, login9, year9, gpa9)



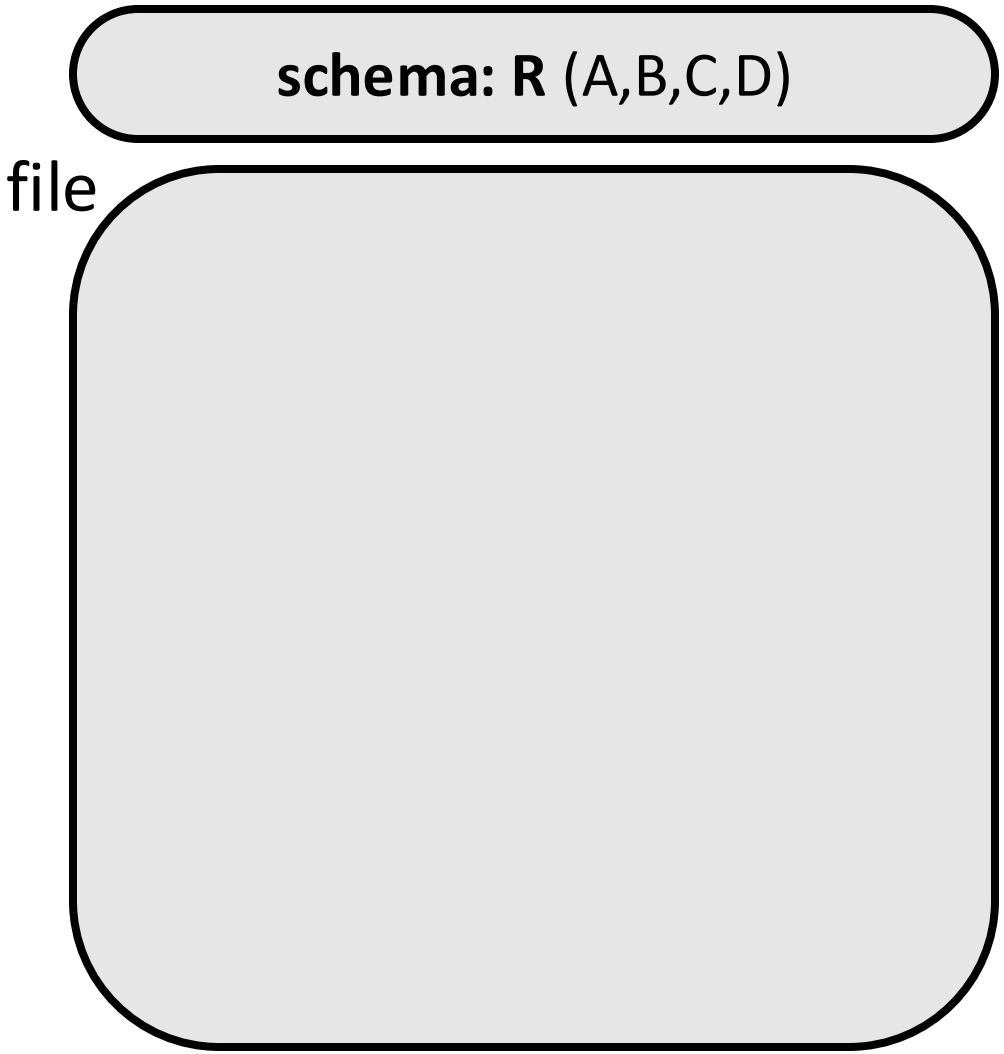
slotted page



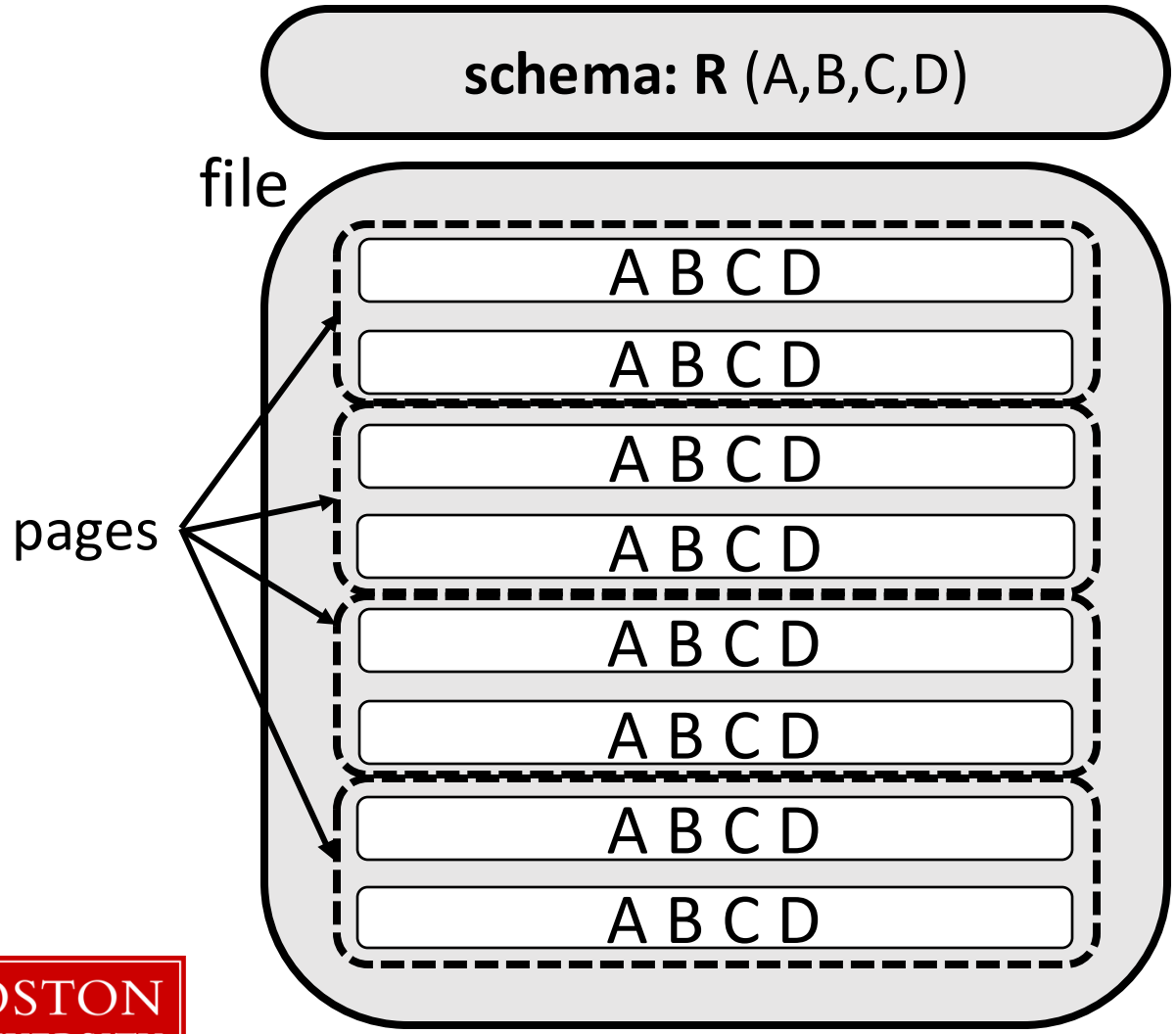
slotted page



querying over slotted pages



querying over slotted pages



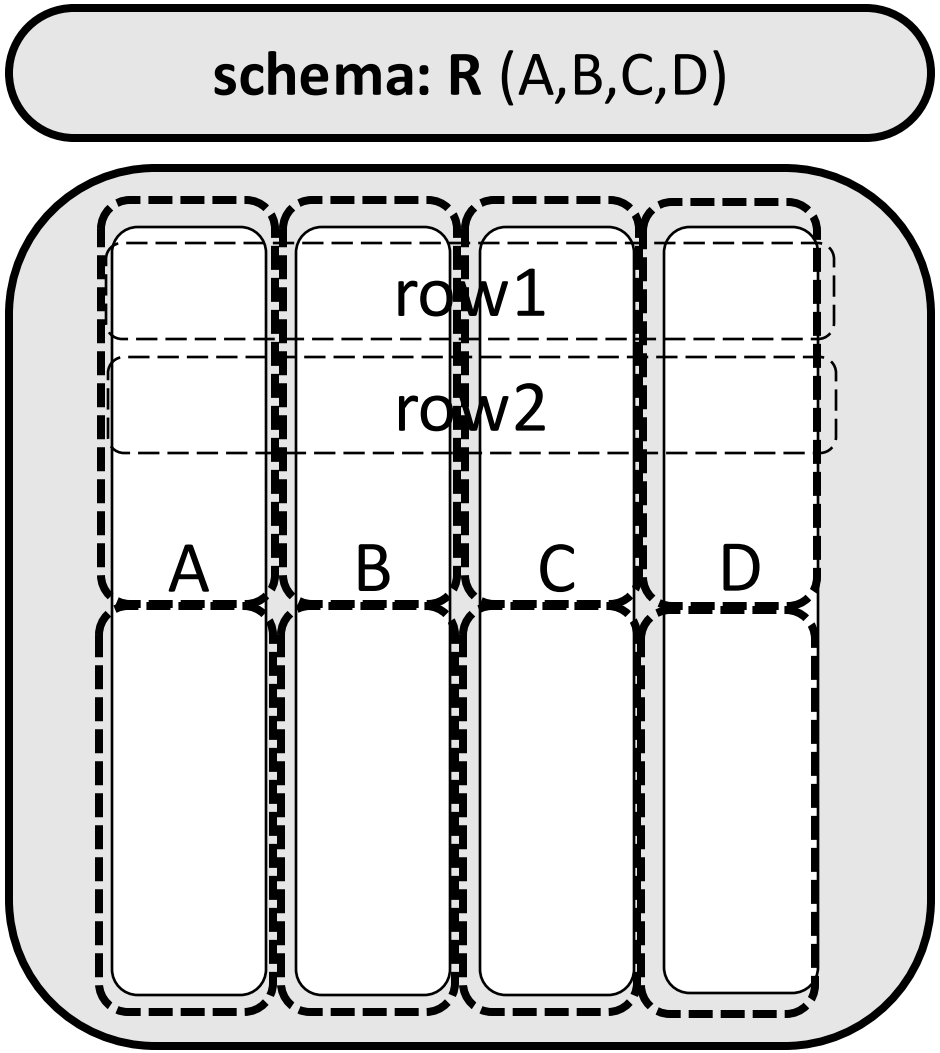
select A,B,C,D **from** R

select A **from** R

each page contains **entire** rows (all their columns)

rows are **contiguous**
(with possible free space at the end)

querying over slotted pages



select A,B,C,D **from** R

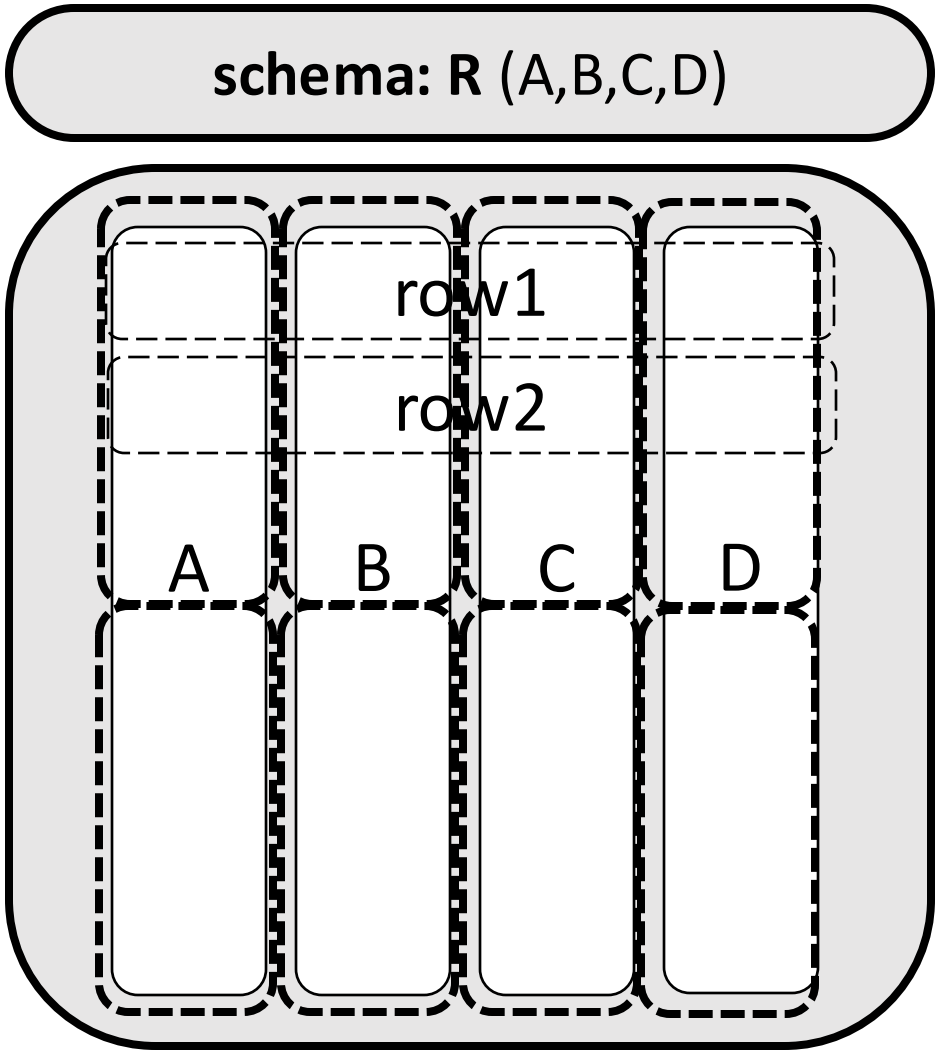
select A **from** R

any drawbacks?

each page contains **columns!**

column store

querying over slotted pages



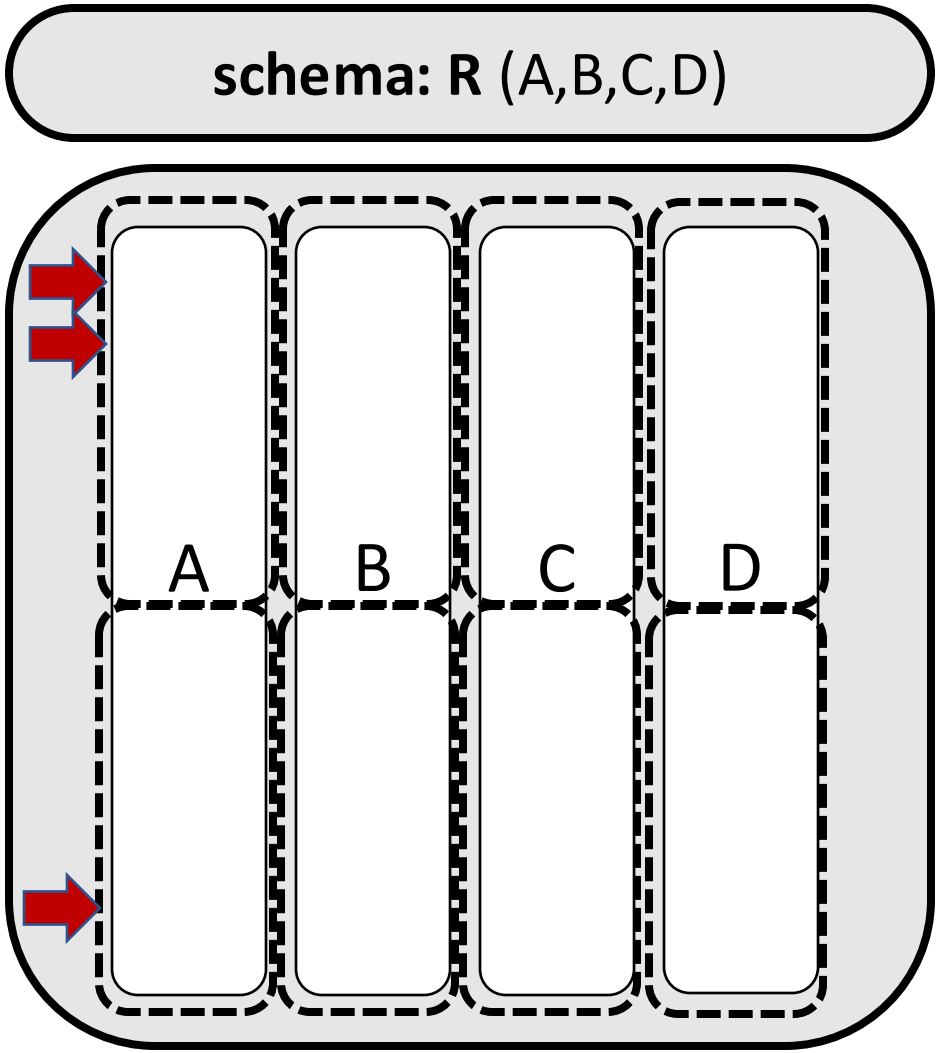
select A,B,C,D **from** R

select A **from** R

select (A+B) **from** R

each page contains **columns!**

querying over slotted pages



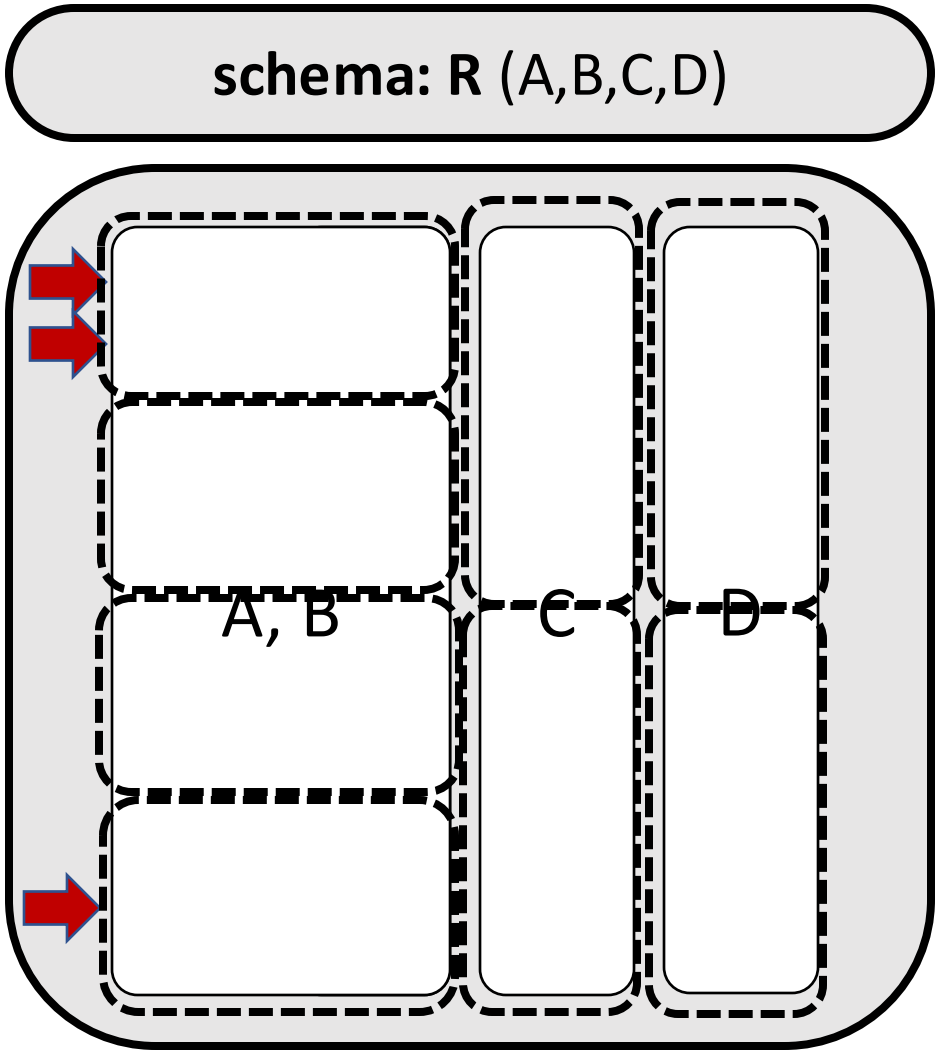
select A,B,C,D **from** R

select A **from** R

select (A+B) **from** R **where** A>10

each page contains **columns!**

querying over slotted pages



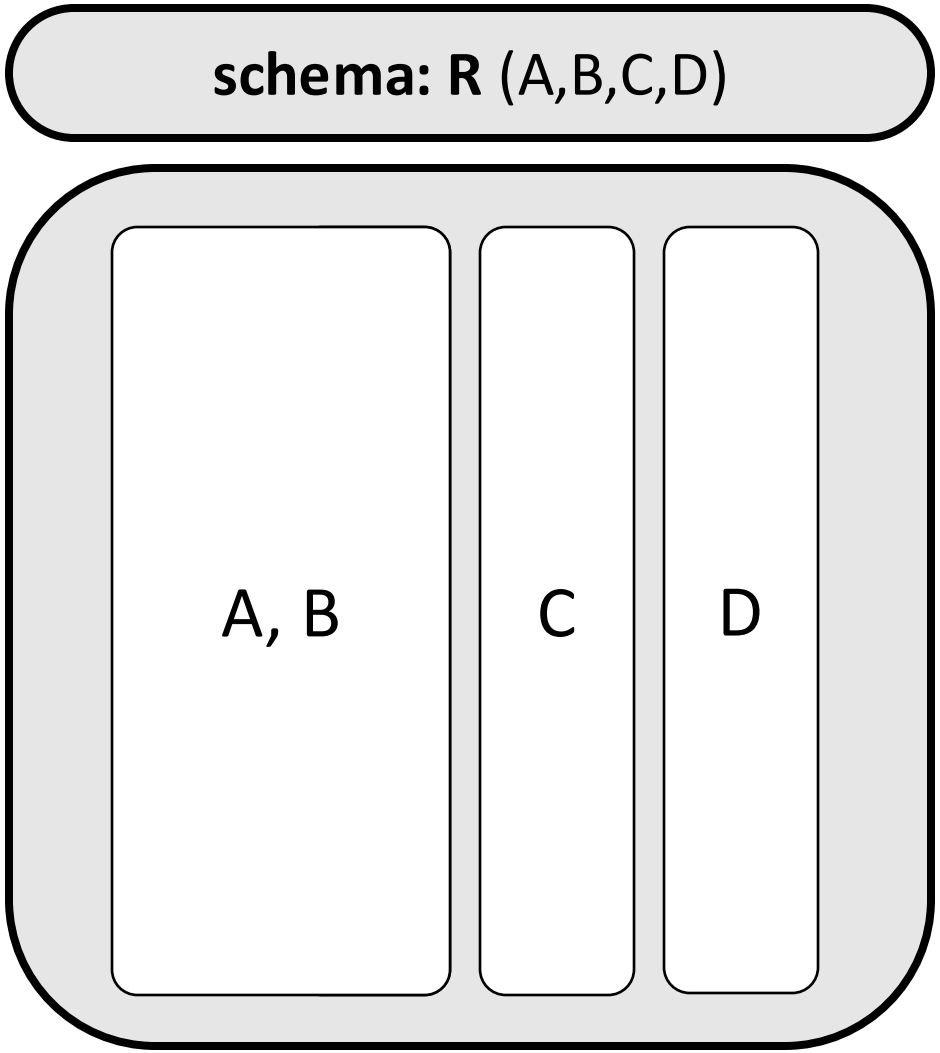
select A,B,C,D **from** R

select A **from** R

select (A+B) **from** R **where** A>10

each page contains **columns** or ***groups of columns***!

querying over slotted pages



select A,B,C,D **from** R

select A **from** R

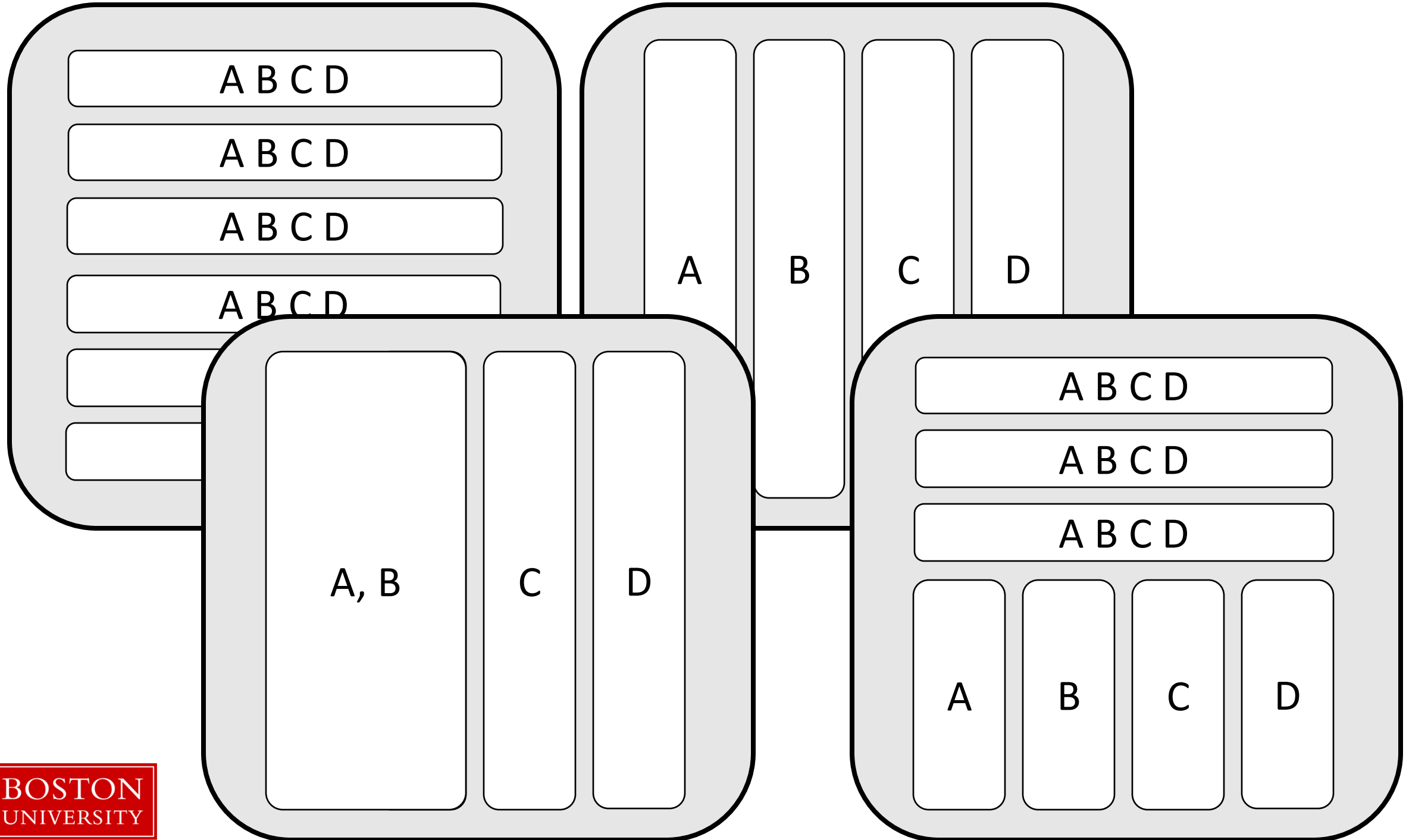
select (A+B) **from** R **where** A>10

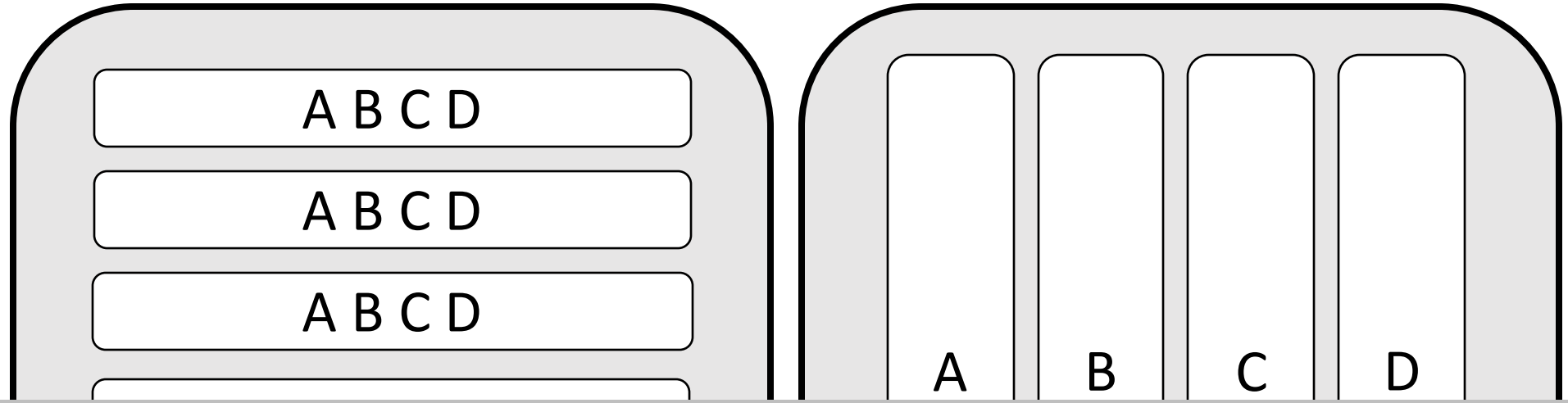
each page contains **columns** or ***groups of columns***!

what if I had all three queries?

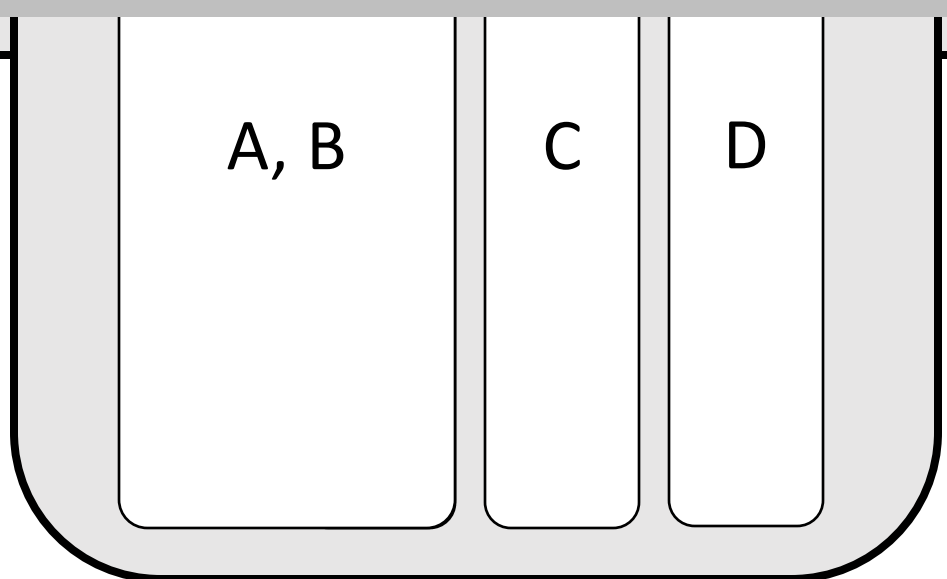
what if only inserts/updates?

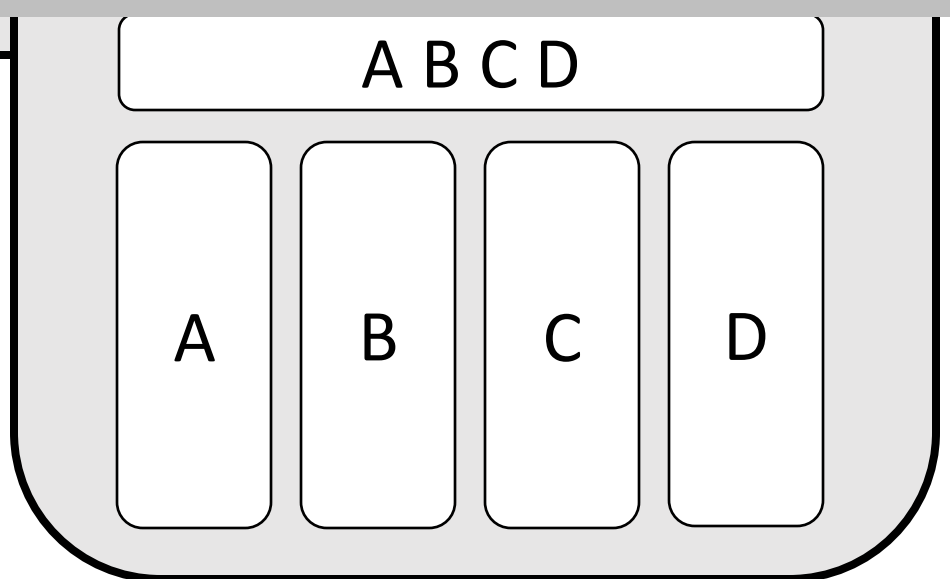
can there be something in between?



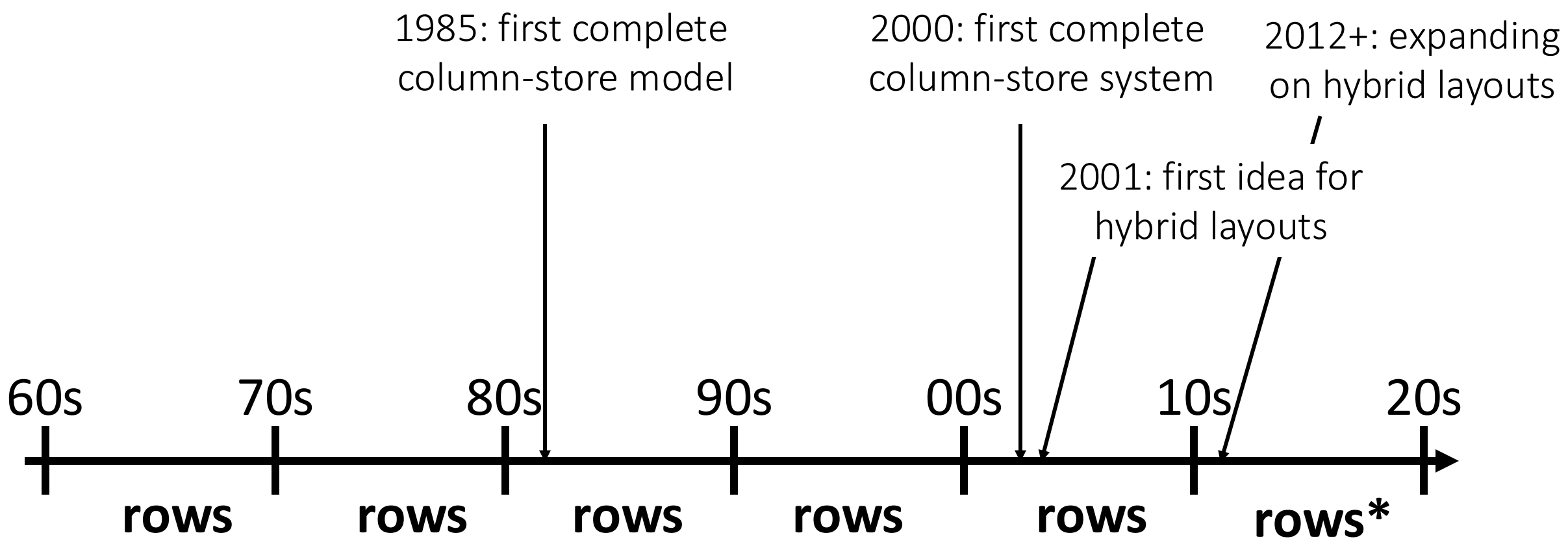


the way we physical store data dictates
what are the possible efficient access methods

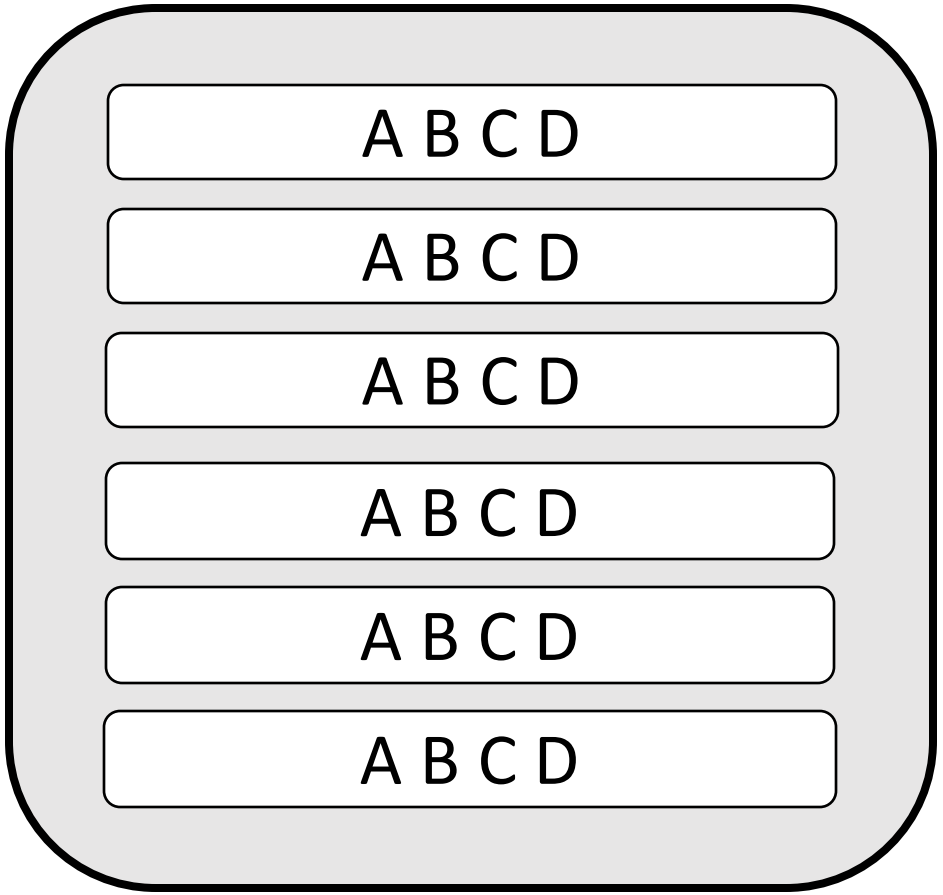




column-stores history line



query evaluation

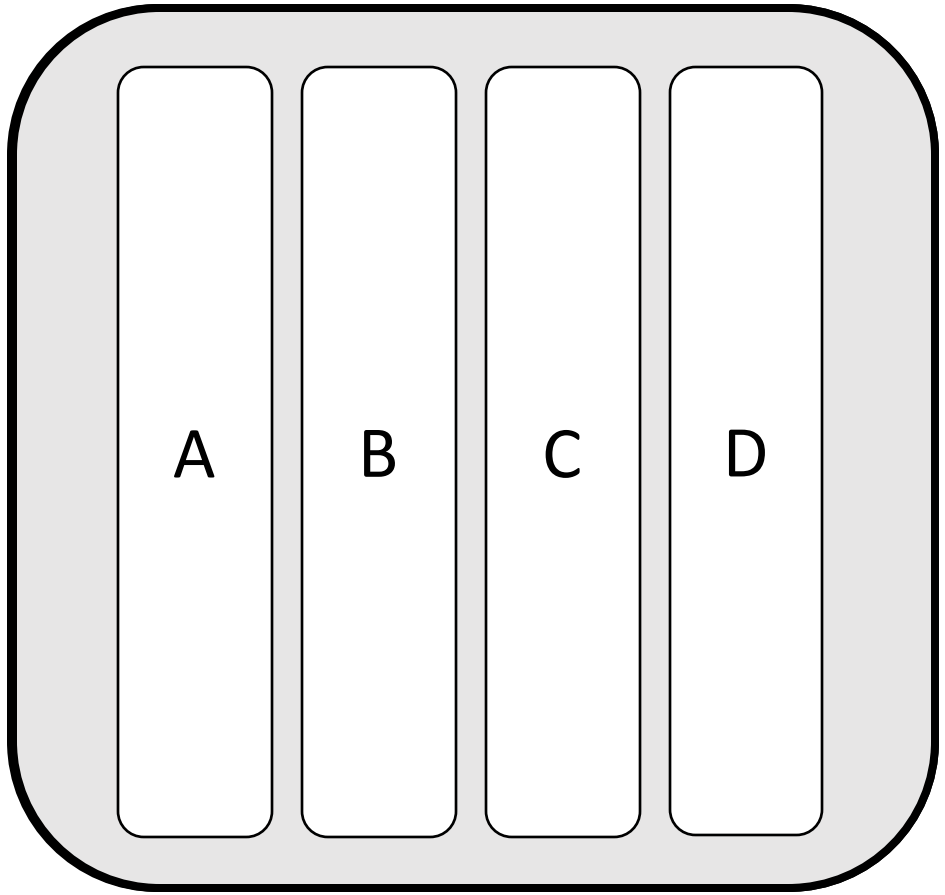


select max(B) **from** R **where** A>5 **and** C<10

tuple reconstruction/early materialization

A B C D

one row at a time

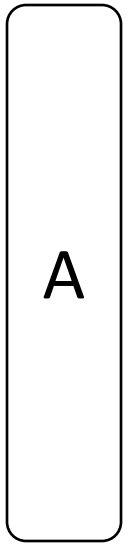


select max(B) **from** R **where** A>5 **and** C<10

tuple reconstruction/early materialization

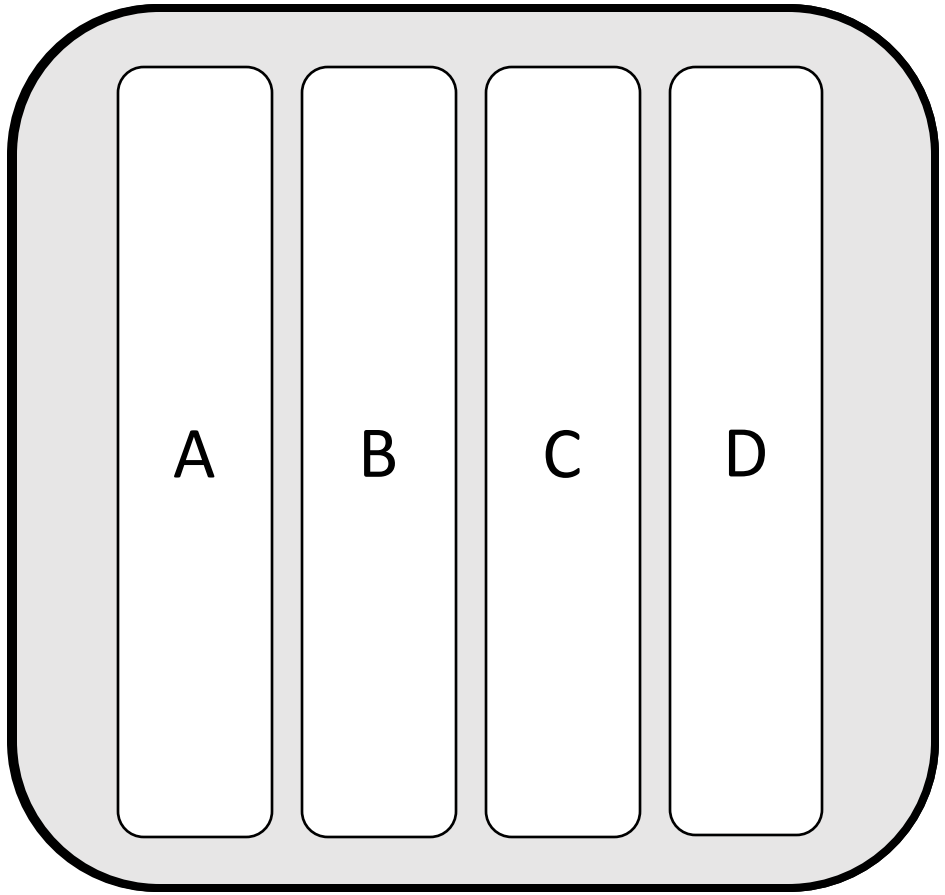
A B C D

one row at a time

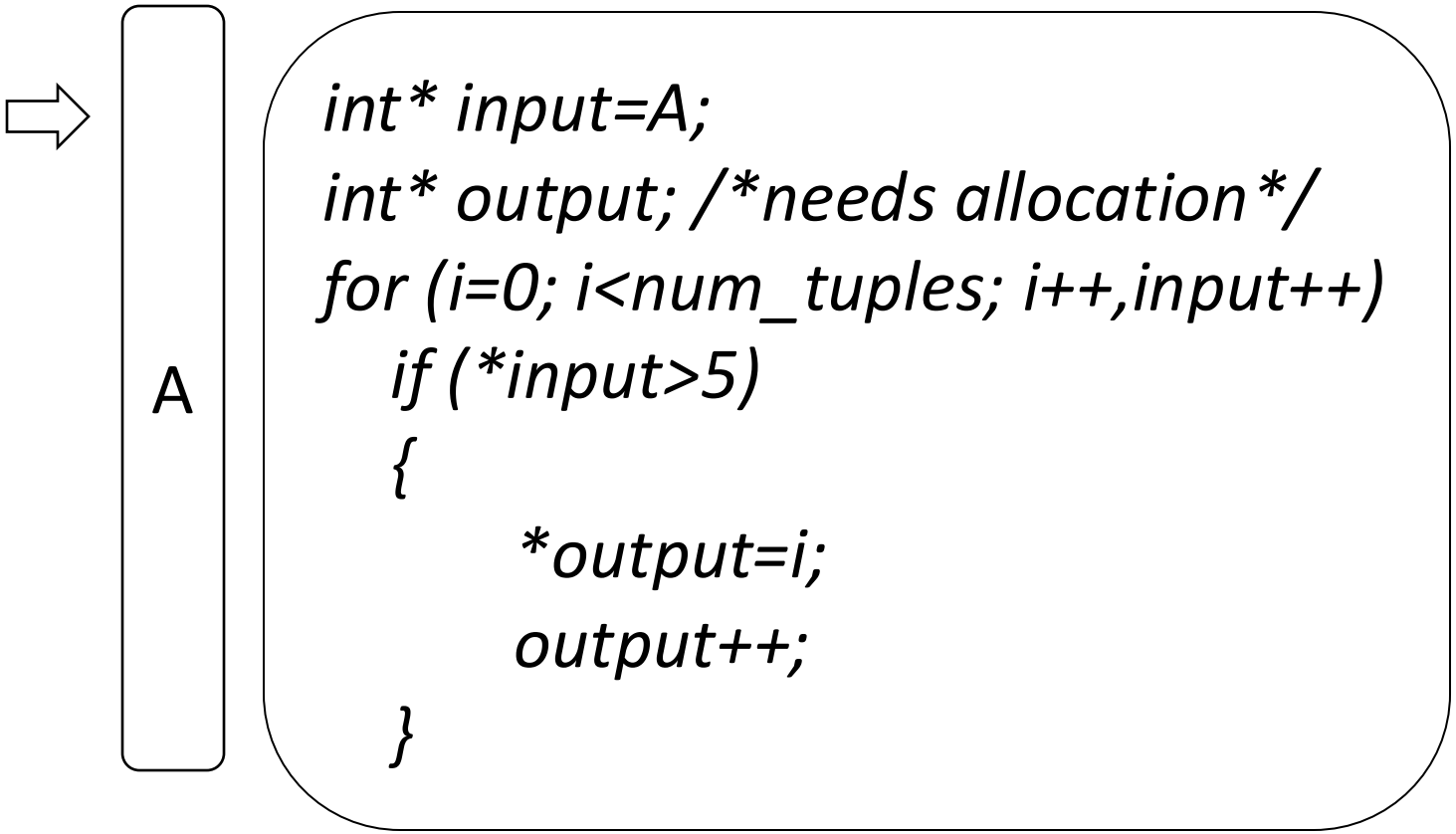


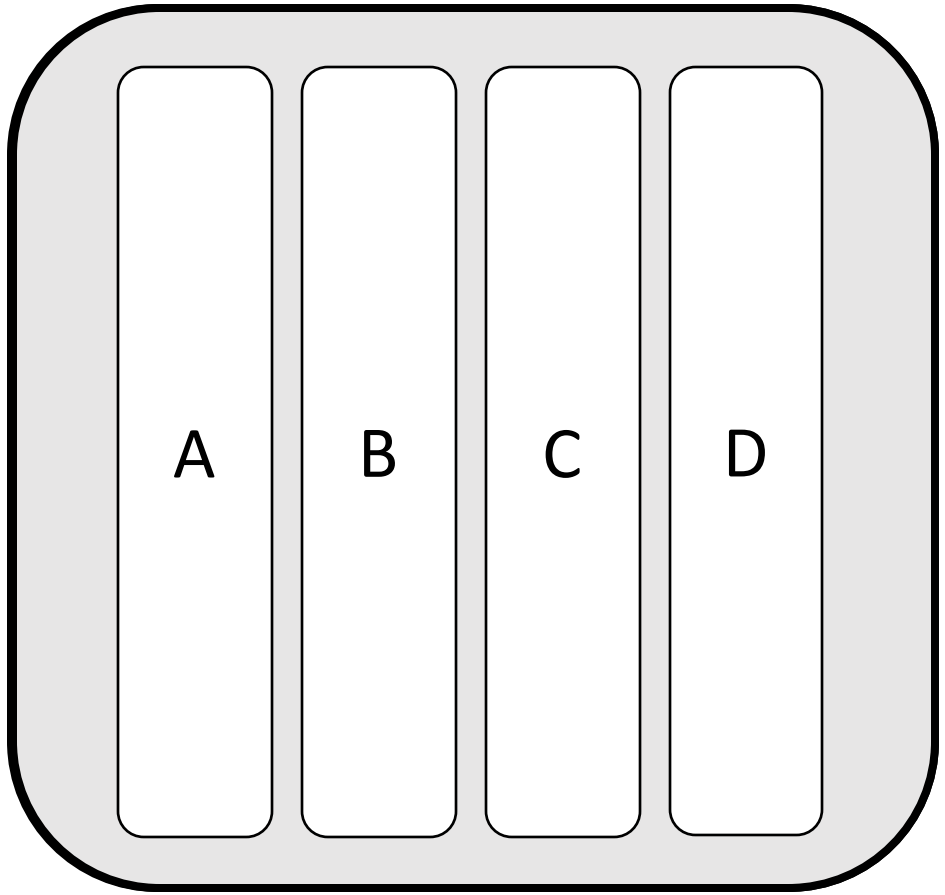
late materialization

column at a time

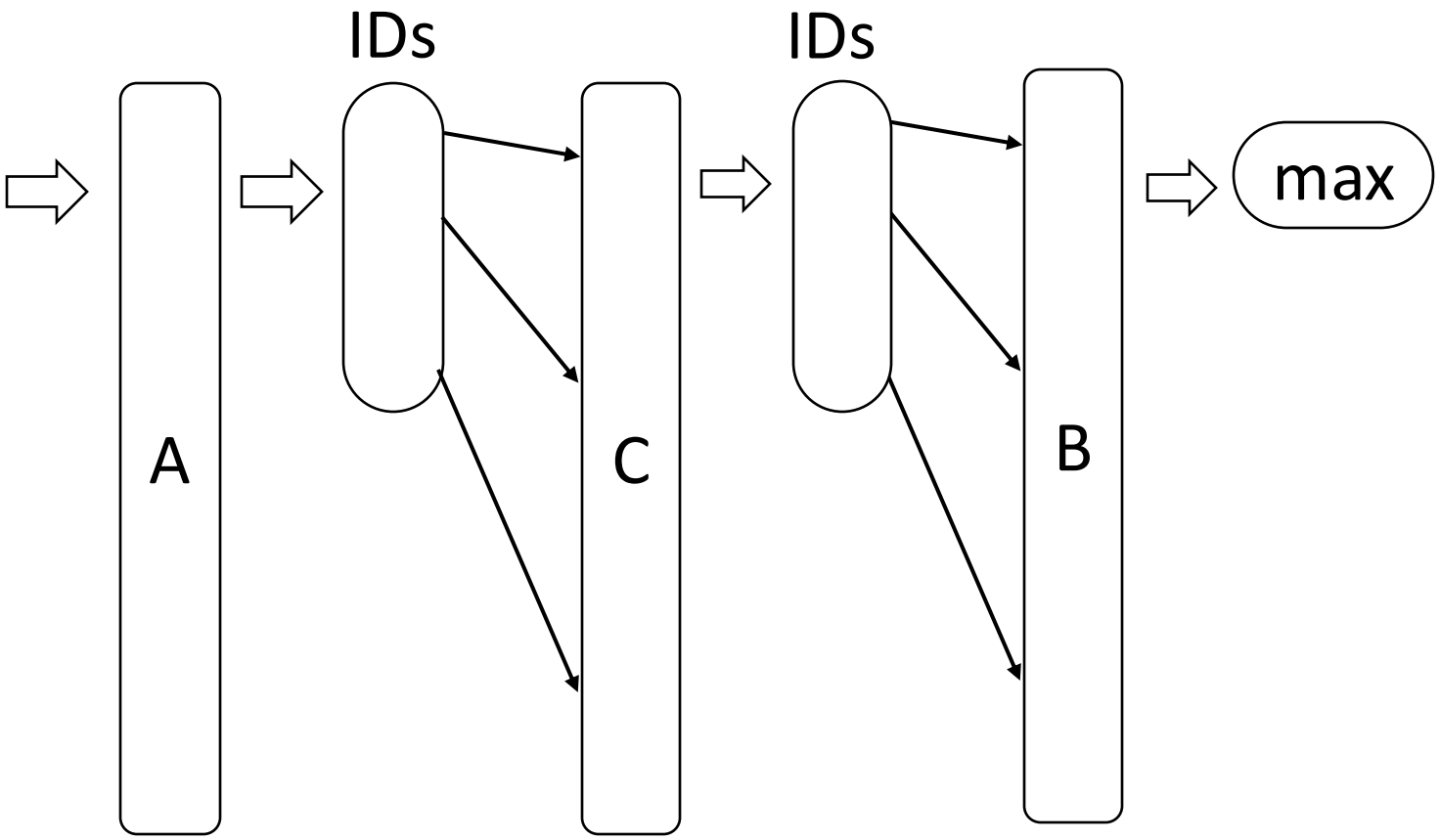


select max(B) **from** R **where** A>5 **and** C<10





select max(B) **from** R **where** A>5 **and** C<10



what is the benefit?

read only useful data

easy to code: working over fixed width and dense columns

scan

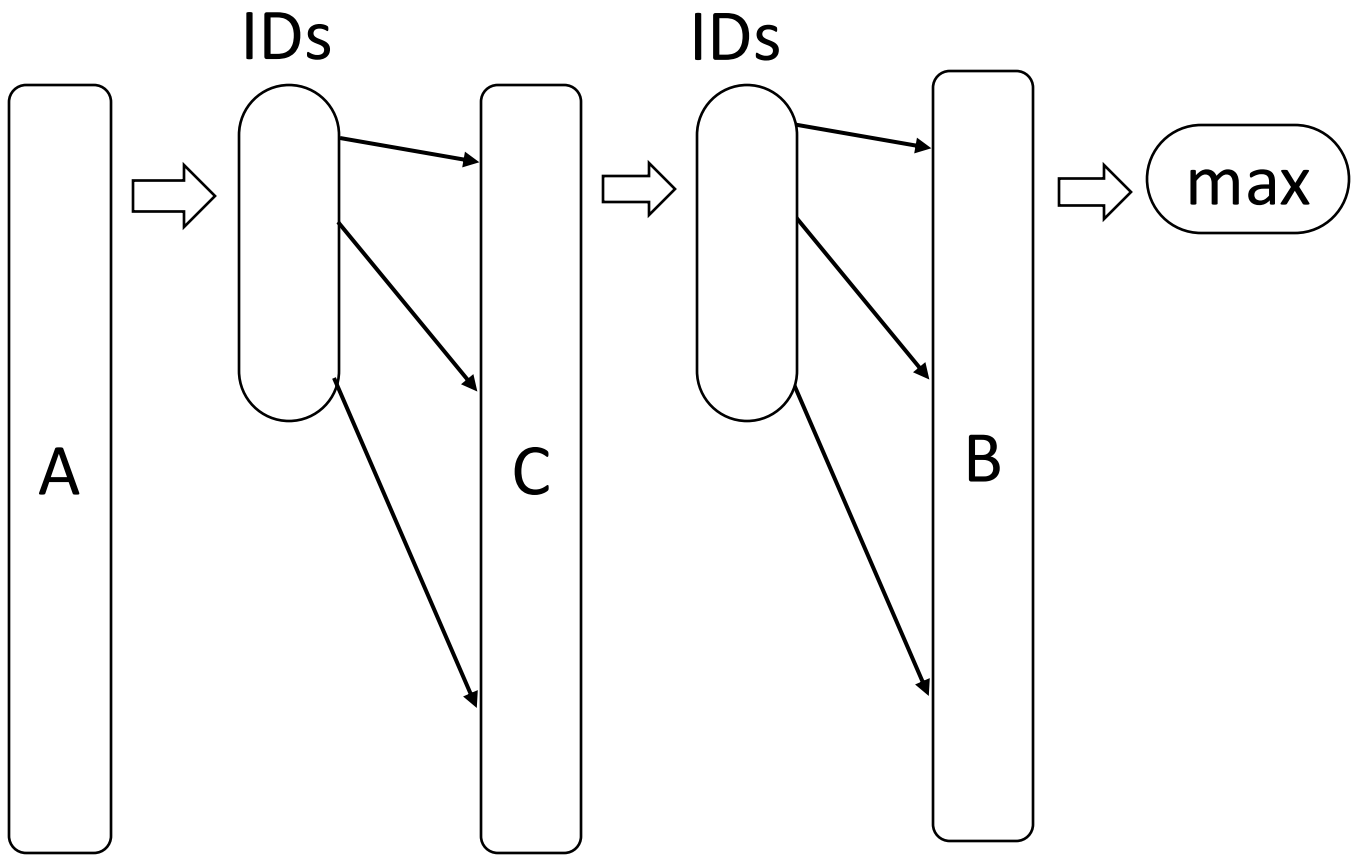
```
for (i=0,j=0; i<size; i++)
    if (column[i] qualifies)
        res[j++]=i;
```

no complex checks
no function calls
no aux metadata
easy to prefetch
as few ifs as possible

fetch

```
for (i=0,j=0; i<fetch_size; i++)
    intermediate_result[j++]=column[ids[i]];
```

select max(B) **from** R **where** A>5 **and** C<10

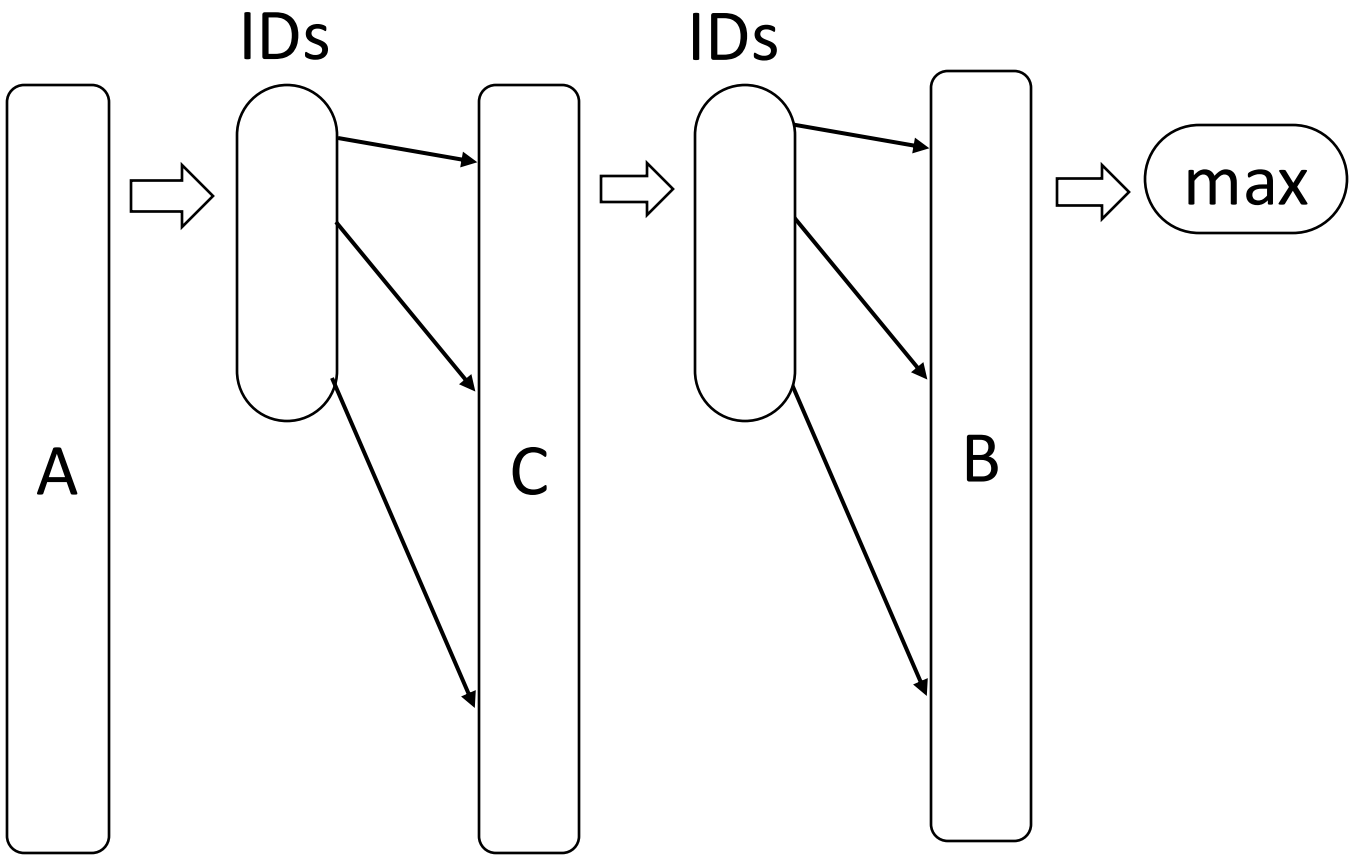


alternative query plans

start from C (why?)

scan A & C in parallel and merge

select max(B) **from** R **where** A>5 **and** C<10



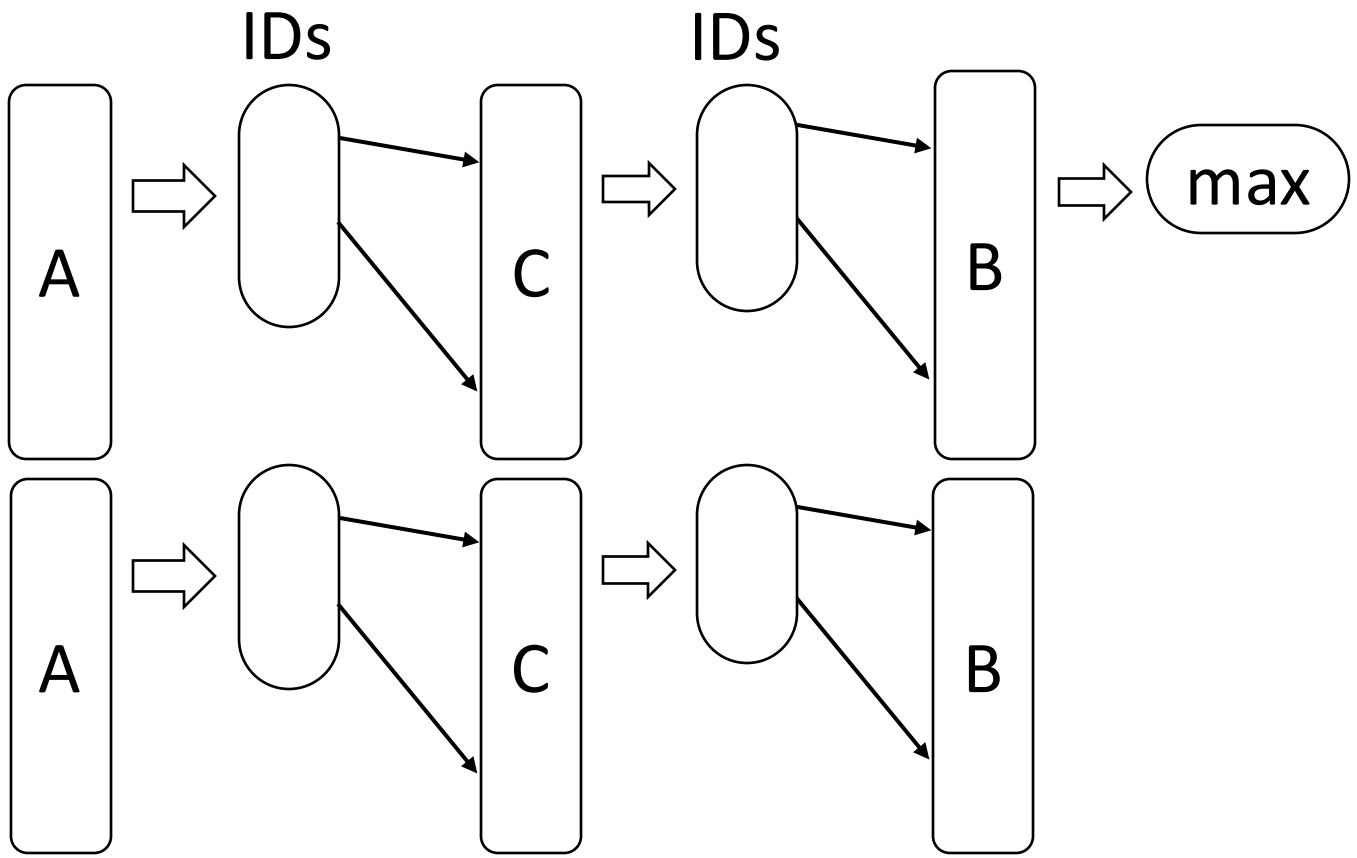
whole column?

row at a time

column at a time

block/vector at a time

select max(B) **from** R **where** A>5 **and** C<10



whole column?

row at a time

column at a time

block/vector at a time

why column-stores are here now?

late materialization – no need to reconstruct tuples

read only useful data

minimize data movement across the memory hierarchy

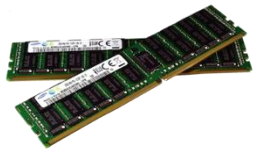
but it required a complete re-write

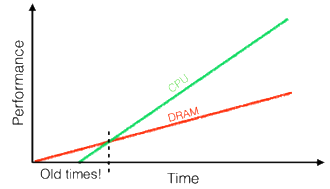
why not before?

legacy technology to catch up

more important: **analytical workloads** (as opposed to only OLTP)

new hardware: **larger memories & memory wall**





Project details are now online (more to come)

detailed discussion in next class

Readings **for the project**

The Log-Structured Merge-Tree (LSM-Tree) by Patrick E. O'Neil, Edward Cheng, Dieter Gawlick, Elizabeth J. O'Neil. Acta Inf. 33(4): 351-385, 1996

Monkey: Optimal Navigable Key-Value Store by Niv Dayan, Manos Athanassoulis, Stratos Idreos. SIGMOD Conference 2017

More readings *(for some research projects)*

Measures of Presortedness and Optimal Sorting Algorithms by Heikki Mannila. IEEE Trans. Computers 34(4): 318-325 (1985)

Small Materialized Aggregates: A Light Weight Index Structure for Data Warehousing by Guido Moerkotte. VLDB 1998

The adaptive radix tree: ARTful indexing for main-memory databases by Viktor Leis, Alfons Kemper, Thomas Neumann. ICDE 2013: 38-49

programming language: C/C++

it gives you **control over exactly** what is happening
it helps you **learn the impact** of design decisions

avoid using libraries unless asked to do,
so you can control storage and access patterns

BOSTON UNIVERSITY

CS 561: Data Systems Architectures

class 3

Relational Recap & Column-Stores Basics

Prof. Manos Athanassoulis

https://bu-disc.github.io/CS561/