Database Management Systems

Lecture 12

Parallel Databases

Sabina S. CS

Parallel Database Systems

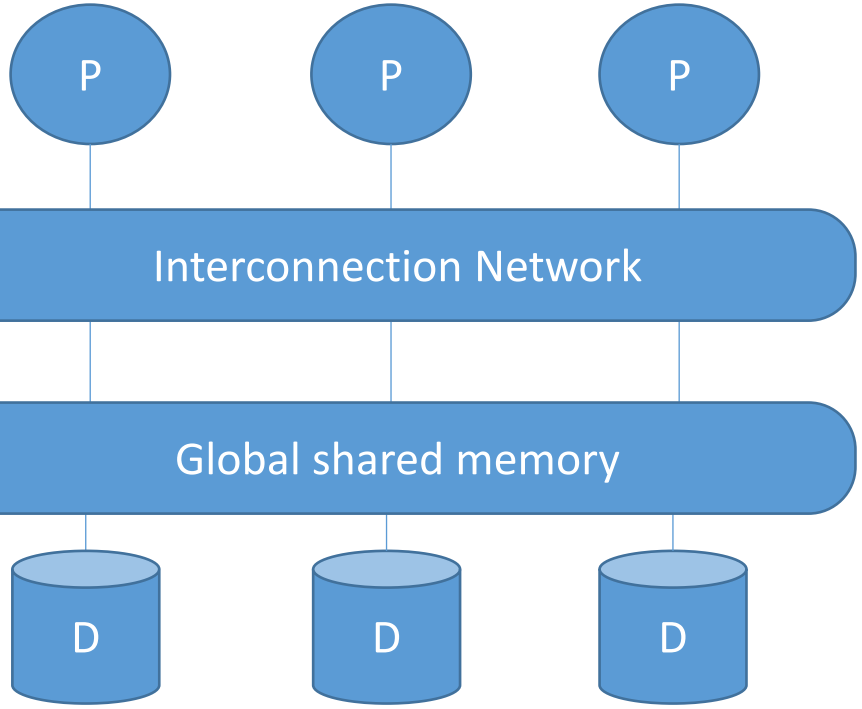
- performance improvement
  - parallelize operations:
    - loading data
    - building indexes
    - query evaluation
  - data can be distributed, but distribution is dictated solely by performance reasons

Parallel Databases - Architectures

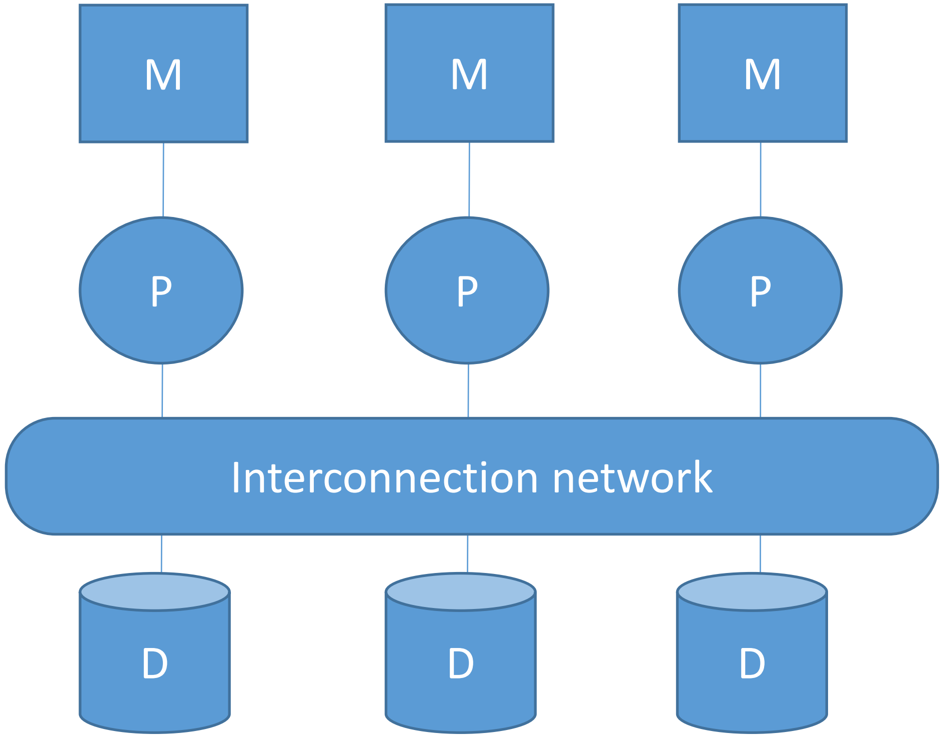
- *shared-memory*
- *shared-disk*
- *shared-nothing*

Parallel Databases - Architectures

- *shared-memory*
    - several CPUs:
        - attached to an interconnection network
        - can access a common region in the main memory

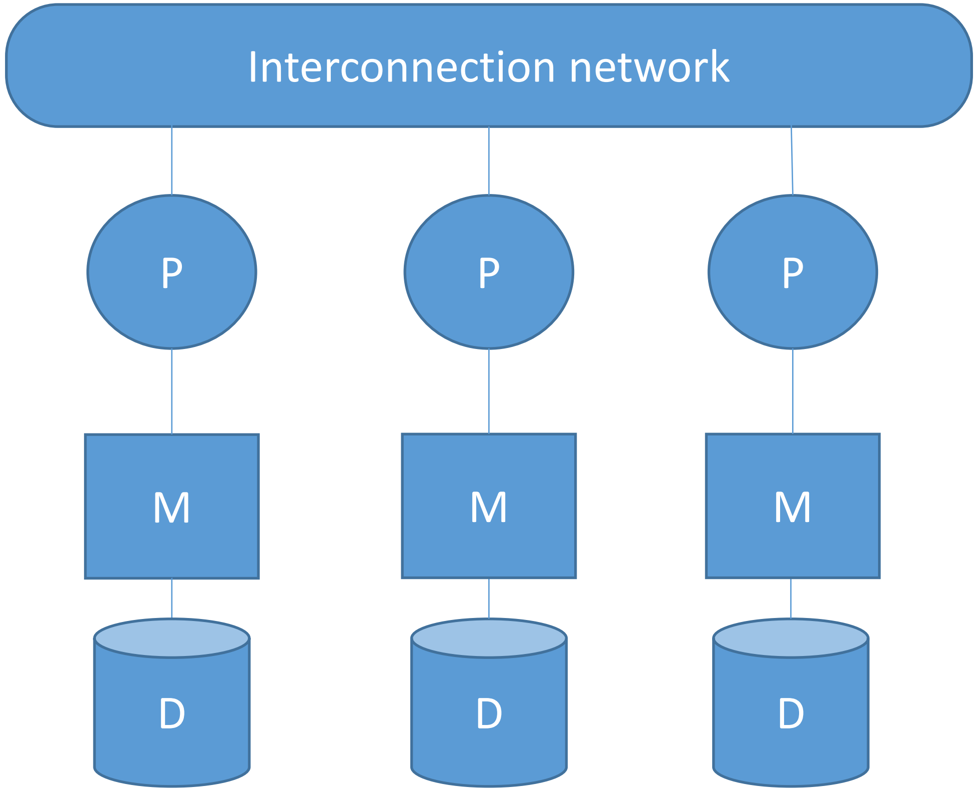


- *shared-disk*
    - a CPU:
        - its own private memory
        - can access all disks through a network



Parallel Databases - Architectures

- *shared-nothing*
  - a CPU:
    - its own local main memory
    - its own disk space
  - 2 different CPUs cannot access the same storage area
  - CPUs communicate through a network



Interference

- specific to shared-memory and shared-disk architectures
- add CPUs:
  - increased contention for memory and network bandwidth

  => existing CPUs are slowing down
- main reason that led to the shared-nothing architecture, currently considered as the best option for large parallel database systems

The Shared-Nothing Architecture

- linear speed-up & linear scale-up

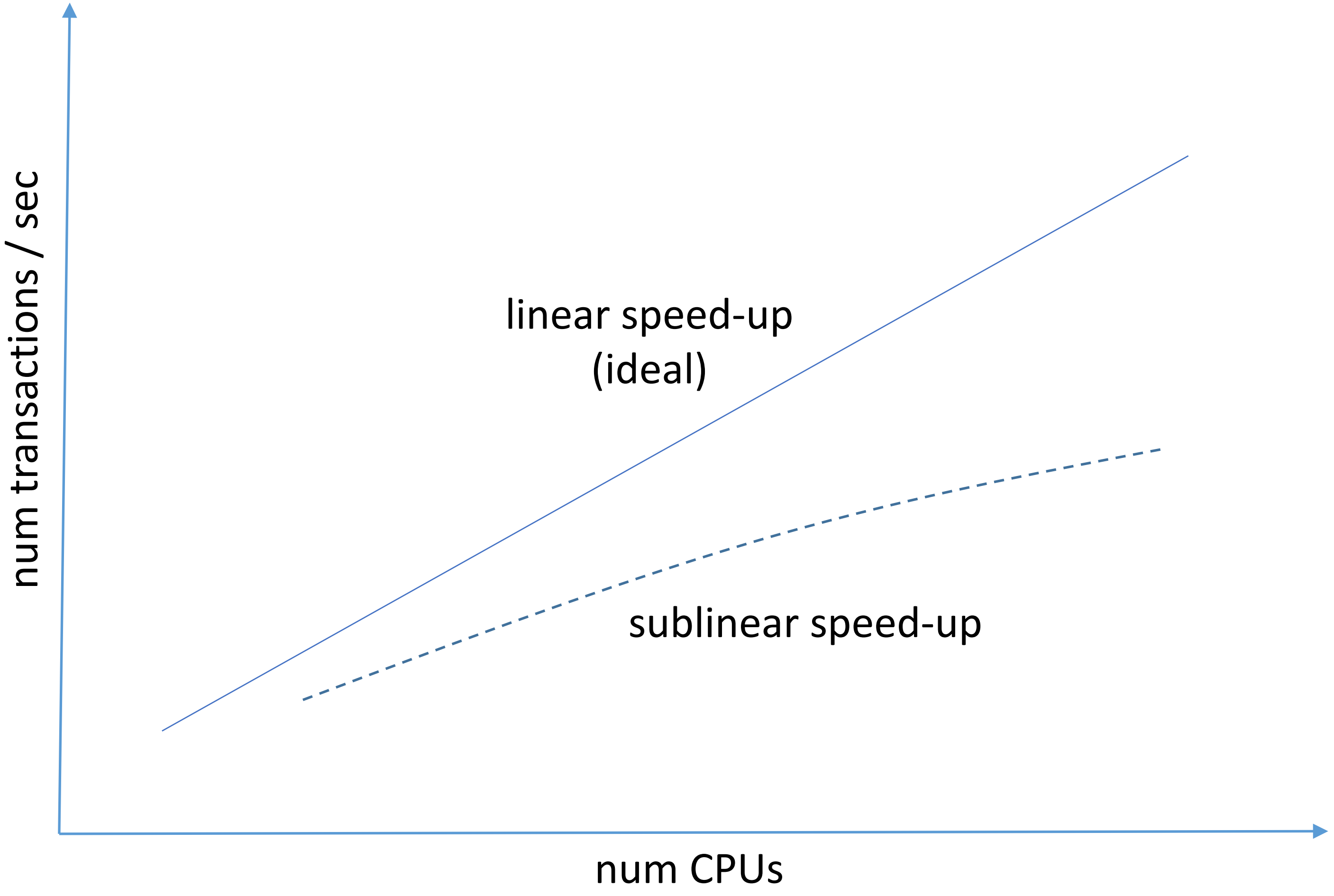
- linear speed-up
  - required processing time for operations decreases proportionally to the increase in the number of CPUs and disks

- linear scale-up
  - num. of CPUs and disks grows proportionally to the amount of data

  => performance is sustained

The Shared-Nothing Architecture

- speed-up

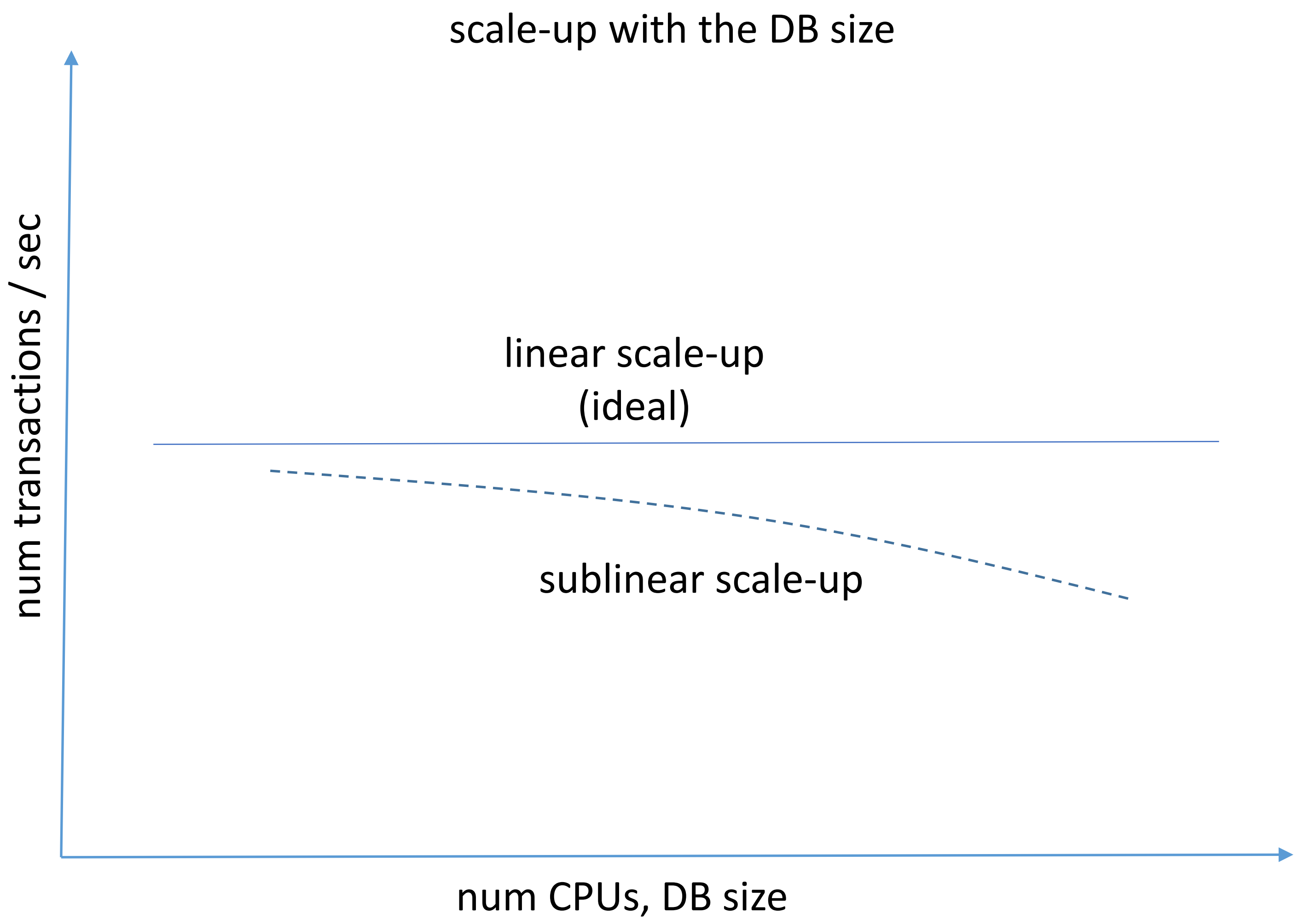


- DB size - fixed
- add CPUs

=> more transactions can be executed per second

The Shared-Nothing Architecture

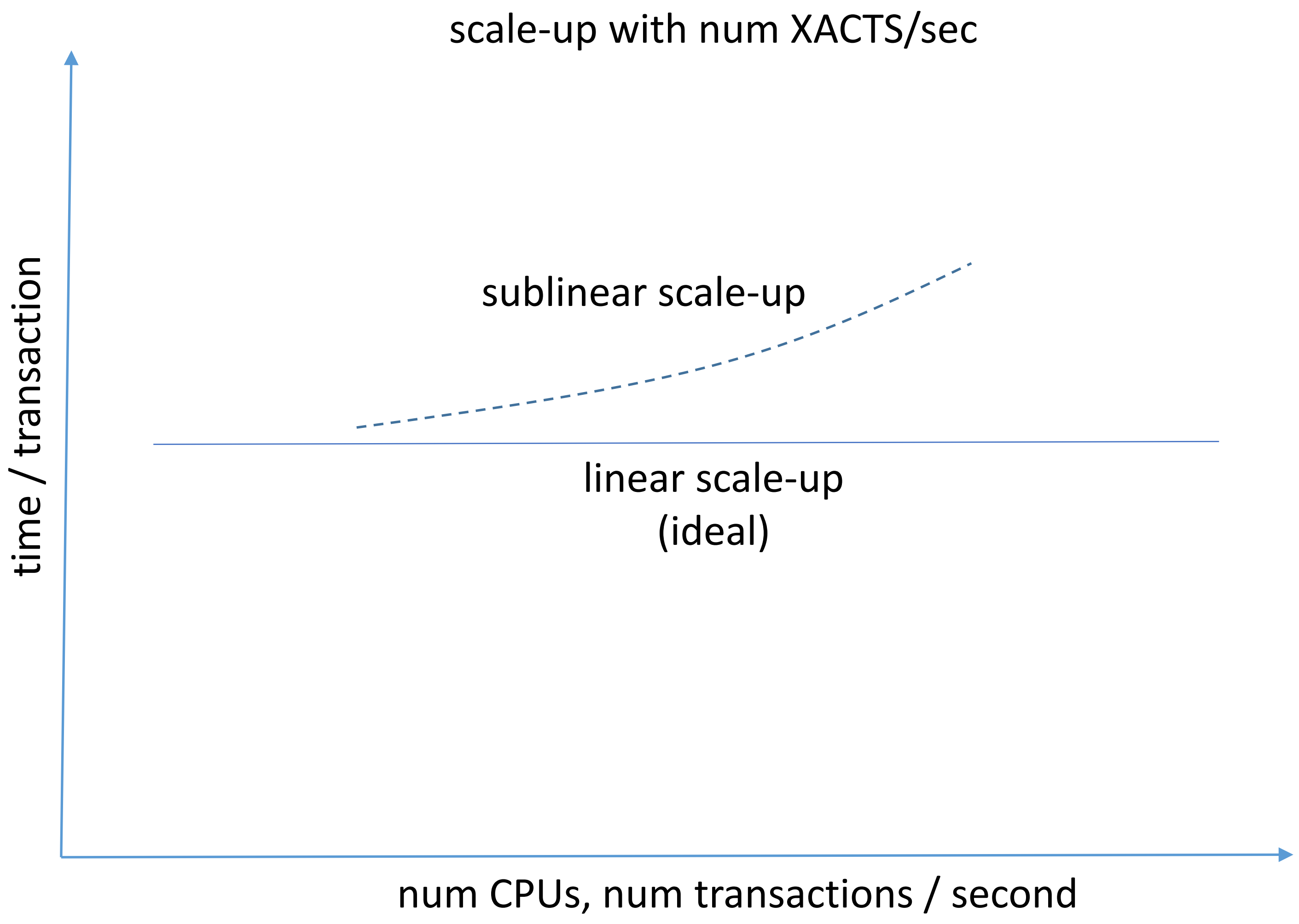
- scale-up



- the number of transactions executed per second, as the DB size and the number of CPUs increase

The Shared-Nothing Architecture

- scale-up



* alternative

- add CPUs as the number of transactions executed per second increases
- evaluate the time required for a transaction

Parallel Query Evaluation

- context
  - DBMS based on a shared-nothing architecture
- evaluate a query in a parallel manner
- operators in an execution plan can be evaluated in parallel
  - 2 operators are evaluated in parallel
  - one operator is evaluated in a parallel manner
- an operator is said to *block* if it doesn’t produce results until it consumes all its inputs (e.g., sorting, aggregation)
- pipelined parallelism
  - an operator consumes the output of another operator
  - limited by blocking operators

Parallel Query Evaluation

- parallel evaluation on partitioned data
  - every operator in a plan can be evaluated in a parallel manner by partitioning the input data
  - partitions are processed in parallel, the results are then combined
- processor = CPU + its local disk

Parallel Query Evaluation – Data Partitioning

- horizontally partition a large dataset on several disks
- partitions are then read / written in parallel
- *round-robin* partitioning
  - n processors
  - the i^{th} tuple is assigned to processor i % n
- hash partitioning
  - determine the processor for a tuple t
    - apply a hash function to t ((some of) its attributes)
- range partitioning
  - n processors
  - order tuples conceptually
  - choose n ranges for the sorting key values s.t. each range contains approximately the same number of tuples
  - tuples in range i are assigned to processor i

Parallel Query Evaluation – Data Partitioning

- queries that scan the entire relation
  - round-robin partitioning - suitable
- queries that operate on a subset of tuples
  - equality selection, e.g., age = 30
    - tuples partitioned on the attributes in the selection condition, e.g., age
    - hash and range partitioning are better than round-robing (one can access only the disks containing the desired tuples)
  - range selection, e.g., 20 < age < 30
    - range partitioning is better than hash partitioning (it’s likely that the desired tuples are grouped on several processors)

Parallelizing Individual Operations

- context
    - DBMS based on a shared-nothing architecture
- each relation is horizontally partitioned on several disks
- scanning a relation
    - pages can be read in parallel
    - obtained tuples can than be reunited
    - similarly – obtain all tuples that meet a selection condition

Parallelizing Individual Operations

- sorting
  - v1
    - each CPU sorts the relation fragment on its disk
    - subsequently, the sorted tuple sets are merged
  - v2
    - redistribute tuples in the relation using range partitioning
    - each processor sorts its tuples with a sequential sorting algorithm => several sorted runs on the disk
    - merge runs => sorted version of the set of tuples assigned to the current processor
    - obtain the entire sorted relation
      - visit processors in an order corresponding to their assigned ranges and scan the tuples

Parallelizing Individual Operations

- sorting
    - v2
        - challenges
            - range partitioning – assign approximately the same number of tuples to each processor
            - a processor that receives a disproportionately large number of tuples will limit scalability

References

- [Ra02] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems (3rd Edition), McGraw-Hill, 2002
- [Da03] DATE, C.J., An Introduction to Database Systems (8th Edition), Addison-Wesley, 2003
- [Ga09] GARCIA-MOLINA, H., ULLMAN, J., WIDOM, J., Database Systems: The Complete Book (2nd Edition), Pearson Education, 2009
- [Ra02S] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems, Slides for the 3rd Edition, http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html
- [Si19] SILBERSCHATZ, A., KORTH, H., SUDARSHAN, S., Database System Concepts (7th Edition), McGraw-Hill, 2019
- [Si19S] SILBERSCHATZ, A., KORTH, H., SUDARSHAN, S., Database System Concepts, Slides for the 7th Edition, http://codex.cs.yale.edu/avi/db-book/
- [Ul11] ULLMAN, J., WIDOM, J., A First Course in Database Systems, http://infolab.stanford.edu/~ullman/fcdb.html