CS 6530: Advanced Database Systems Fall 2022

Lecture 02
History of databases &
Data system architecture

Prashant Pandey

prashant.pandey@utah.edu

Some reminders…

no smartphones

no laptop

- Reading list for first two lectures is posted in Canvas
- Pop quiz #0 is posted
- Join online discussion through Piazza

https://piazza.com/utah/fall2022/cs6530001fall2022

A brief history of databases

Acknowledgement: Slides taken from Prof. Andy Pavlo, CMU

History repeats itself

- Old database issues are still relevant today.
- The **SQL vs. NoSQL** debate is reminiscent of **Relational vs. CODASYL** debate from the 1970s.
  - Spoiler: The relational model almost always wins.
- Many of the ideas in today's database systems are not new.

Acknowledgement: Prof. Andy Pavlo, CMU

1960s – IDS

- **I**ntegrated **D**ata **S**tore
- Developed internally at GE in the early 1960s.
- GE sold their computing division to Honeywell in 1969.
- One of the first DBMSs:
  - Network data model.
  - Tuple-at-a-time queries.

Acknowledgement: Prof. Andy Pavlo, CMU

1960s – CODASYL

- COBOL people got together and proposed a standard for how programs will access a database. Lead by Charles Bachman.
  - Network data model.
  - Tuple-at-a-time queries.

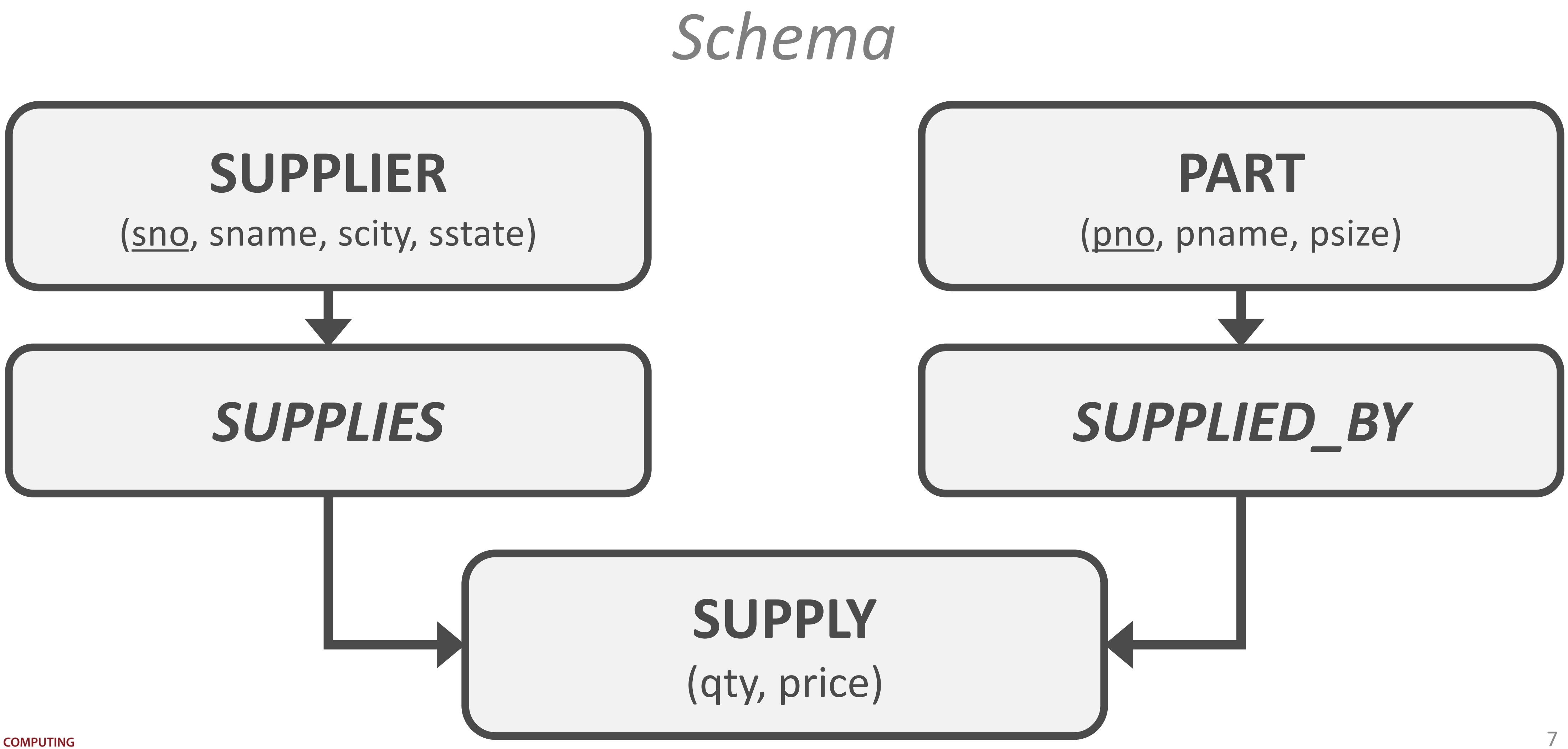
Bachman

- Bachman also worked at Culliane Database Systems in the 1970s to help build **IDMS**.

Turing award 1973

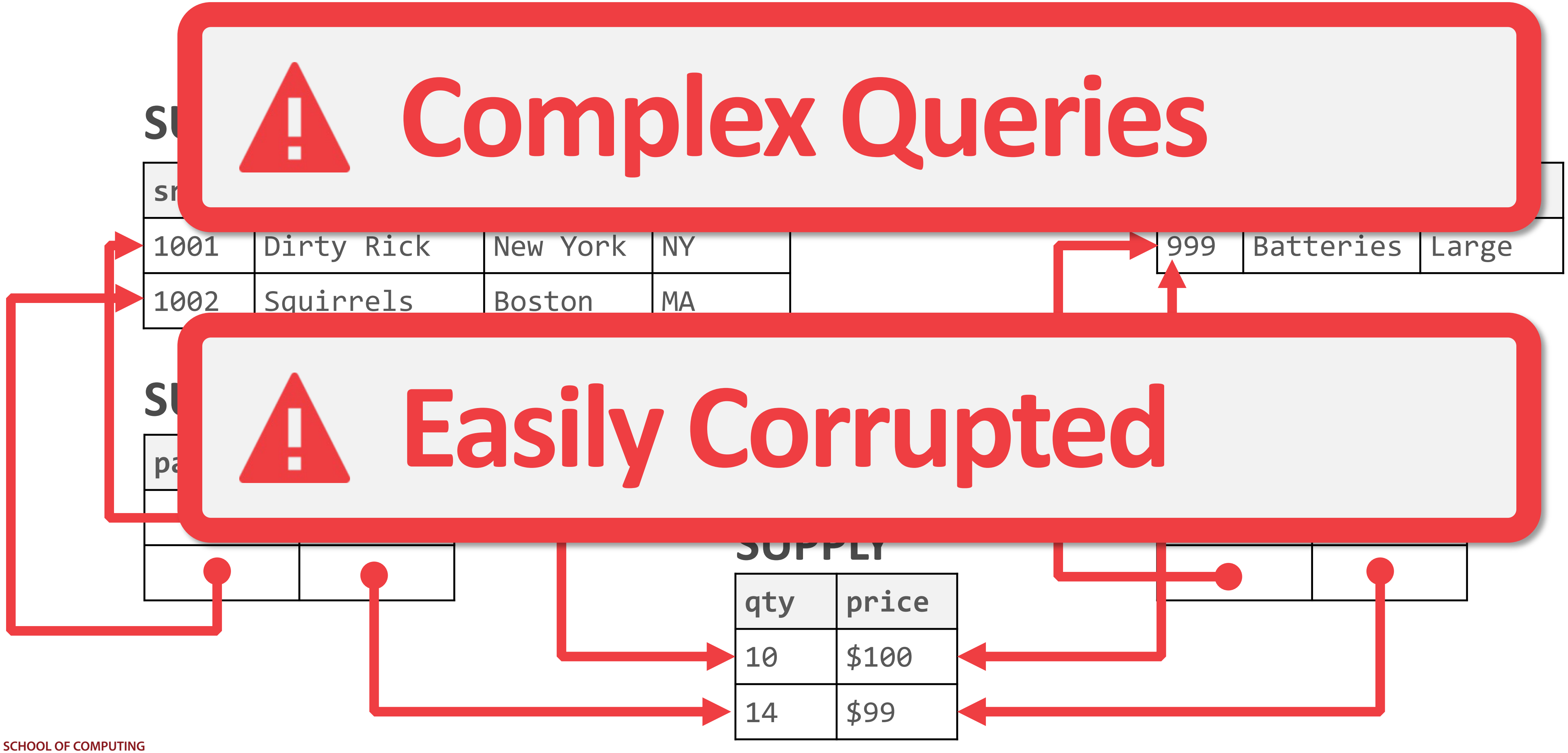
Acknowledgement: Prof. Andy Pavlo, CMU

Network data model



Acknowledgement: Prof. Andy Pavlo, CMU

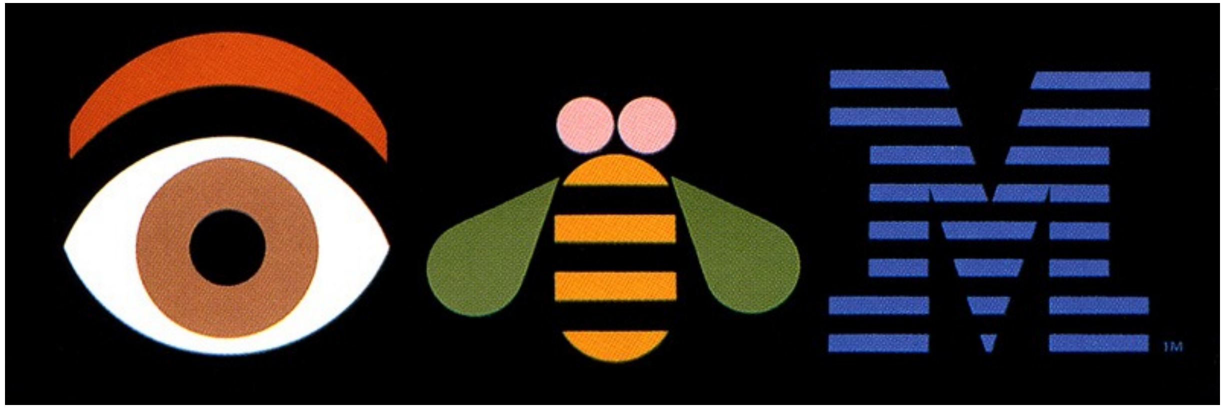
Network data model



Acknowledgement: Prof. Andy Pavlo, CMU

1960S – IBM IMS

- **I**nformation **M**anagement **S**ystem
- Early database system developed to keep track of purchase orders for Apollo moon mission.
  - Hierarchical data model.
  - Programmer-defined physical storage format.
  - Tuple-at-a-time queries.



Acknowledgement: Prof. Andy Pavlo, CMU

hierarchical data model

⚠ Duplicate Data

⚠ No Independence

(sno, sname, scity, sstate)

				parts
1002	Squirrels	Boston	MA	

price
$100

PART

(pno, pname, psize, qty, price)

pno	pname	psize	qty	price
999	Batteries	Large	14	$99

SCHOOL OF COMPUTING
UNIVERSITY OF UTAH

Acknowledgement: Prof. Andy Pavlo, CMU

MOD

matician
w devel
riting II
time th
yout ch
avoid
ata stru
evel lar
mplem

DERIVABILITY, REDUNDANCY AND CONSISTENCY OF RELATIONS
STORED IN LARGE DATA BANKS

E. F. Codd
Research Division
San Jose, California

ABSTRACT: The large, integrated data banks of the future will contain many relations of various degrees in stored form. It will not be unusual for this set of stored relations to be redundant. Two types of redundancy are defined and discussed. One type may be employed to improve accessibility of certain kinds of information which happen to be in great demand. When either type of redundancy exists, those responsible for control of the data bank should know about it and have some means of detecting any "logical" inconsistencies in the total set of stored relations. Consistency checking might be helpful in tracking down unauthorized (and possibly fraudulent) changes in the data bank contents.

RJ 599(# 12343) August 19, 1969

P. BAXENDALE, Editor

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on *n*-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity
CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

1. Relational Model and Normal Form

1.1. Introduction

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question-answering systems. Levein and Maron [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of *data independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive systems.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. Data Dependencies in Present Systems

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed *without logically impairing some application programs* is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In some systems these dependencies are not clearly separable from one another.

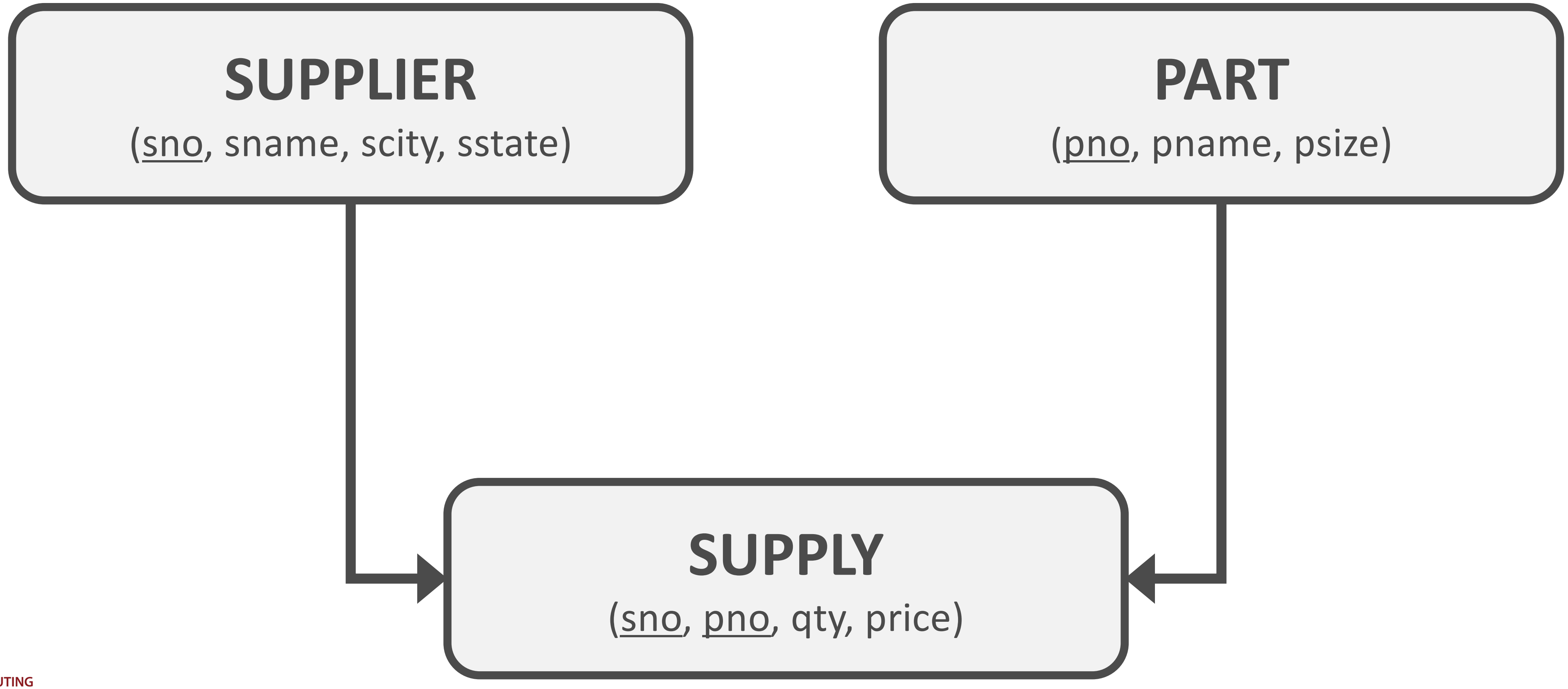
1.2.1. *Ordering Dependence.* Elements of data in a data bank may be stored in a variety of ways, some involving no concern for ordering, some permitting each element to participate in one ordering only, others permitting each element to participate in several orderings. Let us consider those existing systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses. For example, the records of a file concerning parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

Turing award 1981

Acknowledgement: Prof. Andy Pavlo, CMU

Relational data model

Schema



Acknowledgement: Prof. Andy Pavlo, CMU

Relational data model

Instance

SUPPLIER

sno	sname	scity	sstate
1001	Dirty Rick	New York	NY
1002	Squirrels	Boston	MA

PART

pno	pname	psize
999	Batteries	Large

SUPPLY

sno	pno	qty	price
1001	999	10	$100
1002	999	14	$99

Acknowledgement: Prof. Andy Pavlo, CMU

1970s – Relational model

- Early implementations of relational DBMS:
  - **System R** – IBM Research
  - **INGRES** – U.C. Berkeley
  - **Oracle** – Larry Ellison

Turing award 1998

Gray

Turing award 2015

Stonebraker

Ellison

Acknowledgement: Prof. Andy Pavlo, CMU

1980s – Relational model

- The relational model wins.
  - IBM comes out with DB2 in 1983.
  - "SEQUEL" becomes the standard (SQL).
- Many new "enterprise" DBMSs but Oracle wins marketplace.
- Stonebraker creates Postgres.

Acknowledgement: Prof. Andy Pavlo, CMU

1980s – Object-oriented databases

- Avoid “relational-object impedance mismatch” by tightly coupling objects and database.
- Few of these original DBMSs from the 1980s still exist today but many of the technologies exist in other forms (JSON, XML)

Acknowledgement: Prof. Andy Pavlo, CMU

Object-oriented model

⚠ Complex Queries

```
String email;
String phone[];
}
```

"email": "ante@up.com",

⚠ No Standard API

id		
1001	M.O.P.	ante@up.com

(sid, phone)

sid	phone
1001	444-444-4444
1001	555-555-5555

Acknowledgement: Prof. Andy Pavlo, CMU

1990s – Boring days

- No major advancements in database systems or application workloads.
  - Microsoft forks Sybase and creates SQL Server.
  - MySQL is written as a replacement for mSQL.
  - Postgres gets SQL support.
  - SQLite started in early 2000.

Acknowledgement: Prof. Andy Pavlo, CMU

2000s – Internet boom

- All the big players were heavyweight and expensive. Open-source databases were missing important features.

- Many companies wrote their own custom middleware to scale out database across single-node DBMS instances.

Acknowledgement: Prof. Andy Pavlo, CMU

2000s – Data warehouses

- Rise of the special purpose OLAP DBMSs.
  - Distributed / Shared-Nothing
  - Relational / SQL
  - Usually closed-source.
- Significant performance benefits from using **columnar data storage** model.

Acknowledgement: Prof. Andy Pavlo, CMU

2000s – NoSQL Systems

- Focus on high-availability & high-scalability:
  - Schemaless (i.e., "Schema Last")
  - Non-relational data models (document, key/value, etc)
  - No ACID transactions
  - Custom APIs instead of SQL
  - Usually open-source

Acknowledgement: Prof. Andy Pavlo, CMU

2010s – NewSQL

- Provide same performance for OLTP workloads as NoSQL DBMSs without giving up ACID:
  - Relational / SQL
  - Distributed
  - Usually closed-source

dbShards H-Store Clustrix Cockroach LABS Comdb2
ScaleArc SAP HANA VOLTDB NUODB TiDB
Google Spanner MEMSQL HyPer YugaByte

Acknowledgement: Prof. Andy Pavlo, CMU

2010s – Hybrid systems

- **<u>H</u>**ybrid **<u>T</u>**ransactional-**<u>A</u>**nalytical **<u>P</u>**rocessing.
- Execute fast OLTP like a NewSQL system while also executing complex OLAP queries like a data warehouse system.
  - Distributed / Shared-Nothing
  - Relational / SQL
  - Mixed open/closed-source.

MEMSQL HyPer SNAPPY SAP HANA JustOne DB splice MACHINE Peloton

Acknowledgement: Prof. Andy Pavlo, CMU

2010s – Cloud systems

- First database-as-a-service (DBaaS) offerings were "containerized" versions of existing DBMSs.
- There are new DBMSs that are designed from scratch explicitly for running in a cloud environment.

Acknowledgement: Prof. Andy Pavlo, CMU

2010s – Shared-disk engines

- Instead of writing a custom storage manager, the DBMS leverages distributed storage.
  - Scale execution layer independently of storage.
  - Favors log-structured approaches.

- This is what most people think of when they talk about a **data lake**.

Acknowledgement: Prof. Andy Pavlo, CMU

2010s – Stream processing

- Execute continuous queries on streams of tuples.
- Extend processing semantics to include notion of windows.

- Often used in combination of batch-oriented systems in a **<u>lambda architecture</u>** deployment.

Acknowledgement: Prof. Andy Pavlo, CMU

2010s – Graph systems

- Systems for storing and querying graph data.
- Their main advantage over other data models is to provide a graph-centric query API
  - Recent research demonstrated that is unclear whether there is any benefit to using a graph-centric execution engine and storage manager.

Acknowledgement: Prof. Andy Pavlo, CMU

2010s – Timeseries systems

- Specialized systems that are designed to store timeseries / event data.
- The design of these systems make deep assumptions about the distribution of data and workload query patterns.

Acknowledgement: Prof. Andy Pavlo, CMU

2010s – SPECIALIZED SYSTEMS

- Embedded DBMSs
- Multi-Model DBMSs
- Blockchain DBMSs
- Hardware Acceleration

Parting thoughts

- The demarcation lines of DBMS categories will continue to blur over time as specialized systems expand the scope of their domains.

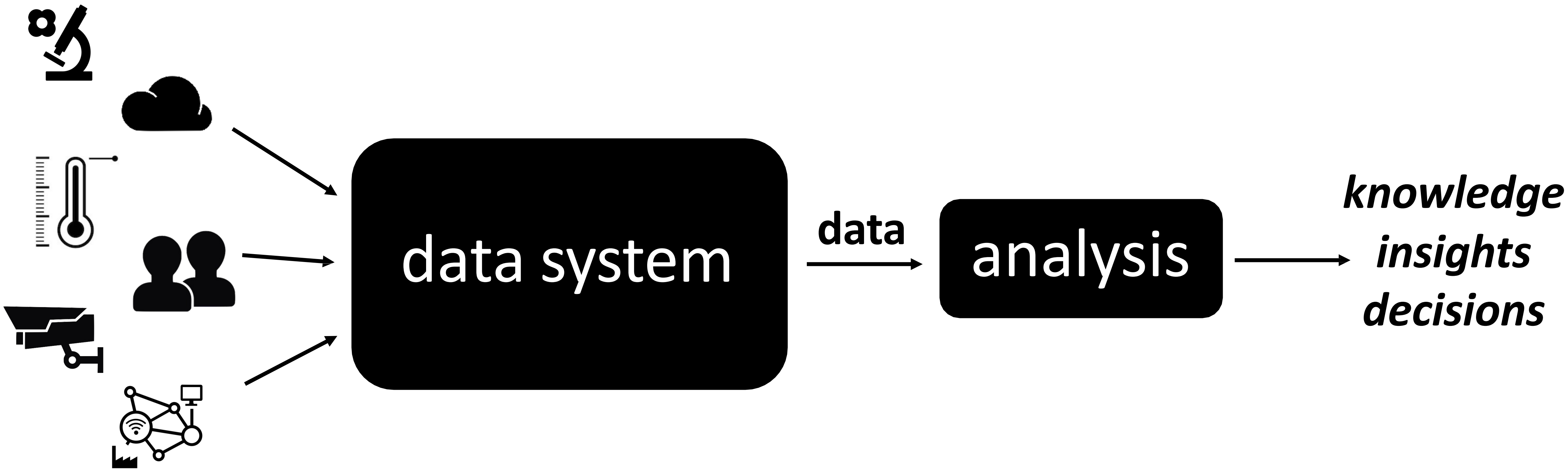
- I believe that the relational model and declarative query languages promote better data engineering.

Acknowledgement: Prof. Andy Pavlo, CMU

Data system architecture essentials

Acknowledgement: Slides taken from Prof. Manos Athanassoulis, BU

A **data system** is a large software system that **stores data**, and provides the **interface** to **update** and **access** them **efficiently**

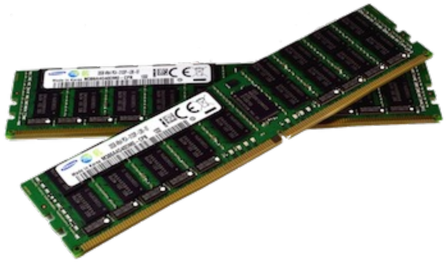


Acknowledgement: Prof. Manos Athanassoulis, BU

Growing need for tailored systems

new applications

new hardware

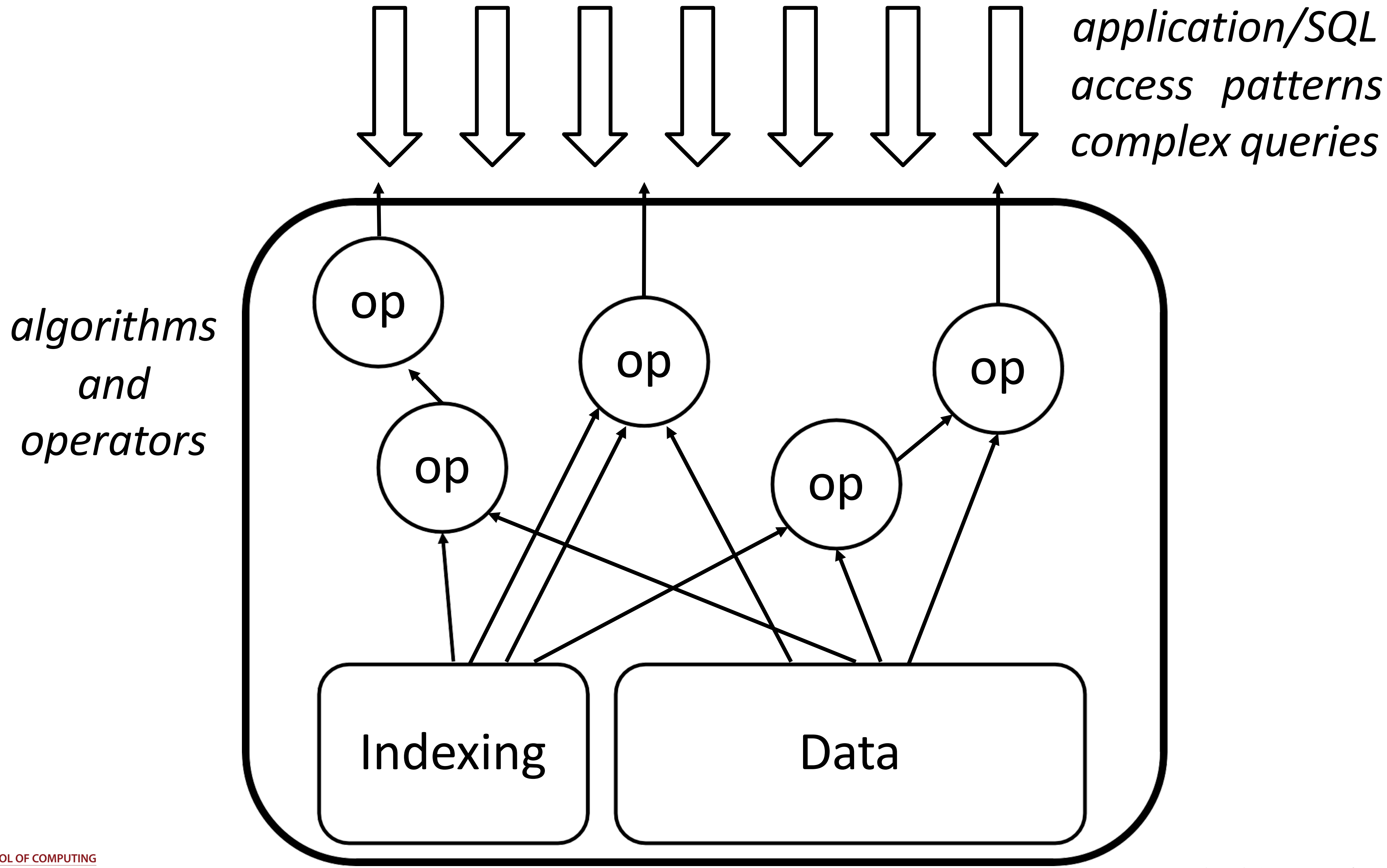


more data

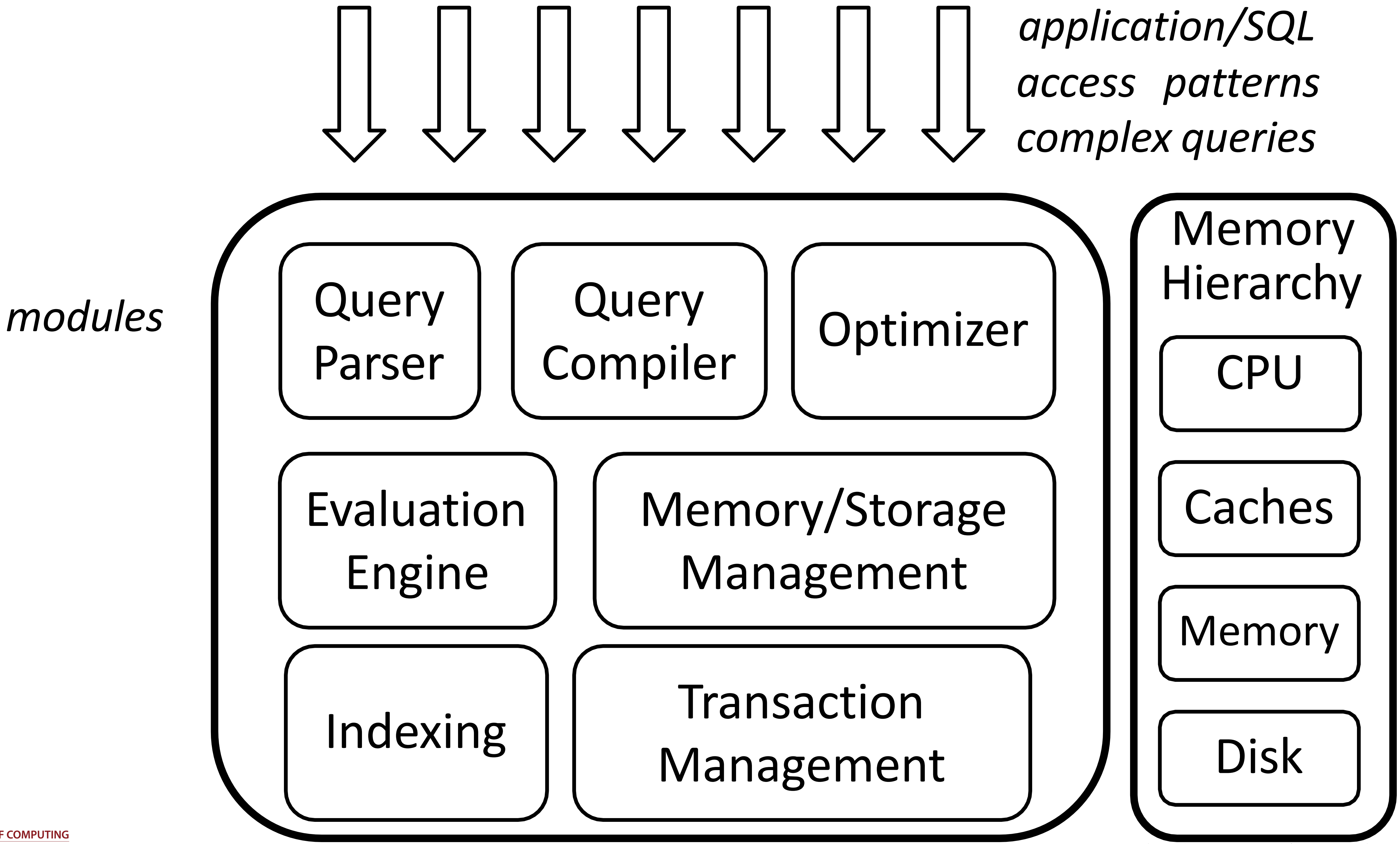
Acknowledgement: Prof. Manos Athanassoulis, BU

Data system, what's inside?

Acknowledgement: Prof. Manos Athanassoulis, BU



Acknowledgement: Prof. Manos Athanassoulis, BU

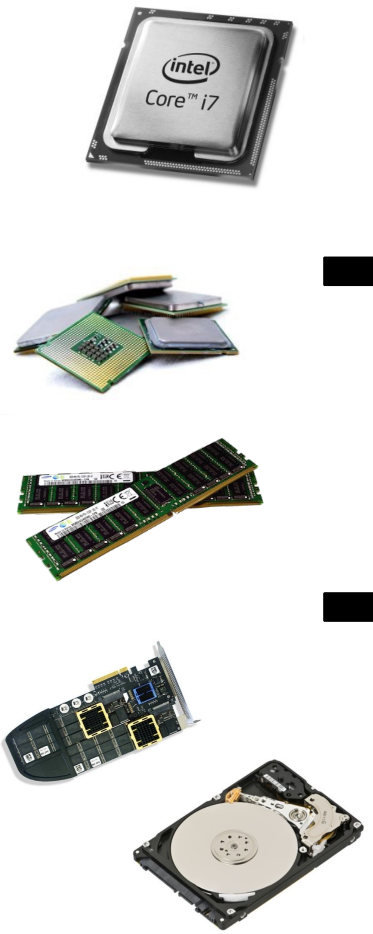


Acknowledgement: Prof. Manos Athanassoulis, BU

Data system, what's underneath?

Acknowledgement: Prof. Manos Athanassoulis, BU

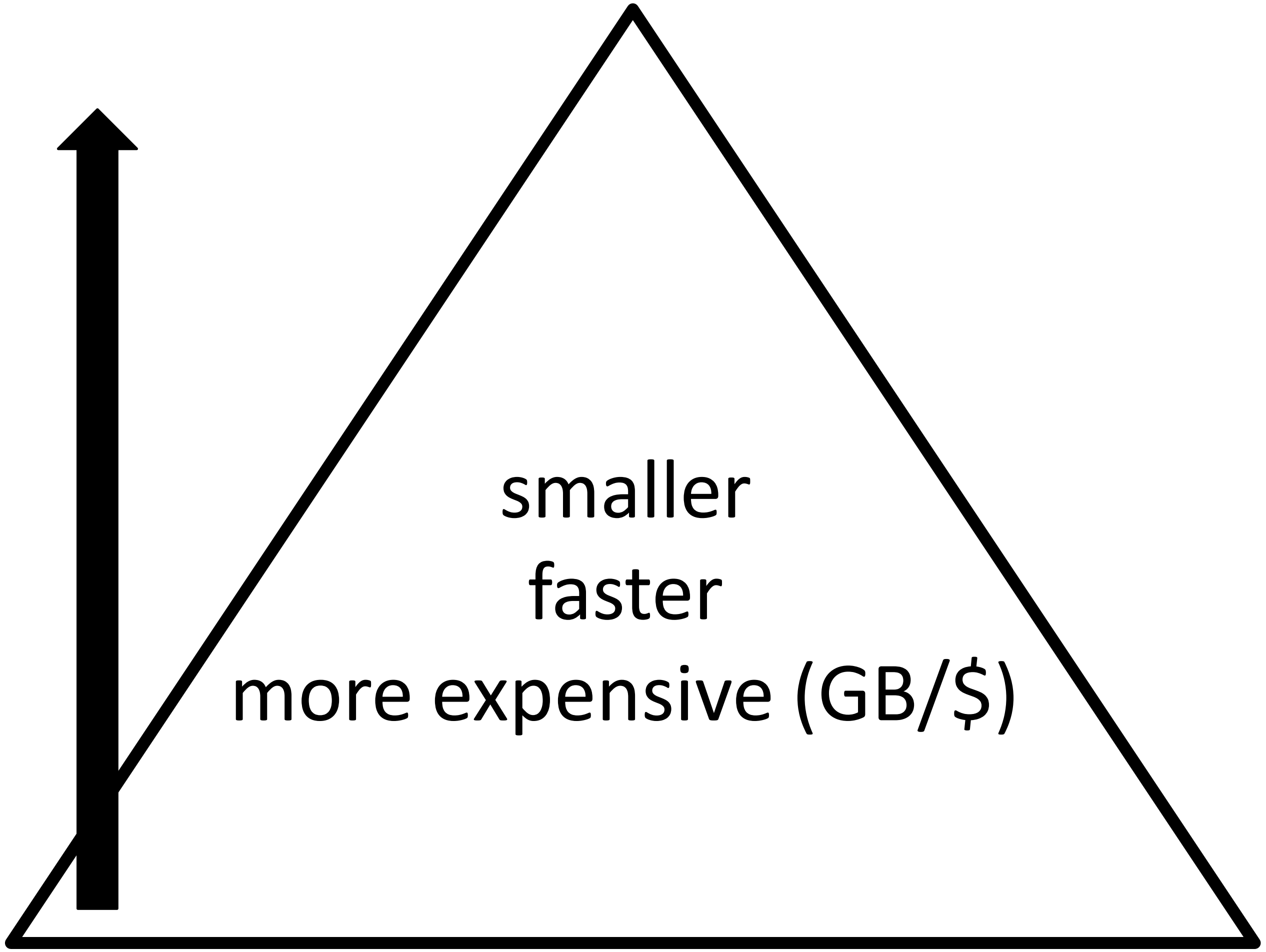
Memory hierarchy



- CPU
- on-chip cache

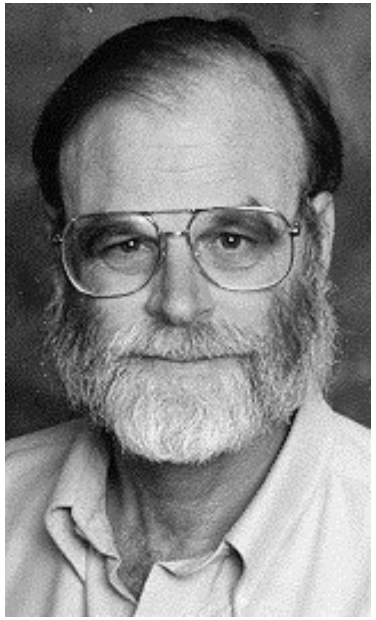
- on-board cache
- main memory

- flash storage
- disks | flash



Acknowledgement: Prof. Manos Athanassoulis, BU

Memory hierarchy (by Jim Gray)

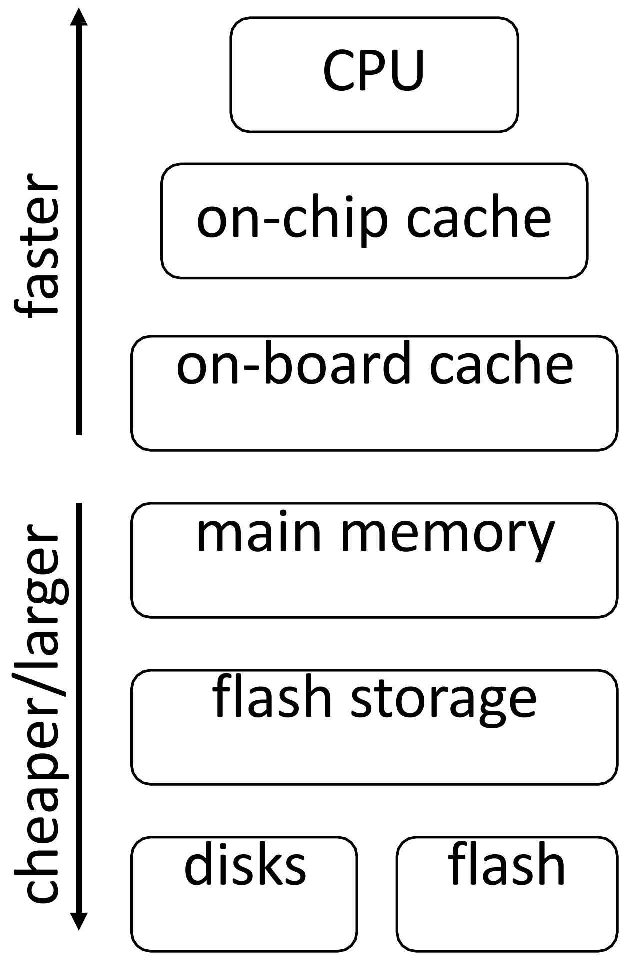


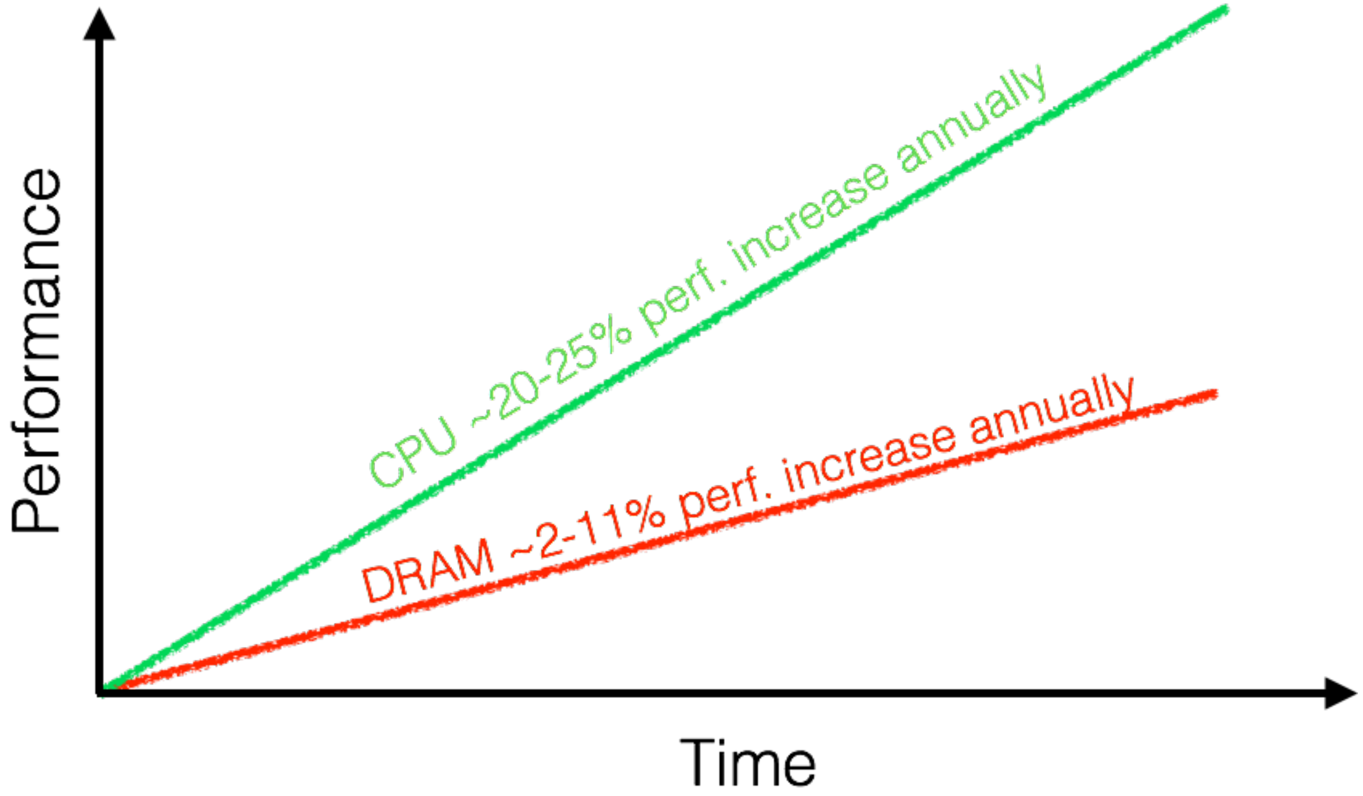
Jim Gray, IBM, Tandem, Microsoft, DEC
“The Fourth Paradigm” is based on his vision
ACM Turing Award 1998
ACM SIGMOD Edgar F. Codd Innovations award 1993

	registers/CPU	my head ~0
2x	**on chip cache**	this room 1min
10x	**on board cache**	this building 10min
100x	**memory**	Washington, DC 5 hours
10^6x	**disk**	Pluto 2 years
10^9x	**tape**	Andromeda 2000 years

Acknowledgement: Prof. Manos Athanassoulis, BU

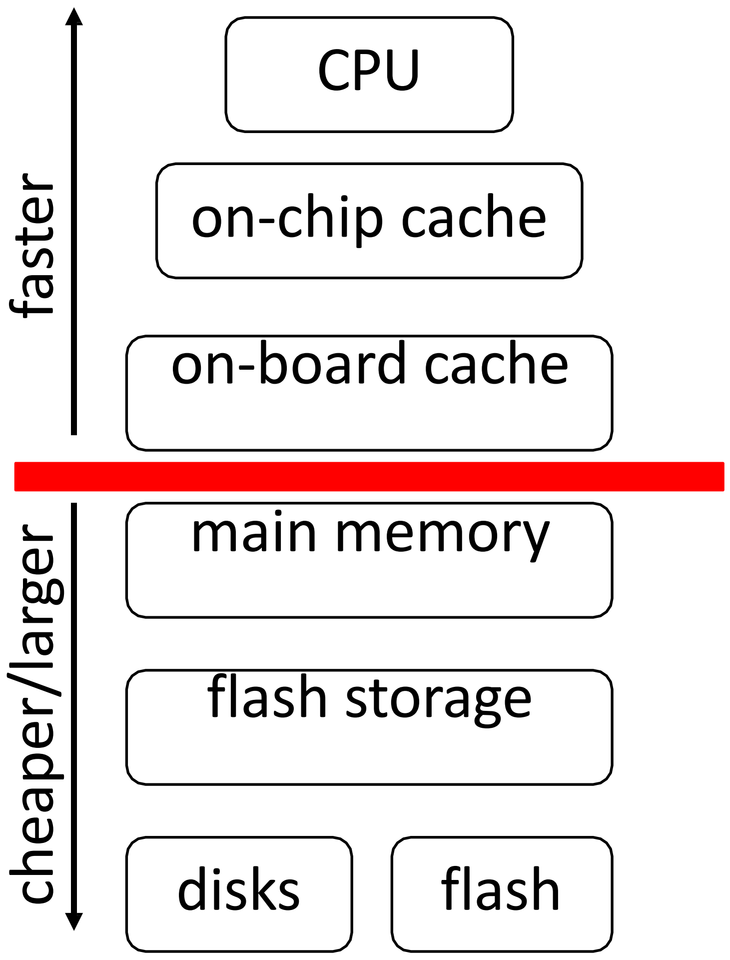
Memory wall

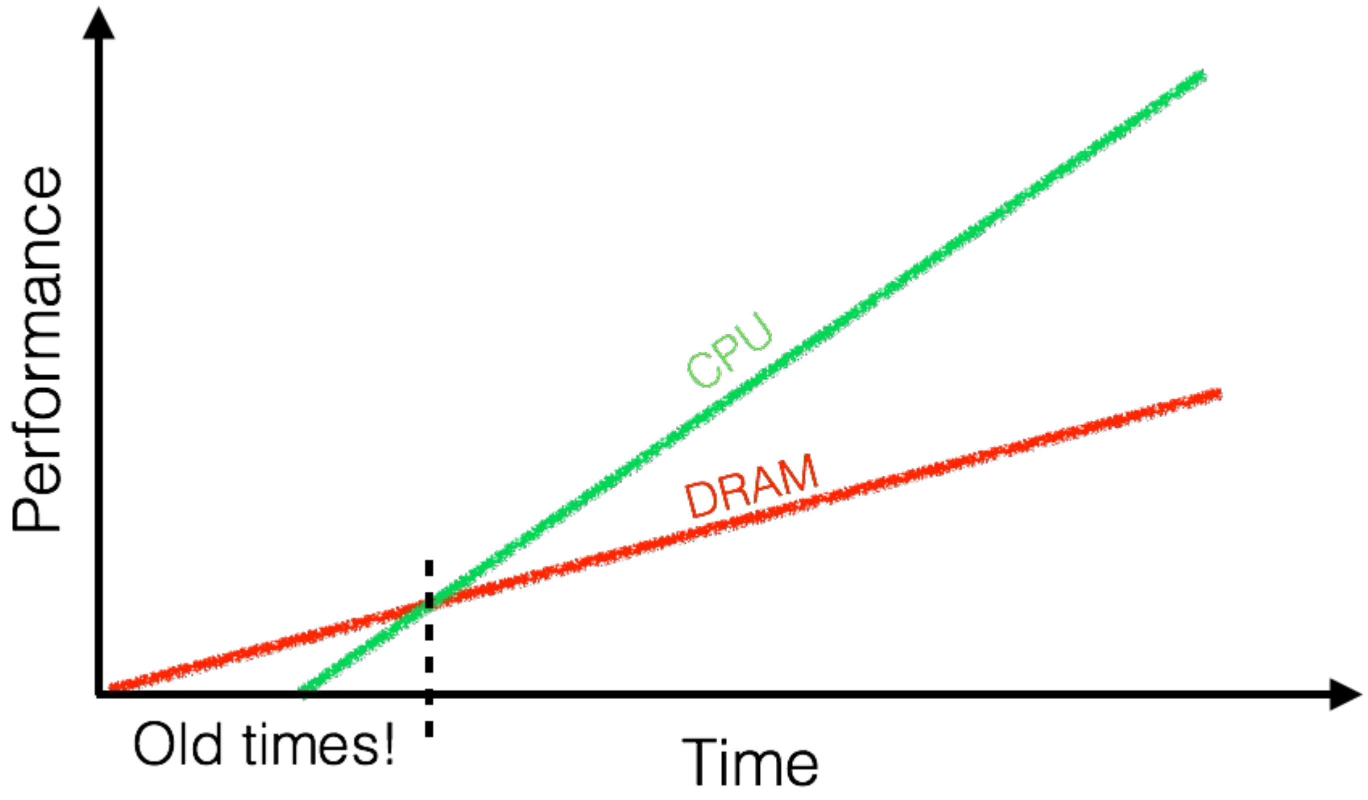




Acknowledgement: Prof. Manos Athanassoulis, BU

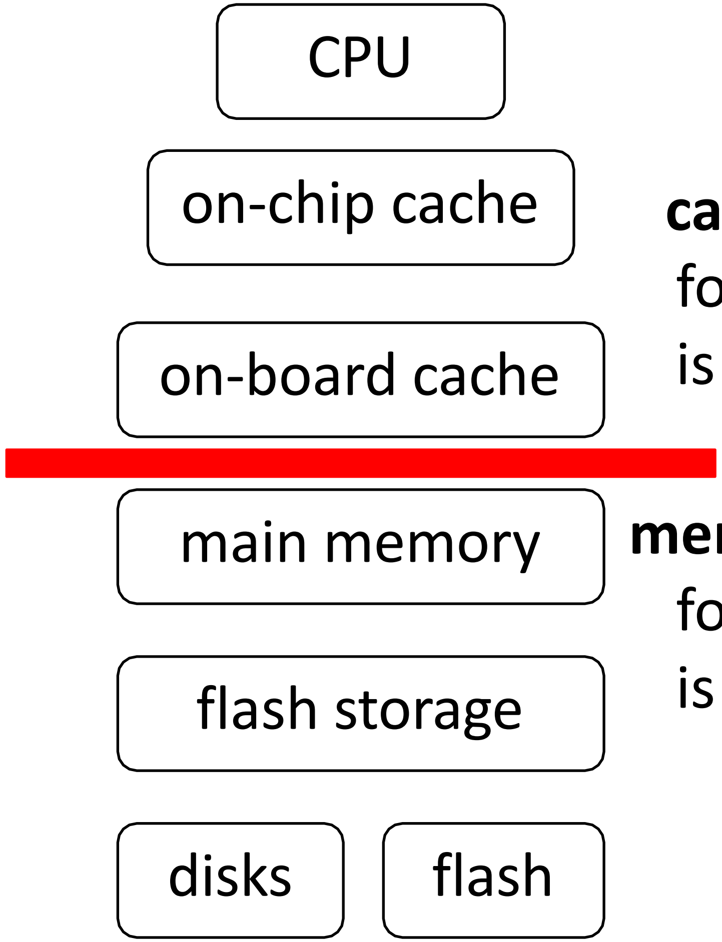
Memory wall





Acknowledgement: Prof. Manos Athanassoulis, BU

Cache/memory misses



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

memory miss: looking for something that is not in memory

what happens if I miss?

Acknowledgement: Prof. Manos Athanassoulis, BU

Data movement

CPU

on-chip cache

on-board cache

main memory

flash storage

disks | flash

data go through
all necessary levels

also read
unnecessary data

Photo by Gary Dineen/NBAE via Getty Images

need to read only X
read the whole page

X page

Acknowledgement: Prof. Manos Athanassoulis, BU

Data movement

CPU

on-chip cache

on-board cache

main memory

flash storage

data go through
all necessary levels

also read
unnecessary data

Photo by Gary Dineen/NBAE via Getty Images

need to read only X
read the whole page

X page

remember!
disk is millions (mem, hundreds) times slower than CPU

Acknowledgement: Prof. Manos Athanassoulis, BU

Page-based access & random access

query x<7

scan

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

Acknowledgement: Prof. Manos Athanassoulis, BU

$ 40 bytes

Page-based access & random access

query x<7

scan →

output

1, 5, 12, 24, 23

1, 5

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

Acknowledgement: Prof. Manos Athanassoulis, BU

Page-based access & random access

$ 40 bytes

query x<7

scan → output

1, 5, 12, 24, 23	2, 7, 13, 9, 8	1, 5

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23	2, 7, 13, 9, 8	10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

Acknowledgement: Prof. Manos Athanassoulis, BU

Page-based access & random access

$ 40 bytes

query x<7

scan → output

1, 5, 12, 24, 23

2, 7, 13, 9, 8

1, 5, 2

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

Acknowledgement: Prof. Manos Athanassoulis, BU

Page-based access & random access

$ 80 bytes

query x<7

scan →

output

1, 5, 12, 24, 23 | 2, 7, 13, 9, 8 | 1, 5, 2

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23 | 2, 7, 13, 9, 8 | 10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

Acknowledgement: Prof. Manos Athanassoulis, BU

$ 80 bytes

Page-based access & random access

query x<7

scan →

output

10, 11, 6, 14, 15	2, 7, 13, 9, 8	1, 5, 2

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23	2, 7, 13, 9, 8	10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

Acknowledgement: Prof. Manos Athanassoulis, BU

Page-based access & random access

$ 80 bytes

query x<7

scan →

output

10, 11, 6, 14, 15

2, 7, 13, 9, 8

1, 5, 2, 6

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

Acknowledgement: Prof. Manos Athanassoulis, BU

$ 120 bytes

Page-based access & random access

query x<7

scan →

output

10, 11, 6, 14, 15

2, 7, 13, 9, 8

1, 5, 2, 6

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

Acknowledgement: Prof. Manos Athanassoulis, BU

What if we had an oracle (perfect index)?

Acknowledgement: Prof. Manos Athanassoulis, BU

Page-based access & random access

query x<7

scan

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23	2, 7, 13, 9, 8	10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

SCHOOL OF COMPUTING
UNIVERSITY OF UTAH

Acknowledgement: Prof. Manos Athanassoulis, BU

Page-based access & random access

$ 40 bytes

query x<7

oracle

output

1, 5, 12, 24, 23

1, 5

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

Acknowledgement: Prof. Manos Athanassoulis, BU

Page-based access & random access

$ 40 bytes

query x<7

oracle

output

1, 5, 12, 24, 23

2, 7, 13, 9, 8

1, 5

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

Acknowledgement: Prof. Manos Athanassoulis, BU

Page-based access & random access

$ 40 bytes

query x<7

oracle

output

1, 5, 12, 24, 23

2, 7, 13, 9, 8

1, 5, 2

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

Acknowledgement: Prof. Manos Athanassoulis, BU

Page-based access & random access

$ 80 bytes

query x<7

oracle

output

1, 5, 12, 24, 23

2, 7, 13, 9, 8

1, 5, 2

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

Acknowledgement: Prof. Manos Athanassoulis, BU

Page-based access & random access

$ 80 bytes

query x<7

oracle

output

10, 11, 6, 14, 15

2, 7, 13, 9, 8

1, 5, 2

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23

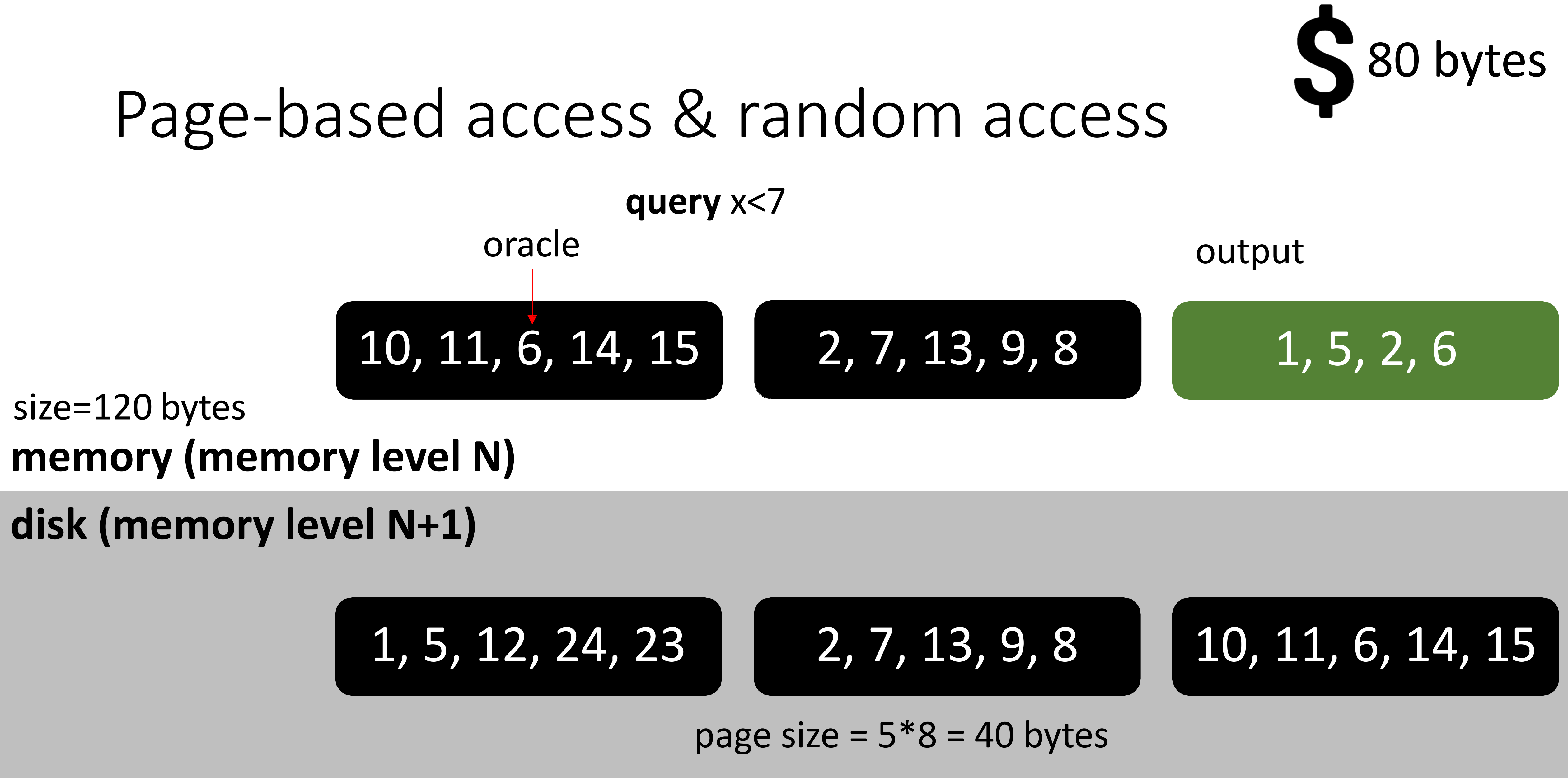
2, 7, 13, 9, 8

10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

Acknowledgement: Prof. Manos Athanassoulis, BU

Page-based access & random access



Acknowledgement: Prof. Manos Athanassoulis, BU

Page-based access & random access

$ 120 bytes

query x<7

was the oracle helpful?

oracle

output

10, 11, 6, 14, 15	2, 7, 13, 9, 8	1, 5, 2, 6

size=120 bytes

memory (memory level N)

disk (memory level N+1)

1, 5, 12, 24, 23	2, 7, 13, 9, 8	10, 11, 6, 14, 15

page size = 5*8 = 40 bytes

Acknowledgement: Prof. Manos Athanassoulis, BU

When is the oracle helpful?

for which query would an oracle help us?

how to decide whether to use the oracle?

1, 5, 12, 24, 23

2, 7, 13, 9, 8

10, 11, 6, 14, 15

Acknowledgement: Prof. Manos Athanassoulis, BU

how we store data

layouts, indexes

every **byte** counts

overheads and tradeoffs

know the **query**

access path selection

index
design space

Acknowledgement: Prof. Manos Athanassoulis, BU

Rules of thumb

sequential access

read one block; consume it completely; discard it; read next;

hardware can predict and start prefetching

prefetching can exploit full memory/disk bandwidth

random access

read one block; consume it partially; discard it; (may re-use);

read random next;

ideal random access?

the one that helps us **avoid a large number of accesses** (random or sequential)

Acknowledgement: Prof. Manos Athanassoulis, BU

The language of efficient systems: C/C++

why?

low-level control over hardware

make decisions about physical data placement and consumptions

fewer assumptions

Acknowledgement: Prof. Manos Athanassoulis, BU

The language of efficient systems: C/C++

why?

low-level control over hardware

we want you in the project to make low-level decisions

Acknowledgement: Prof. Manos Athanassoulis, BU

Next class

- In-memory indexing

Make sure to read the related papers from the reading list

Acknowledgement: Prof. Manos Athanassoulis, BU