Distributed, Parallel, and Alternative Architecture Databases

Luiz Celso Gomes Jr - André Santanchè

MC536 2013/2

Based on:

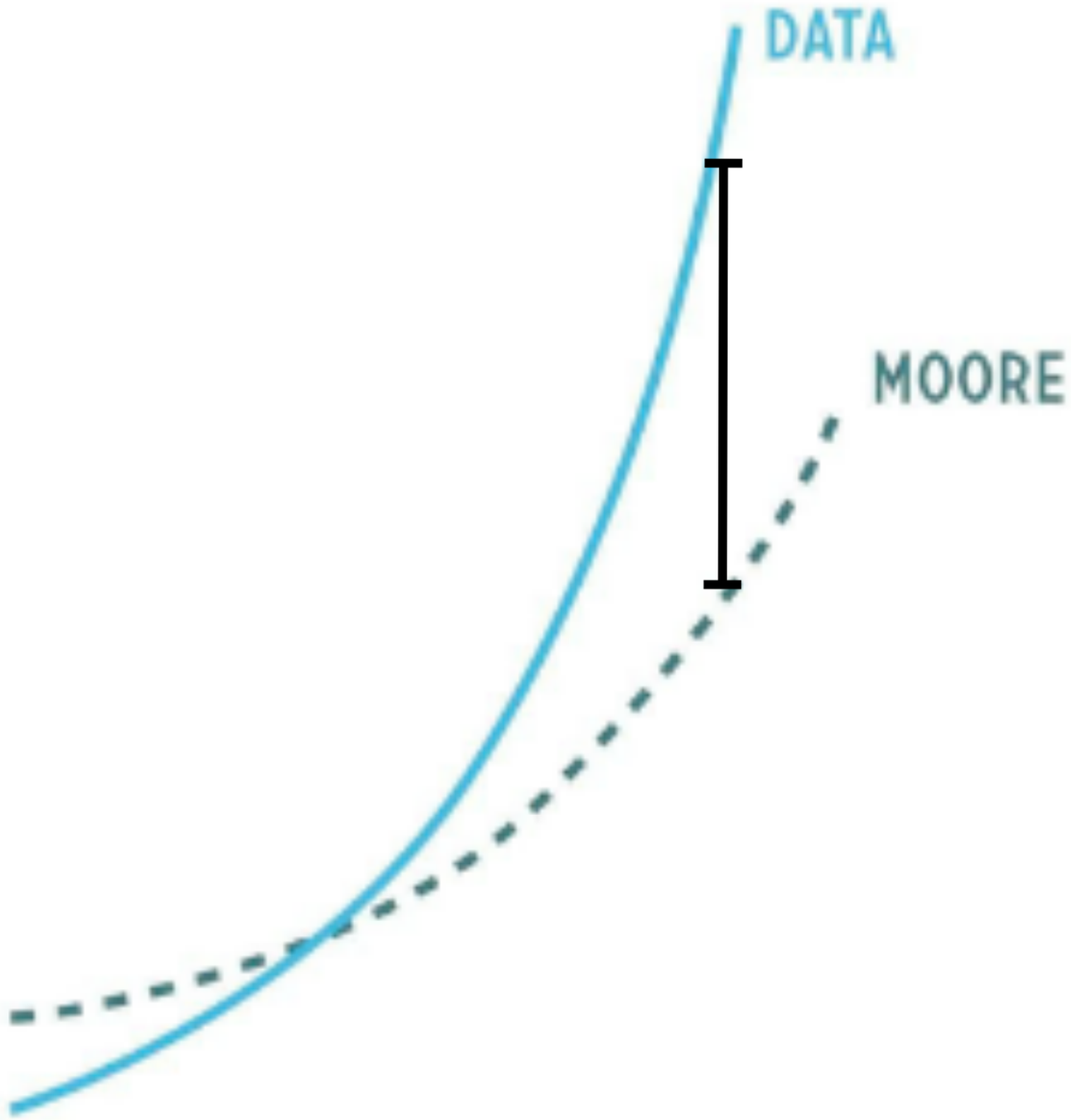
Fundamentals of Database Systems, 6th Edition. Elmasri and Navathe

Database Management Systems, 2nd Edition. Raghu Ramakrishnan and Johannes Gehrke

Outline

- Terminology
- Parallel Databases
- Distributed Databases
- Client-server Architecture
- Alternative Architectures

Need for speed



Exercício 1

- [Preliminares] Suponha que a DAC esteja enfrentando problemas para atender as consultas online de CR dos alunos (o tempo de resposta é muito longo). As tabelas do banco são descritas abaixo. Quais técnicas (ao menos duas) vocês poderiam aplicar para melhorar o desempenho das consultas?

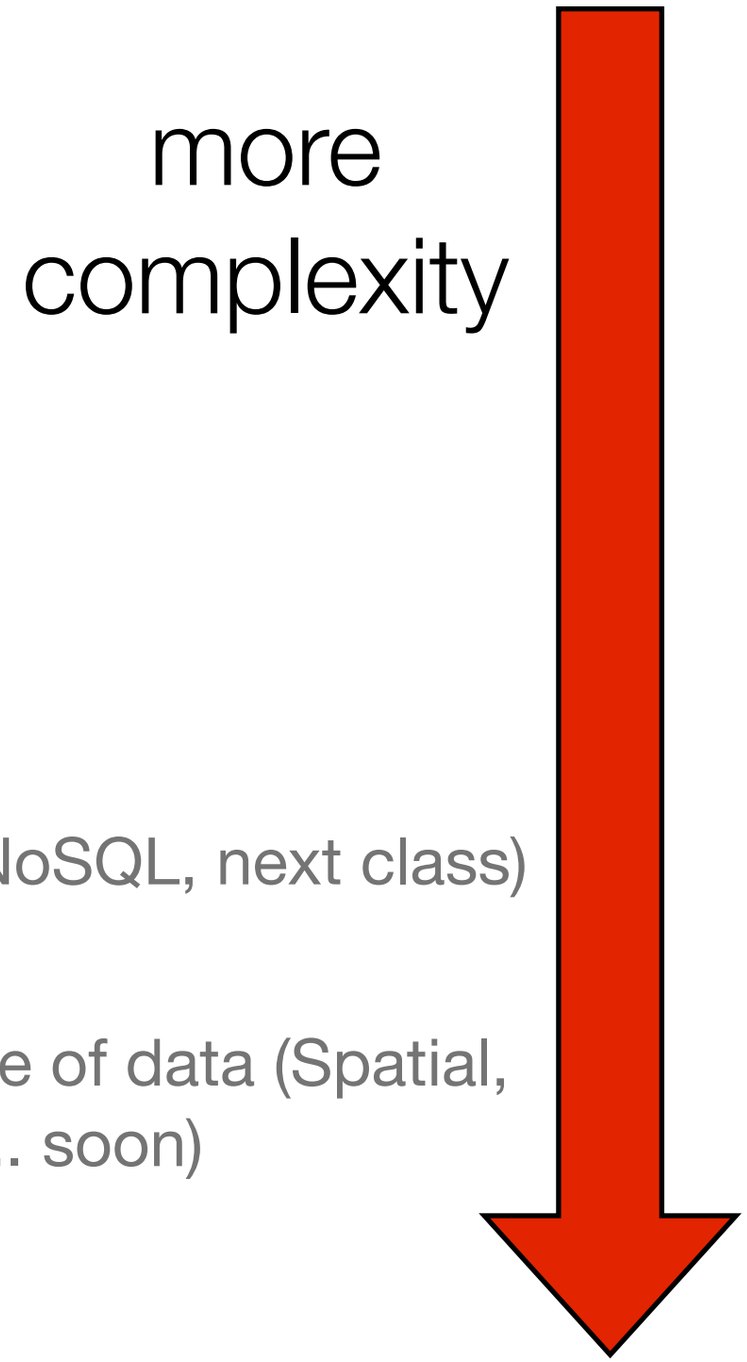
- Aluno(RA, nome, curso)

- Disciplina(codigo, nome)

- Cursa(RA, codigo, nota)

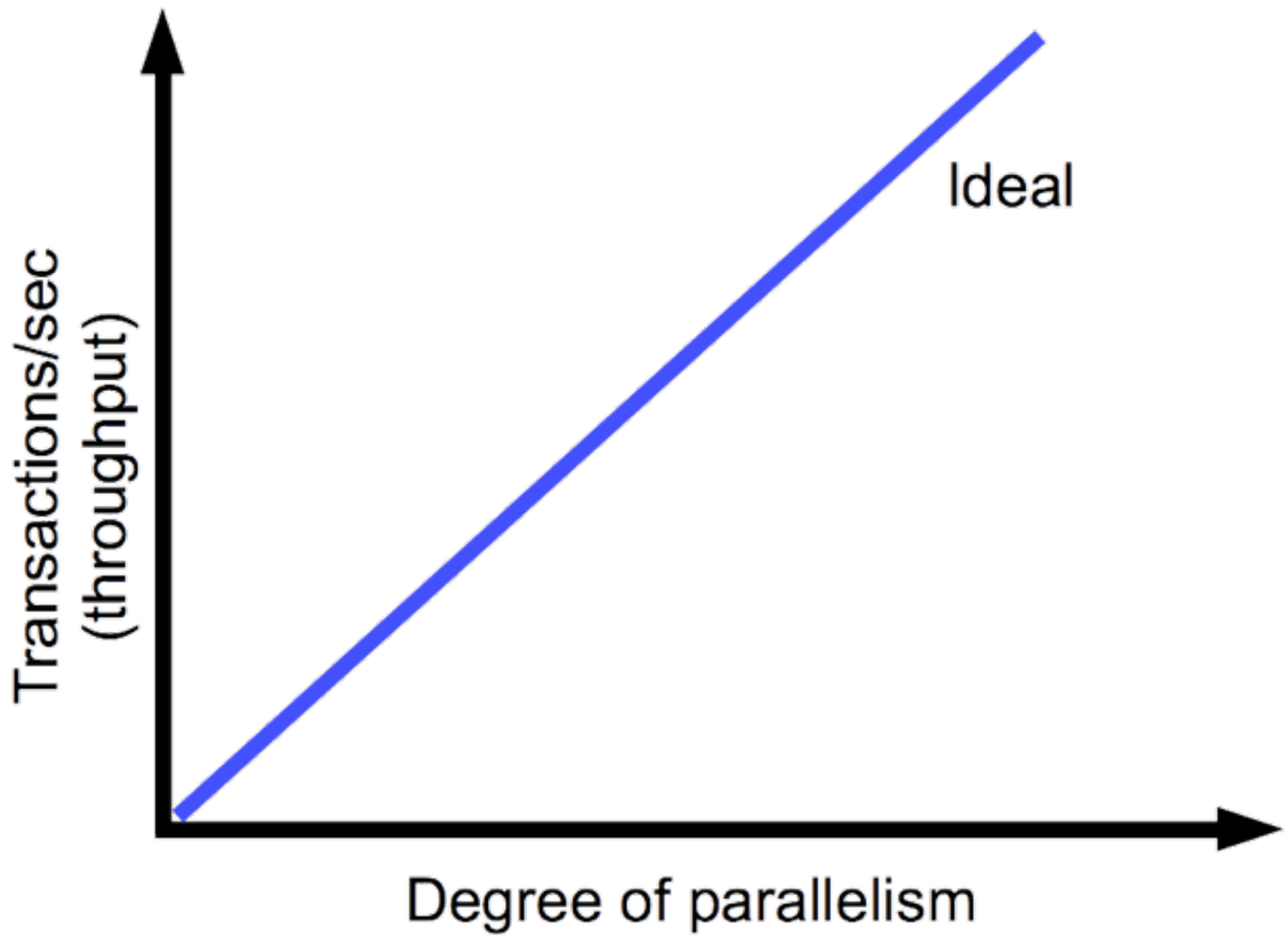
Need for speed

- Bigger computers: Faster CPUs
- Parallel: Multiple CPUs
- Distributed: Multiple Servers
- Alternative Architectures: Specialized CPUs
- Alternative Frameworks: adapt DBMS to the task (NoSQL, next class)
- Alternative Data Structures: adapt DBMS to the type of data (Spatial, Multimedia, Temporal, Active, Documents, Graphs... soon)



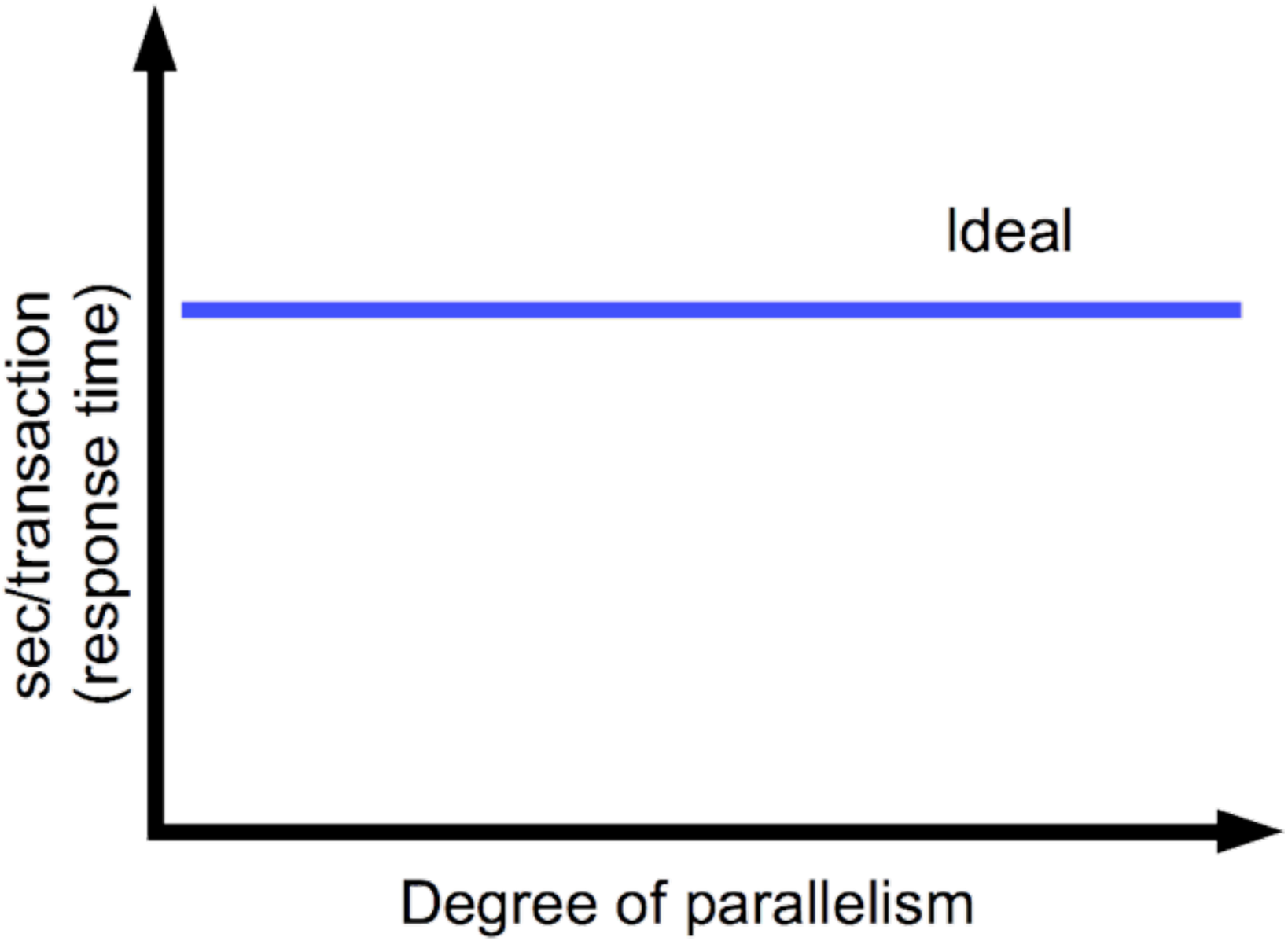
Terminology - Speed-Up

- More resources means proportionally less time for given amount of data.



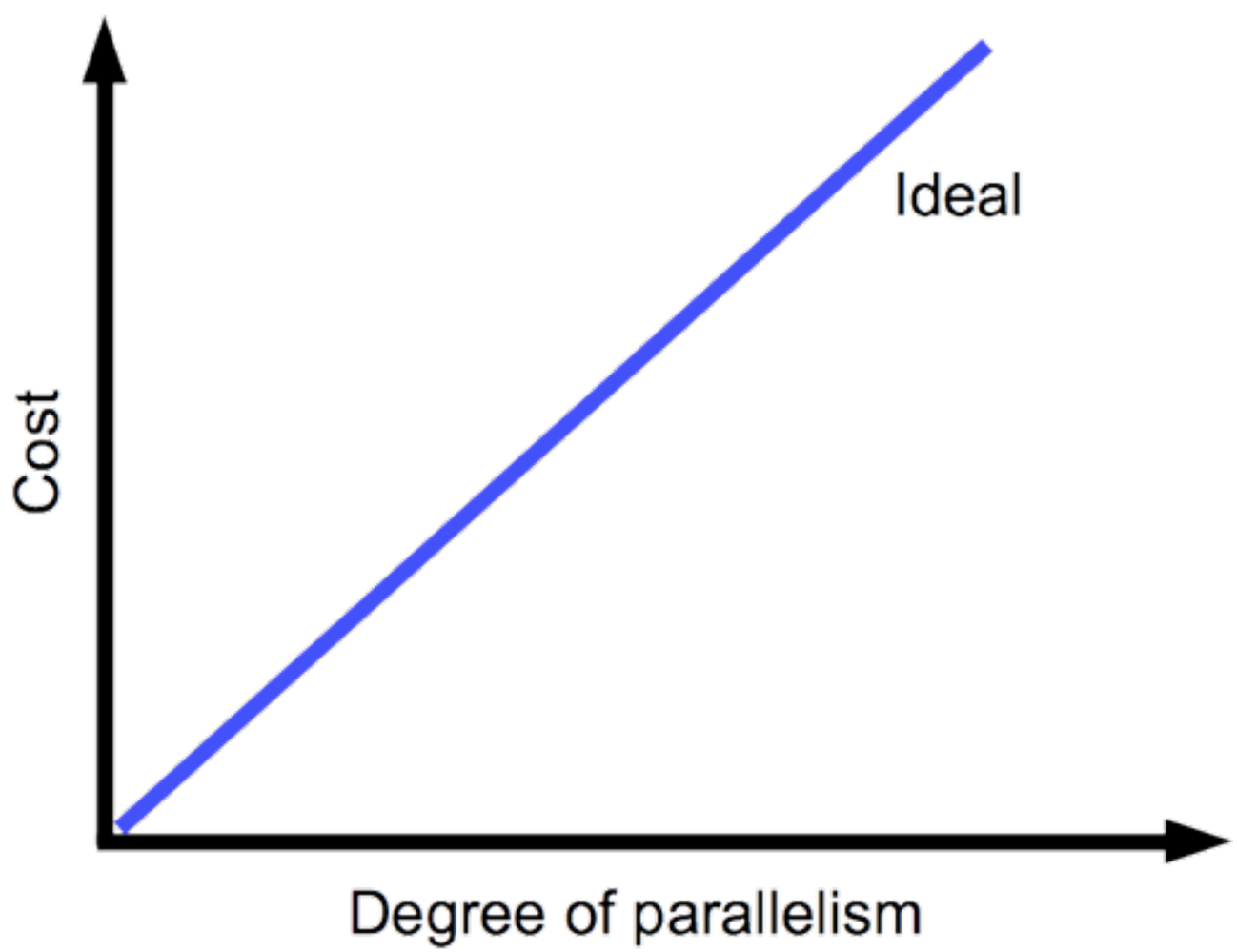
Terminology - Scale-Up

- If resources increased in proportion to increase in data size, time is constant.



Also: proportional cost

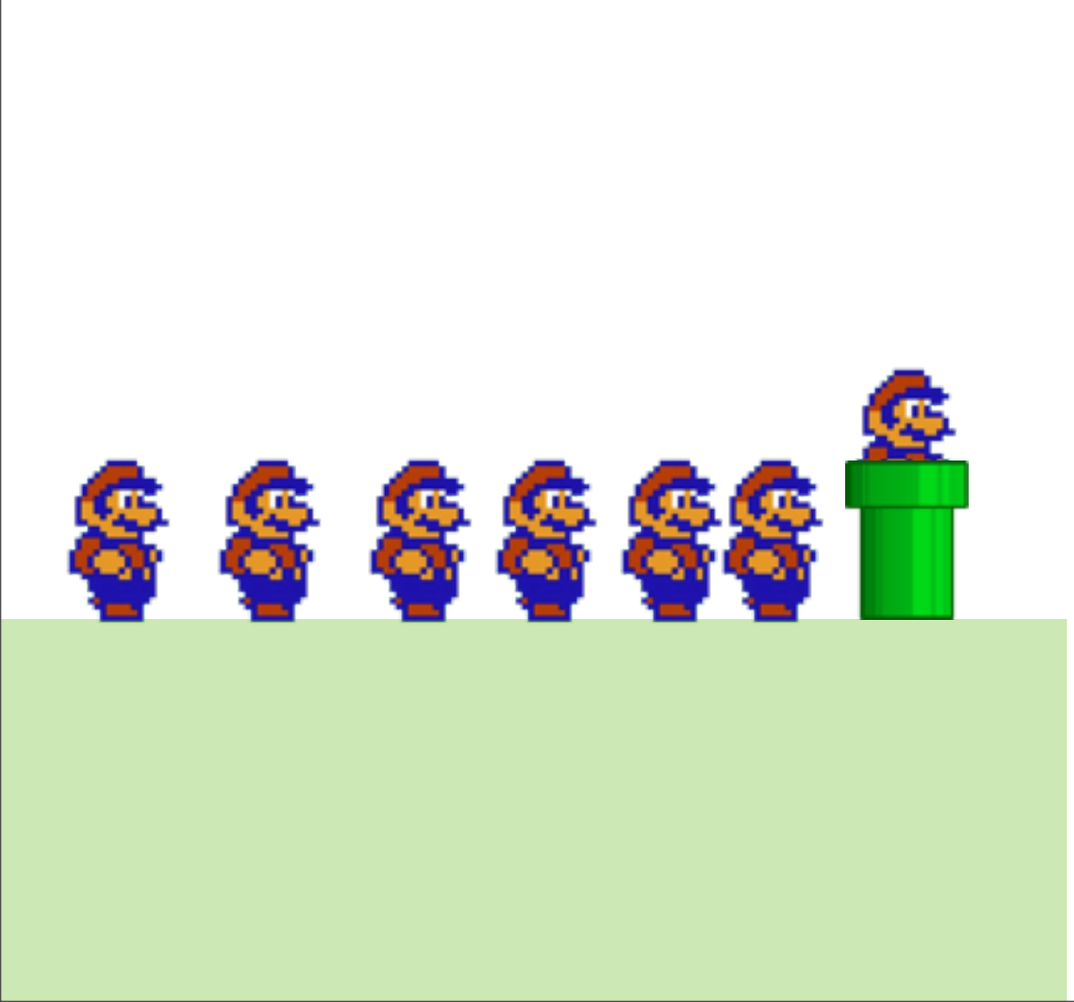
- Infrastructures cost should remain proportional as number of CPUs grow.

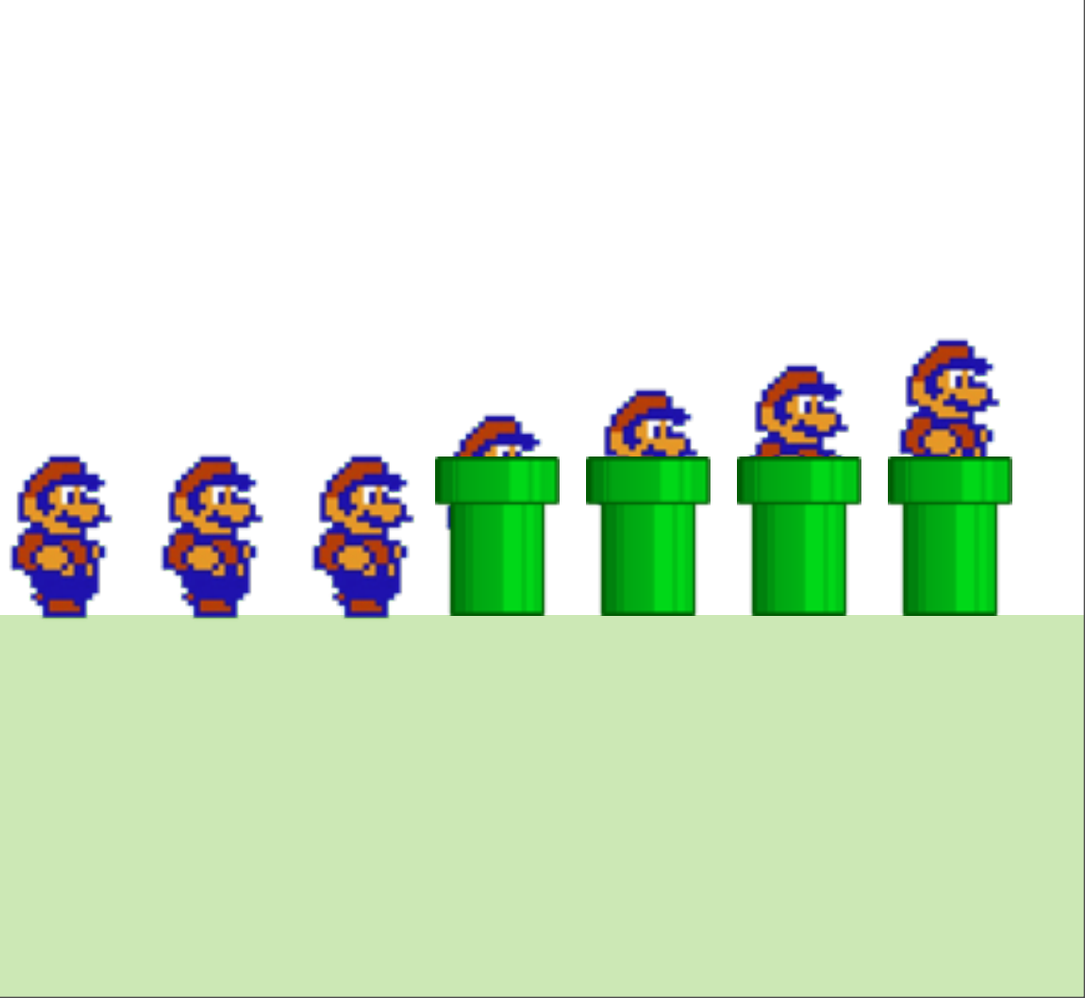


Parallel Databases

Parallelism

- More processors -> Better Throughput
- Divide big problems into smaller ones





DBMS are suited for parallelism

- Bulk processing of data partitions
- Natural pipelining (execution plan)
- Users don’t need to write parallel queries

Parallelism over time

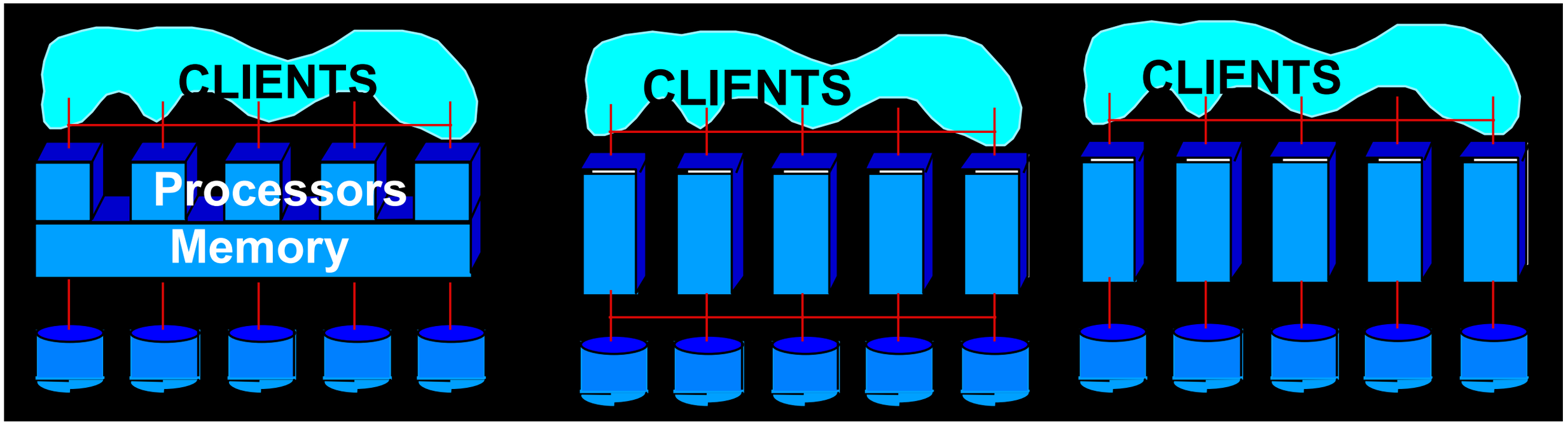
- Before: big parallel computers
- Now: small multicore servers organized in clusters

Levels of sharing

- Shared memory
- Shared disk
- Shared nothing (network)

Architecture Issue: Shared What?

Shared Memory	Shared Disk	Shared Nothing (network)



- Easy to program
- Expensive to build
- Difficult to scale up

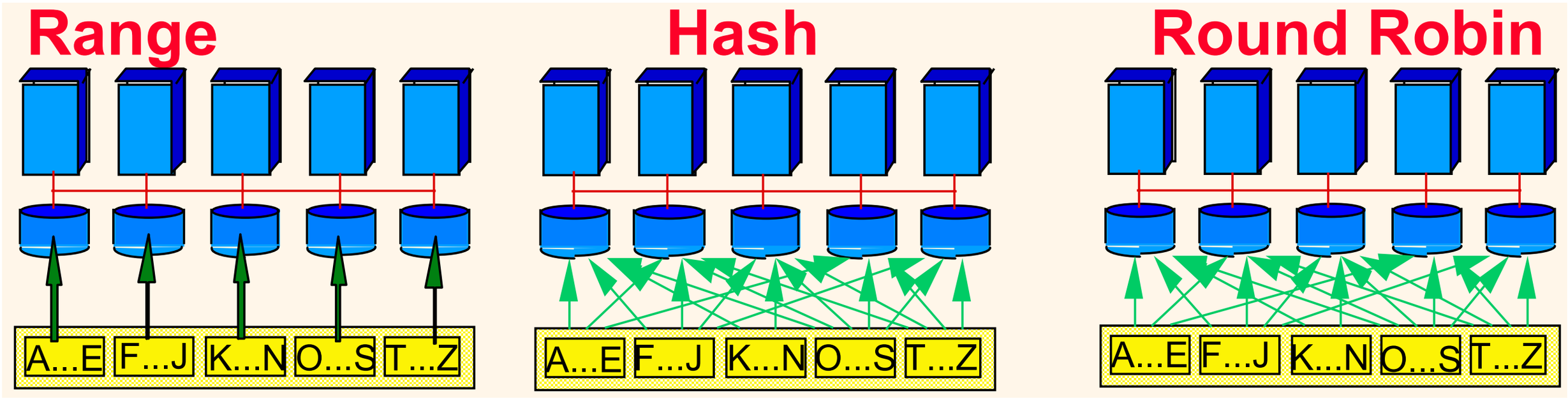
- Hard to program
- Cheap to build
- Easy to scale up

Types of DBMS parallelism

- Intra-operator parallelism (focus here)
    - get all machines working to compute a given operation (scan, sort, join)
- Inter-operator parallelism
    - each operator may run concurrently on a different site (exploits pipelining)
- Inter-query parallelism
    - different queries run on different sites

Automatic Data Partitioning

Partitioning a table:



Good for equijoins, range queries, group-by

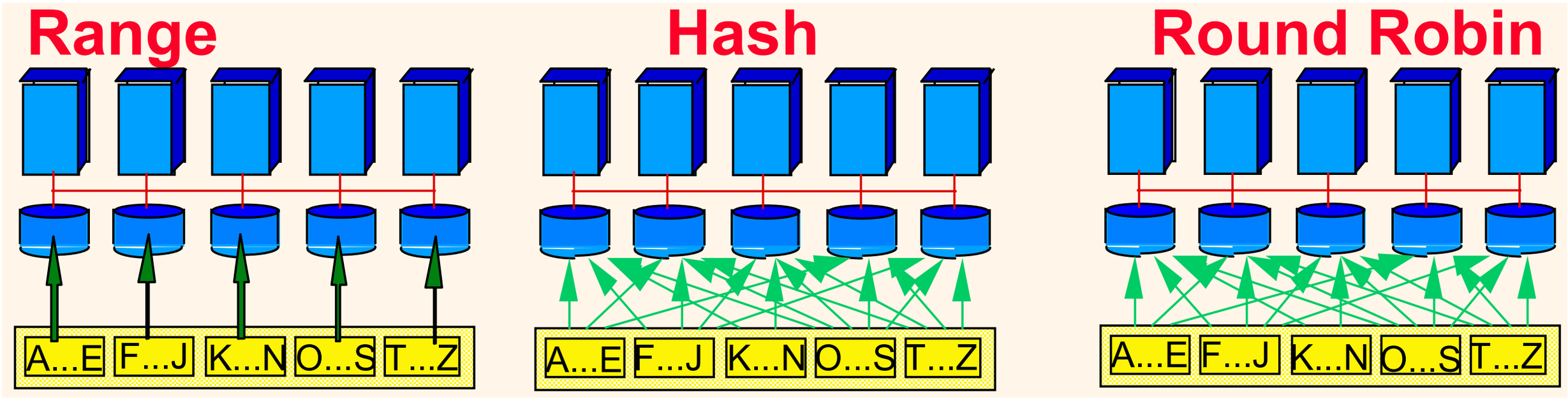
Good for equijoins

Good to spread load

Shared disk and memory less sensitive to partitioning,
Shared nothing benefits from "good" partitioning

Exercício 2

- Ordene os tipos de técnica de particionamento de dados (Range, Hash, Round Robin) de acordo com o tamanho físico dos índices que precisam ser mantidos para localizar o disco ou CPU que contém cada tupla. Justifique sua resposta.

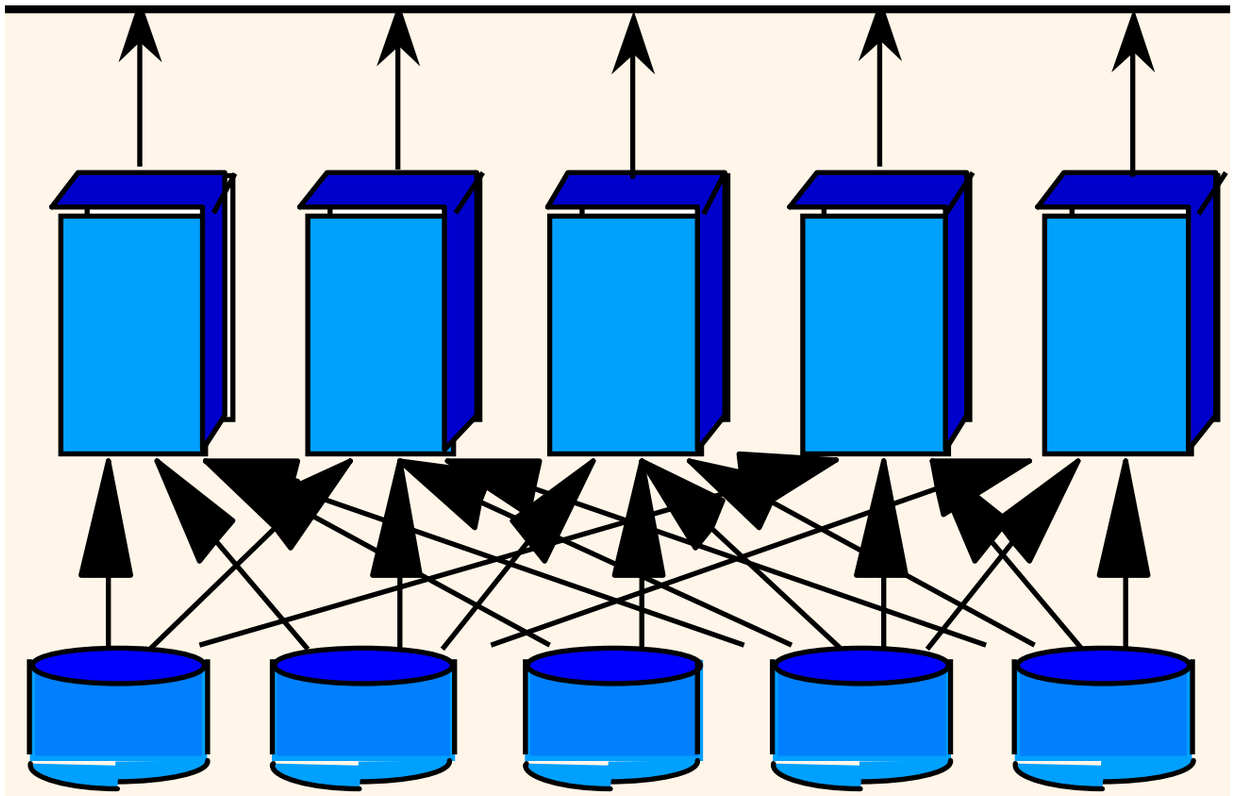


Parallel Scans

- Scan in parallel, and merge.
- Selection may not require all sites for range or hash partitioning.
- Indexes can be built at each partition.

Parallel Sorting

- Scan in parallel, and range-partition as you go.
- As tuples come in, begin “local” sorting on each
- Resulting data is sorted, and range-partitioned.
- Problem: skew!
- Solution: “sample” the data at start to determine partition points.



Parallel Joins

- Nested loop:
  - Each outer tuple must be compared with each inner tuple that might join.
  - Easy for range partitioning on join cols, hard otherwise!
- Sort-Merge (or plain Merge-Join):
  - Sorting gives range-partitioning.
    - But what about handling 2 skews?
  - Merging partitioned tables is local.

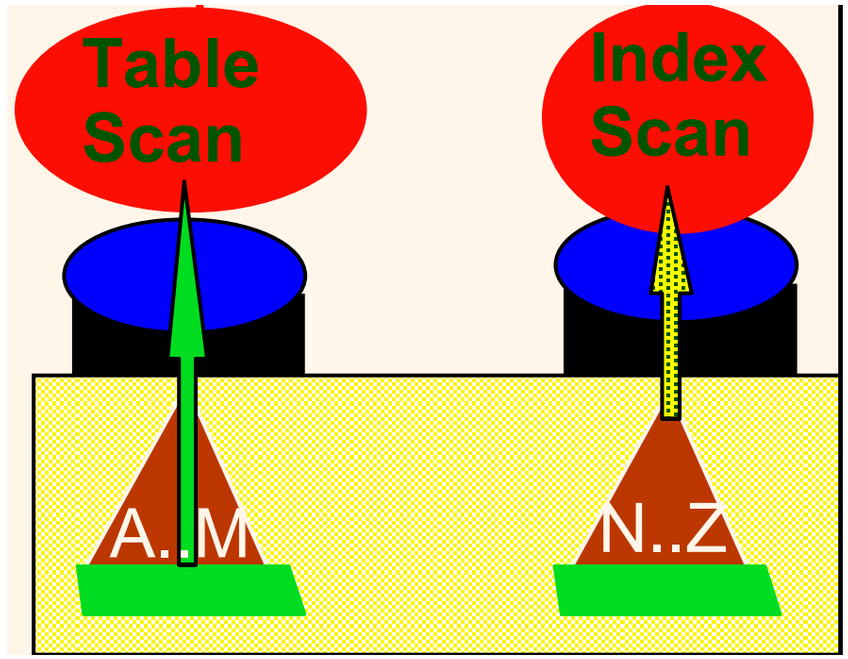
Observations

- It is relatively easy to build a fast parallel query executor
- It is hard to write a robust and world-class parallel query optimizer.
    - There are many tricks.
    - One quickly hits the complexity barrier.
    - Still open research!

Distribution adds complexity

- Best serial plan != Best parallel plan! Why?
- Trivial example:
  - Table partitioned with local secondary index at two nodes
  - Range query: all of node 1 and 1% of node 2.
  - Node 1 should do a scan of its partition.
  - Node 2 should use secondary index.

```
SELECT *
FROM telephone_book
WHERE name < “NoGood”;
```



Parallel DBMS Summary

- Parallelism natural to query processing:
    - Both pipeline and partition parallelism!
- Shared-Nothing vs. Shared-Mem
    - Shared-disk too, but less standard
    - Shared-mem easy, costly. Doesn’t scaleup.
    - Shared-nothing cheap, scales well, harder to implement
- Intra-op, Inter-op, & Inter-query parallelism all possible.

Distributed Databases

Distributed Databases - Definition

- A **transaction** can be executed by multiple **networked computers** in a unified manner.

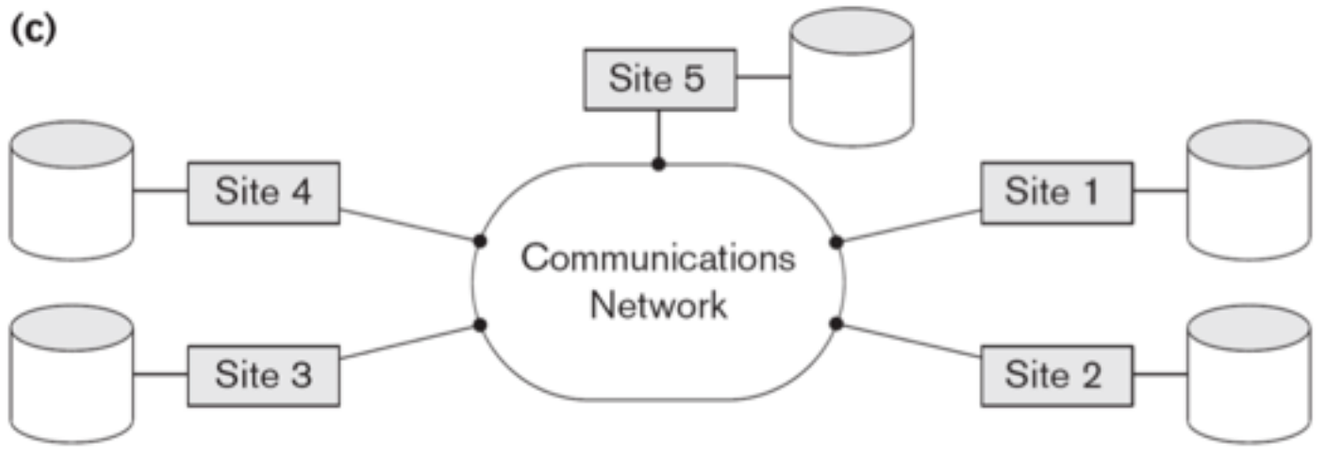
- A **distributed database** (DDB) is a collection of multiple logically related database **distributed over a computer network**

- A **distributed database management system** (DDBMS) is a software **system that manages** a distributed database while making the distribution **transparent** to the user.

Distributed Database System

- Management of distributed data with different levels of transparency:
  - This refers to the physical placement of data (files, relations, etc.) which is not known to the user (distribution transparency).

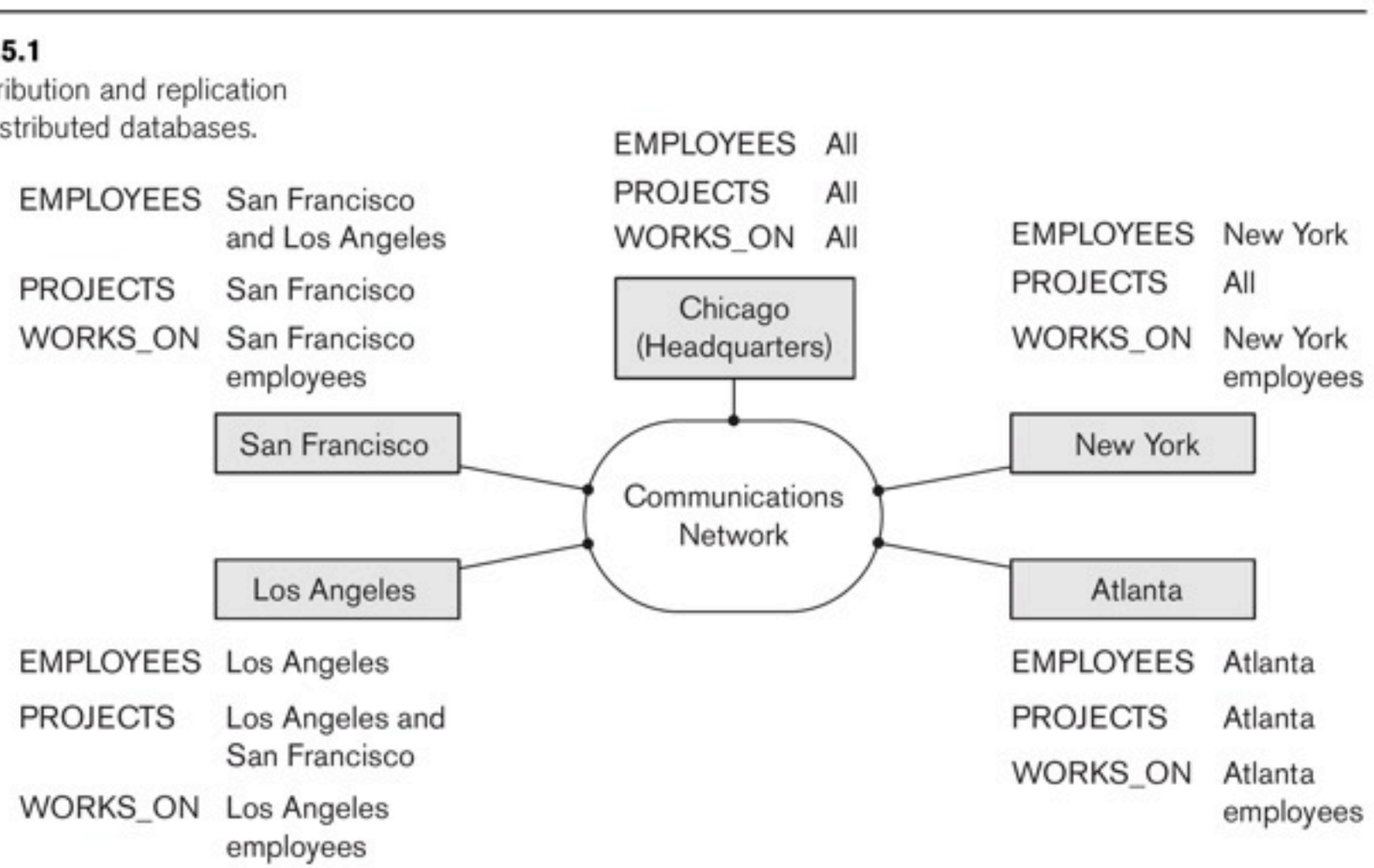
Figure 25.3
Some different database system architectures. (a) Shared nothing architecture. (b) A networked architecture with a centralized database at one of the sites. (c) A truly distributed database architecture.



Distributed Database System

- Transparency
  - The EMPLOYEE, PROJECT, and WORKS_ON tables may be fragmented horizontally and stored with possible replication as shown below.

Figure 25.1
Data distribution and replication among distributed databases.



Advantages (transparency, contd.)

- Distribution and Network transparency:
  - Users do not have to worry about operational details of the network.
  - There is **Location transparency**, which refers to freedom of issuing command from any location without affecting its working.
  - Then there is **Naming transparency**, which allows access to any names object (files, relations, etc.) from any location.

DDS - Advantages (transparency, contd.)

- **Replication transparency**:
  - It allows to store copies of a data at multiple sites.
  - This is done to minimize access time to the required data.
- **Fragmentation transparency**:
  - Allows to fragment a relation horizontally (create a subset of tuples of a relation) or vertically (create a subset of columns of a relation).

DDS - Advantages (transparency, contd.)

- Increased reliability and availability:
    - **Reliability** refers to system live time, that is, system is running efficiently most of the time. Reliability is often characterized in terms of **mean time between failures** (MTBF).
    - **Availability** is the probability that the system is continuously available (usable or accessible) during a time interval. Availability is typically given as **a percentage of the time a system is expected to be available**, e.g., 99.999 percent ("five nines").
- A distributed database system has multiple nodes (computers) and if one fails then others are available to do the job.

DDS - Advantages (transparency, contd.)

- Improved performance:
    - A distributed DBMS fragments the database to keep data closer to where it is needed most.
    - This reduces data management (access and modification) time significantly.
- Easier expansion (scalability):
    - Allows new nodes (computers) to be added anytime without chaining the entire configuration.

Data Fragmentation, Replication and Allocation

- Data Fragmentation
    - Split a relation into logically related and correct parts. A relation can be fragmented in two ways:
    - Horizontal Fragmentation
    - Vertical Fragmentation

Horizontal fragmentation

- It is a horizontal subset of a relation which contain those of tuples which satisfy selection conditions.

- Consider the Employee relation with selection condition (DNO = 5). All tuples satisfy this condition will create a subset which will be a horizontal fragment of Employee relation.

- A selection condition may be composed of several conditions connected by AND or OR.

Horizontal fragmentation

- **Complete relation:**

Vno	Vname	City	Vbal
1	Sears	Toronto	200.00
2	Kmart	Ottawa	671.05
3	Eatons	Toronto	301.00
4	The Bay	Ottawa	162.99

- **Horizontally fragmented relation (two sites):**

Site 1 (Ottawa site)

Vno	Vname	City	Vbal
2	Kmart	Ottawa	671.05
4	The Bay	Ottawa	162.99

Site 2 (Toronto site)

Vno	Vname	City	Vbal
1	Sears	Toronto	200.00
3	Eatons	Toronto	301.00

Vertical fragmentation

- It is a subset of a relation which is created by a subset of columns. Thus a vertical fragment of a relation will contain values of selected columns. There is no selection condition used in vertical fragmentation.

- Consider the Employee relation. A vertical fragment of can be created by keeping the values of Name, Bdate, Sex, and Address.

- Because there is no condition for creating a vertical fragment, each fragment must include the primary key attribute of the parent relation Employee. In this way all vertical fragments of a relation are connected.

Vertical fragmentation

- Complete relation:

Vno	Vname	City	Vbal
1	Sears	Toronto	200.00
2	Kmart	Ottawa	671.05
3	Eatons	Toronto	301.00
4	The Bay	Ottawa	162.99

- Vertically fragmented relation (two sites):

Site 1

Vno	Vbal
1	200.00
2	671.05
3	301.00
4	162.99

Site 2

Vno	Vname	City
1	Sears	Toronto
2	Kmart	Ottawa
3	Eatons	Toronto
4	The Bay	Ottawa

Representation - Horizontal fragmentation

- Each horizontal fragment on a relation can be specified by a σ_{Ci} (R) operation in the relational algebra.

- Complete horizontal fragmentation: A set of horizontal fragments whose conditions C1, C2, …, Cn include all the tuples in R- that is, every tuple in R satisfies (C1 OR C2 OR … OR Cn).

- Disjoint complete horizontal fragmentation: No tuple in R satisfies (Ci AND Cj) where $i \neq j$.

Representation - Vertical fragmentation

- A vertical fragment on a relation can be specified by a $\Pi_{Li}(R)$ operation in the relational algebra.

- Complete vertical fragmentation: A set of vertical fragments whose projection lists L1, L2, …, Ln include all the attributes in R but share only the primary key of R. In this case the projection lists satisfy the following two conditions:

- L1 ∪ L2 ∪ ... ∪ Ln = ATTRS (R)

- Li ∩ Lj = PK(R) for any i j, where ATTRS (R) is the set of attributes of R and PK(R) is the primary key of R.

Data Fragmentation, Replication and Allocation

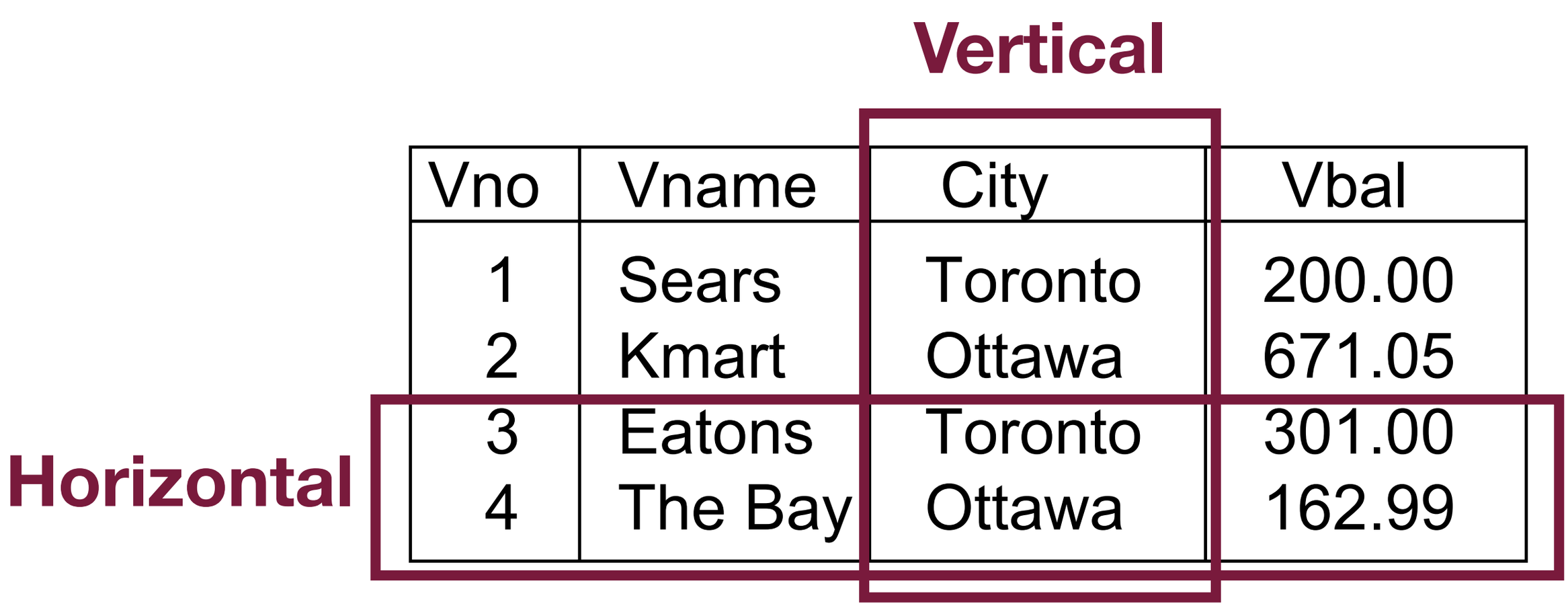
- **Fragmentation schema**
    - A definition of a set of fragments (horizontal or vertical or horizontal and vertical) that includes all attributes and tuples in the database that satisfies the condition that the whole database can be reconstructed from the fragments.
- **Allocation schema**
    - It describes the distribution of fragments to sites of distributed databases. It can be fully or partially replicated or can be partitioned.

Replication and Allocation

- Data Replication
  - Database is replicated to all sites.
  - In **full replication** the entire database is replicated and in **partial replication** some selected part is replicated to some of the sites.
  - Data replication is achieved through a **replication schema**.
- Data Distribution (Data Allocation)
  - This is relevant only in the case of partial replication or partition.
  - The selected portion of the database is distributed to the database sites.

Exercício 3

- Considere a relação R(a,b,c). Quais operações da álgebra relacional são necessárias para recompor a tabela em caso de fragmentação horizontal? E para fragmentação vertical?

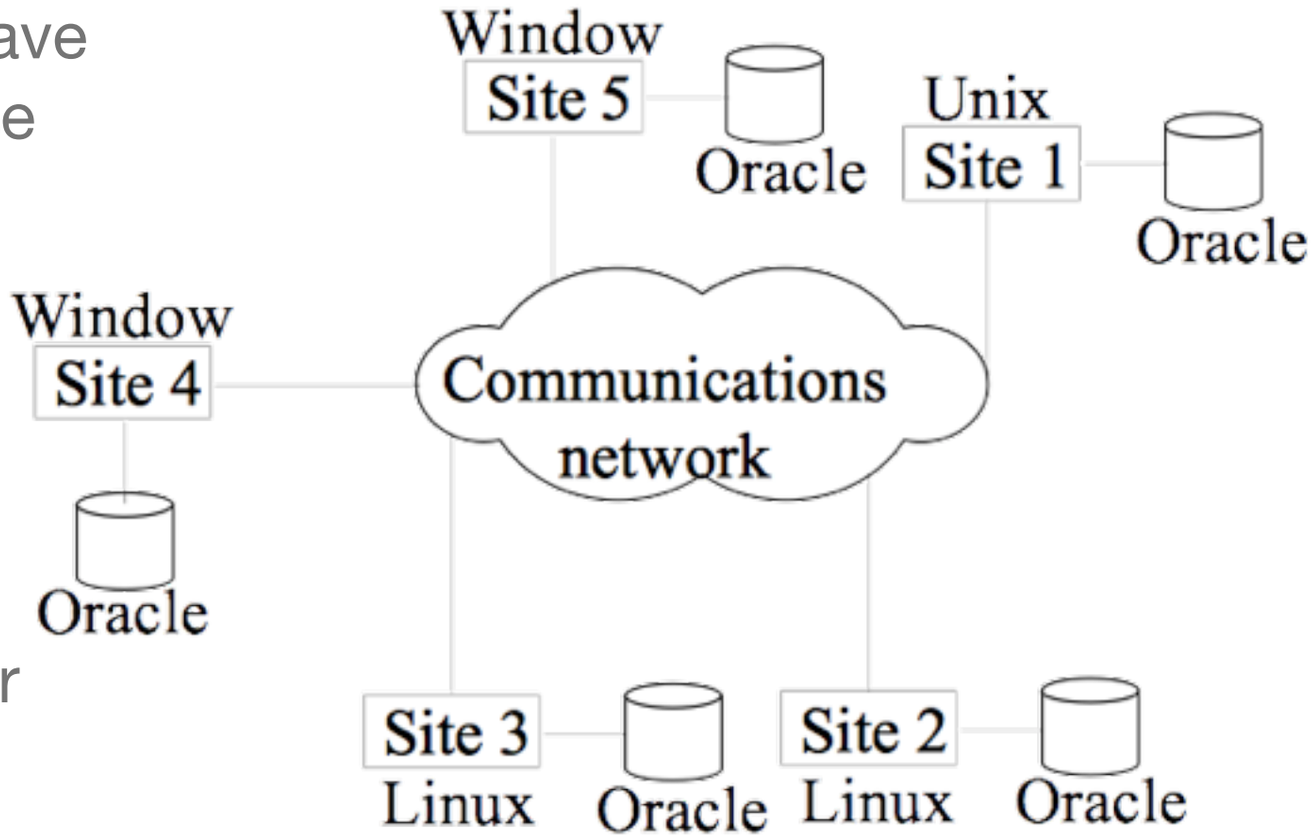


Types of Distributed Database Systems

- Homogeneous
- Heterogeneous

Homogeneous DDB

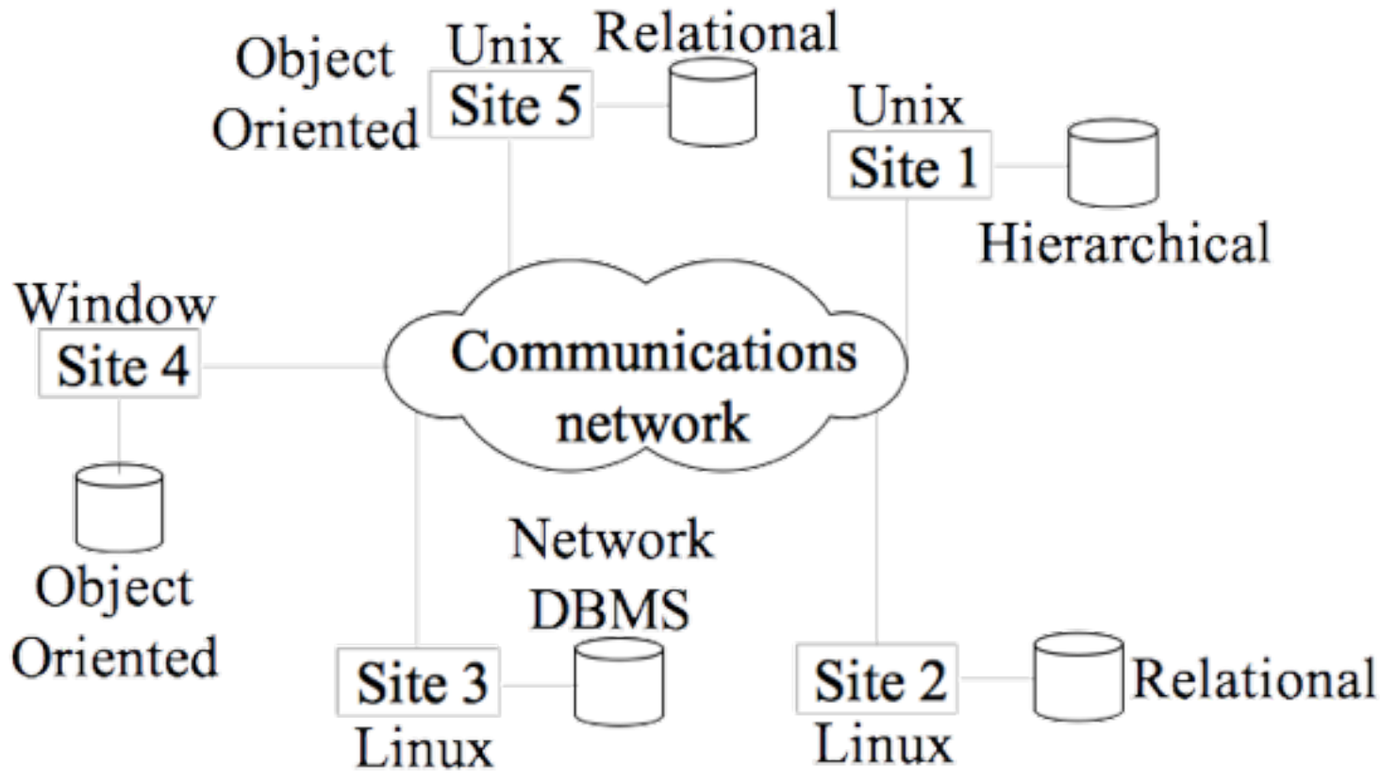
- All sites of the database system have identical setup, i.e., same database system software.
- The underlying operating system may be different.
- For example, all sites run Oracle or DB2, or Sybase or some other database system.
- The underlying operating systems can be a mixture of Linux, Window, Unix, etc.



Heterogeneous DDB

- **Federated**: Each site may run different database system but the data access is **managed through a single conceptual schema**.
    - This implies that the degree of local autonomy is minimum. Each site must adhere to a centralized access policy. There may be a global schema.

- **Multidatabase**: There is no one conceptual global schema. For data access a schema is constructed dynamically as needed by the application software.



Federated Database Management Systems Issues

- Differences in data models:
    - Relational, Objected oriented, hierarchical, network, etc.
- Differences in constraints:
    - Each site may have their own data accessing and processing constraints.
- Differences in query language:
    - Some site may use SQL, some may use SQL-89, some may use SQL-92, and so on.

Query Processing in Distributed Databases

- Cost of transferring data (files and results) over the network.
    - This cost is usually high so some optimization is necessary.
- Example relations: Employee at site 1 and Department at Site 2
- Employee at site 1. 10,000 rows. Row size = 100 bytes. Table size = 10^6 bytes.
- Department at Site 2. 100 rows. Row size = 35 bytes. Table size = 3,500 bytes.
- Q: For each employee, retrieve employee name and department name Where the employee works.

Query Processing in Distributed Databases

- The result of this query will have 10,000 tuples, assuming that every employee is related to a department.

- Suppose each result tuple is 40 bytes long. The query is submitted at site 3 and the result is sent to this site.

- Problem: Employee and Department relations are not present at site 3.

Strategies

- Transfer Employee and Department to site 3.
    - Total transfer bytes = 1,000,000 + 3500 = 1,003,500 bytes.
- Transfer Employee to site 2, execute join at site 2 and send the result to site 3.
    - Query result size = 40 * 10,000 = 400,000 bytes. Total transfer size = 400,000 + 1,000,000 = 1,400,000 bytes.
- Transfer Department relation to site 1, execute the join at site 1, and send the result to site 3.
    - Total bytes transferred = 400,000 + 3500 = 403,500 bytes.

Concurrency Control and Recovery

- Dealing with multiple copies of data items
- Failure of individual sites
- Communication link failure
- Distributed commit
- Distributed deadlock

Parallel vs distributed servers

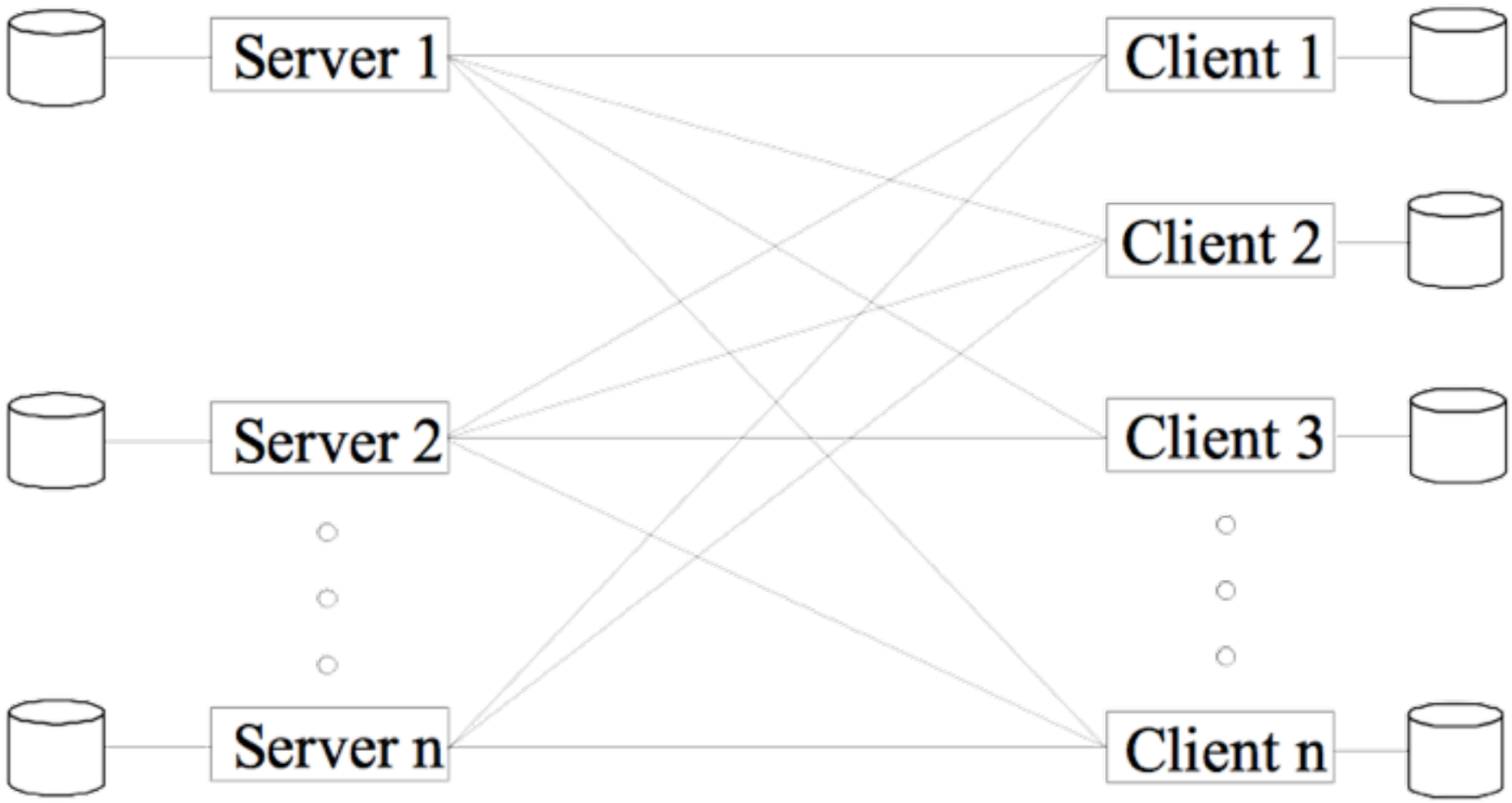
- parallel database server:
  - servers in physical proximity to each other
  - fast, high-bandwidth communication between servers, usually via a LAN
  - most queries processed cooperatively by all servers
- distributed database server:
  - servers may be widely separated
  - server-to-server communication may be slower, possibly via a WAN
  - queries often processed by a single server

Client-Server Database Architecture

Client-Server Database Architecture

- It consists of clients running client software, a set of servers which provide all database functionalities and a reliable communication infrastructure.

- 3-Tier Architecture



Client-Server Database Architecture

- Clients reach server for desired service, but server does reach clients.
- The server software is responsible for local data management at a site, much like centralized DBMS software.
- The client software is responsible for most of the distribution function.
- The communication software manages communication among clients and servers.

Processing of SQL queries

- Client parses a user query and decomposes it into a number of independent sub-queries. Each subquery is sent to appropriate site for execution.
- Each server processes its query and sends the result to the client.
- The client combines the results of subqueries and produces the final result.

Alternative Database Architectures

Hardware-based optimizations

- **In-Memory Databases**
- SSD Databases
- GPU Databases
- **IBM SyNAPSE Chips**
- **Crowdsourced Databases**

In-Memory Databases

- Becoming popular as RAM prices drop
- Offered by main vendors (MySQL offers in-memory storage engine)
- Durability (ACID) support?

In-Memory Databases - Durability

- Snapshot files: generated periodically - may lose recent information
- Transaction logging: as in RDBMS - disk may be bottleneck
- Non-Volatile DIMM: more expensive
- Non-volatile random access memory: usually RAM backed up with battery power
- Database replication

Crowdsourced Databases

- For tasks that are hard for computers to process
- e.g. interpreting images
- Uses crowdsourcing infrastructures such as Amazon Mechanical Turk

Crowdsourced Databases

- SELECT * FROM images where isFlower(img)

```
TASK isFlower(Image img) RETURN BOOL:
TaskType: Question
Text: ``Does this image: <img src=`%s'>
contain a flower?'',URLify(img)
Response: Choice(``YES'',``NO'')
```

IBM SyNAPSE Chips

- Systems of Neuromorphic Adaptive Plastic Scalable Electronics
- Capable of adapting the connection strength between two neurons in a manner analogous to that seen in biological systems
- Low power consumption
- Fast for parallel association tasks (learning, pattern matching, etc)
- Primary target: autonomous robots
- Databases?

Aplicação no trabalho

- DBs em memória (MySQL). Comparação de tempo de consulta. Cold vs hot runs.
- Infraestruturas distribuídas (próxima aula).
- Implementação em CUDA (linguagem da NVIDIA para GPUs).
- . . .

Respostas - Exercícios de BDs Distribuídos e Paralelos

Luiz Celso Gomes Jr - André Santanchè

MC536 2013/2

Exercício 1

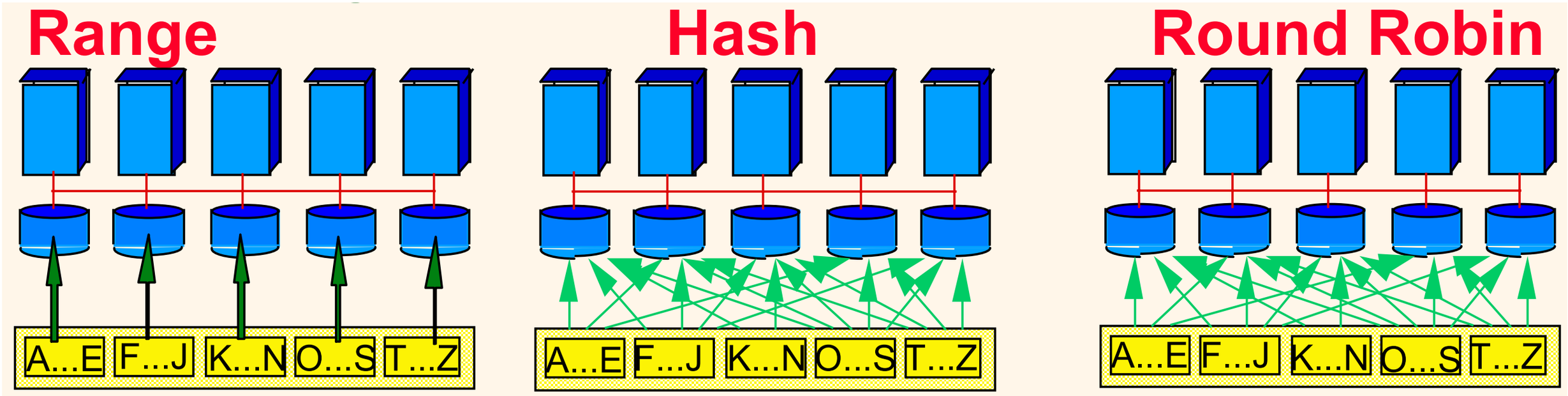
- [Preliminares] Suponha que a DAC esteja enfrentando problemas para atender as consultas online de CR dos alunos (o tempo de resposta é muito longo). As tabelas do banco são descritas abaixo. Quais técnicas (ao menos duas) vocês poderiam aplicar para melhorar o desempenho das consultas?

- Aluno(RA, nome, curso)
- Disciplina(codigo, nome)
- Cursa(RA, codigo, nota)

Resposta: Índice de Hash na tabela aluno(RA) e Cursa(RA). Denormalização, calculando o CR sempre no fim de dia (ou alguma outra periodicidade). etc...

Exercício 2

- Ordene os tipos de técnica de particionamento de dados (Range, Hash, Round Robin) de acordo com o tamanho físico dos índices que precisam ser mantidos para localizar o disco ou CPU que contém cada tupla. Justifique sua resposta.



Resposta:

Hash, Range, Round Robin. Hash nao precisa de indice. Range so precisa armazenar os limites. RR precisa registrar todas as tuplas.

Exercício 3

- Considere a relação R(a,b,c). Quais operações da álgebra relacional são necessárias para recompor a tabela em caso de fragmentação horizontal? E para fragmentação vertical?

Vertical

Horizontal

Vno	Vname	City	Vbal
1	Sears	Toronto	200.00
2	Kmart	Ottawa	671.05
3	Eatons	Toronto	301.00
4	The Bay	Ottawa	162.99

Horizontal: Union

Vertical: Outer Join (natural)