Distributed Databases

And their relevance for NOSQL

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

NOSQL Goals

Recall the NOSQL Goals:

- Schemaless: Allow flexible (even runtime) schema definition [data structure]
- Reliability / availability: Keep delivering service even if its software or hardware components fail [recovery]
- Scalability: Continuously evolve to support a growing amount of tasks [distribution]
- Efficiency: How well the system performs, usually measured in terms of response *time* (latency) and *throughput* (bandwith) [distribution]

NOSQL Goals

Recall the NOSQL Goals:

- Schemaless: Allow flexible (even runtime) schema definition [data structure]
- Reliability / availability: Keep delivering service even if its software or hardware components fail [distribution]
- Scalability: Continuously evolve to support a growing amount of tasks [distribution]
- Efficiency: How well the system performs, usually measured in terms of response *time* (latency) and *throughput* (bandwith) [distribution]

NOSQL Goals

- Reliability / availability: Keep delivering service even if its software or hardware components fail [distribution]
- Scalability: Continuously evolve to support a growing amount of tasks [distribution]
- Efficiency: How well the system performs, usually measured in terms of response *time* (latency) and *throughput* (bandwith) [distribution]

NOSQL Goals

- Reliability / availability: Keep delivering service even if its software or hardware components fail [distribution]
- Scalability: Continuously evolve to support a growing amount of tasks [distribution]
- Efficiency: How well the system performs, usually measured in terms of response *time* (latency) and *throughput* (bandwith) [distribution]

Most NOSQL goals can be achieved by means of distribution!

Activity: Why Distribution?

- *Objective: Recognize the benefits of distributing data*
- *Tasks:*
  1. *(15') By pairs, answer the following questions:*
     a) *How long would it take to read 1TB with sequential access (fig. a)? (in secs)*
        - *Can you identify any additional drawback to be considered?*

     b) *How long would it take to read 1TB with parallel access (fig. b)? Assume 100 disks on the same machine with shared-memory and infinite CPU capacity.*
        - *Can you identify any additional drawback to be considered?*

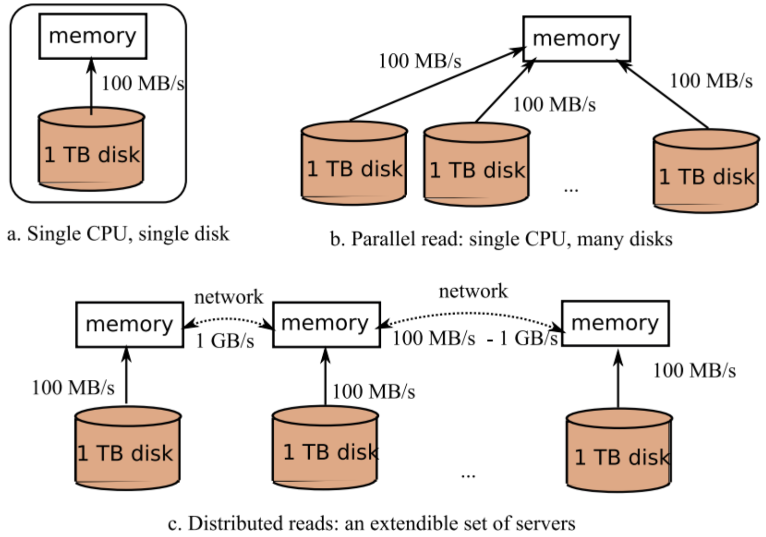
     c) *How long would it take to read 1TB with distributed access (fig. c)? Assume 100 shared-nothing machines in a star-shape LAN in a single rack where all data is sent to the centre.*
        - *Can you identify any additional drawback to be considered?*

     d) *Now, repeat the exercise considering a single random access. What changes?*
  2. *(5') Discussion*

Type	Latency	Bandwidth
Disk	$\approx 5 \times 10^{-3}$s (5 millisec.);	At best 100 MB/s
LAN	$\approx 1 - 2 \times 10^{-3}$s (1-2 millisec.);	$\approx$ 1GB/s (single rack); $\approx$ 100MB/s (switched);
Internet	Highly variable. Typ. 10-100 ms.;	Highly variable. Typ. a few MB/s.;

Bottom line (1): it is approx. one order of magnitude faster to exchange main memory data between 2 machines in a data center, that to read on the disk.

Bottom line (2): exchanging through the Internet is slow and unreliable with respect to LANs.



a. Single CPU, single disk

b. Parallel read: single CPU, many disks

c. Distributed reads: an extendible set of servers

Activity: Why Distribution?

- *Objective: Recognize the benefits of distributing data*
- *Tasks:*
  1. *(15') By pairs, answer the following questions:*
     a) *How long would it take to read 1TB with sequential access (fig. a)? (in secs)*
        - *Can you identify any additional drawback to be considered?*

     b) *How long would it take to read 1TB with parallel access (fig. b)? Assume 100 disks on the same machine with shared-memory and infinite CPU capacity.*
        - *Can you identify any additional drawback to be considered?*

     c) *How long would it take to read 1TB with distributed access (fig. c)? Assume 100 shared-nothing machines in a star-shape LAN in a single rack where all data is sent to the centre.*
        - *Can you identify any additional drawback to be considered?*

     d) *Now, repeat the exercise considering a single random access. What changes?*
  2. *(5') Discussion*

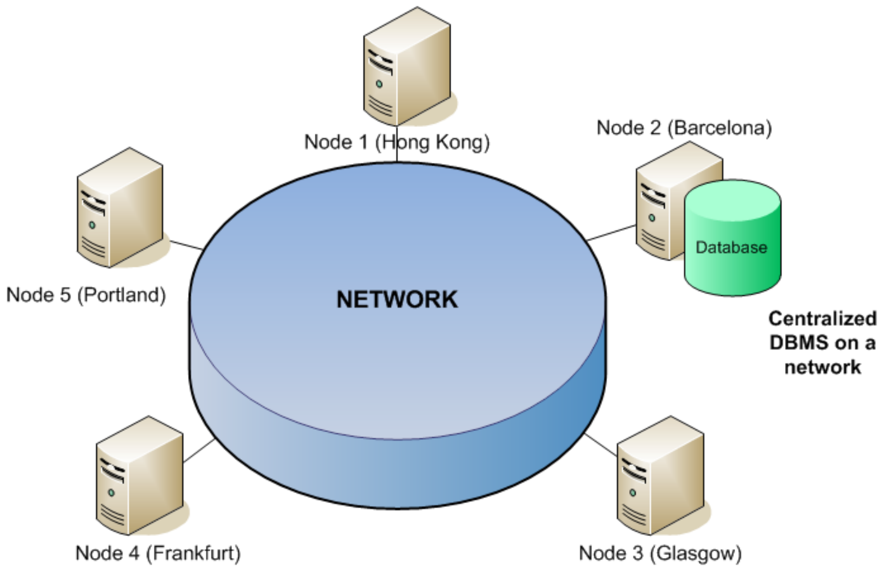
Conclusions:

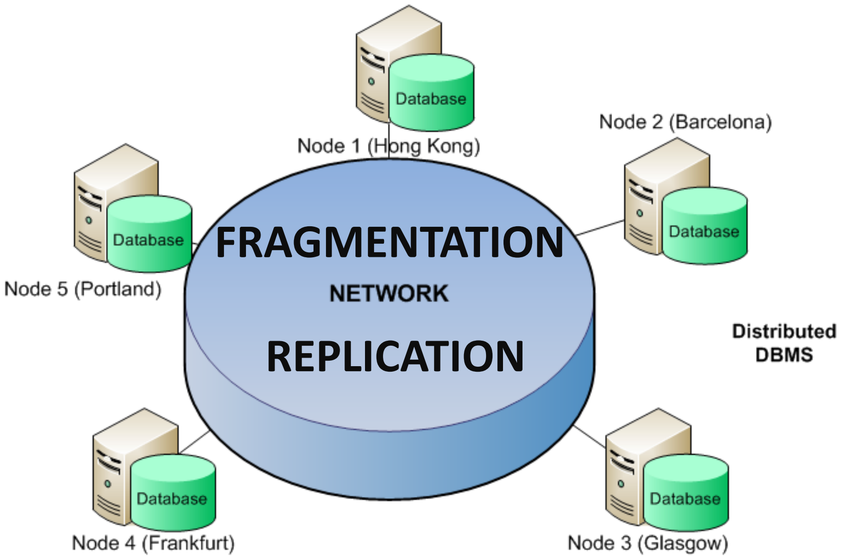
- Disk transfer rate is a bottleneck for batch processing of large scale datasets; parallelization and distribution can help to eliminate this bottleneck
- Disk seek time is a bottleneck for transactional applications that submit a high rate of random accesses; replication, distribution of writes and distribution of reads make such applications scalable
- When possible, data should be accessed where they are stored (or near) to avoid costly data exchange over the network

Definition and Distributed Architectures

Distributed Database

- A distributed database (DDB) is a database where data management is distributed over several nodes in a network.
  - Each node is a database itself
    - Potential heterogeneity
  - Nodes communicate through the network





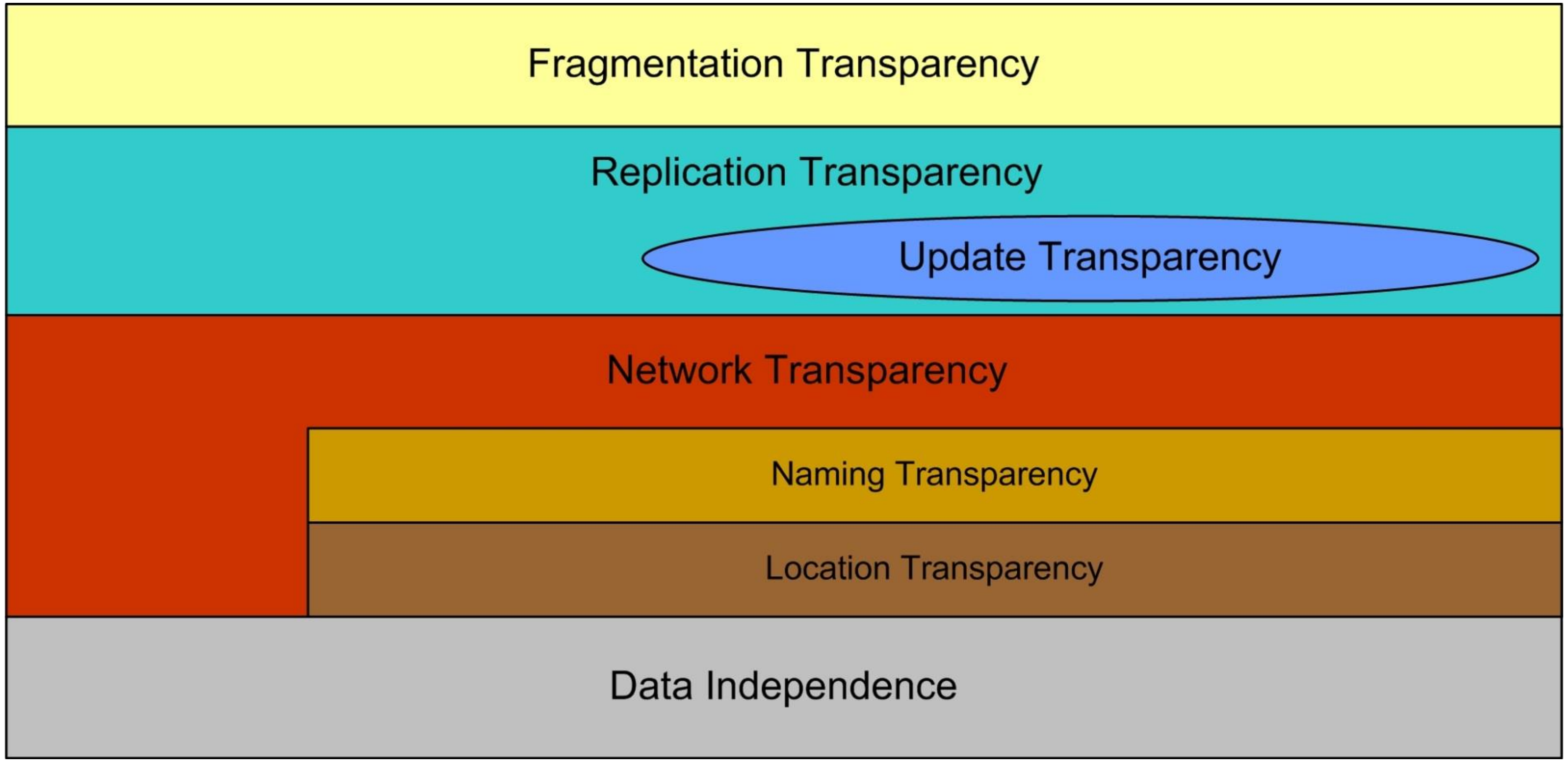
Distributed Architectures

- Main objective: hide implementation (i.e., physical) details to the users who should not know they are dealing with a distributed system
  - Network transparency
    - Data access must be independent regardless where data is stored
    - Each data object must have a unique name
  - Replication transparency
    - The user must not be aware of the existing replicas
  - Fragmentation transparency
    - The user must not be aware of partitioning

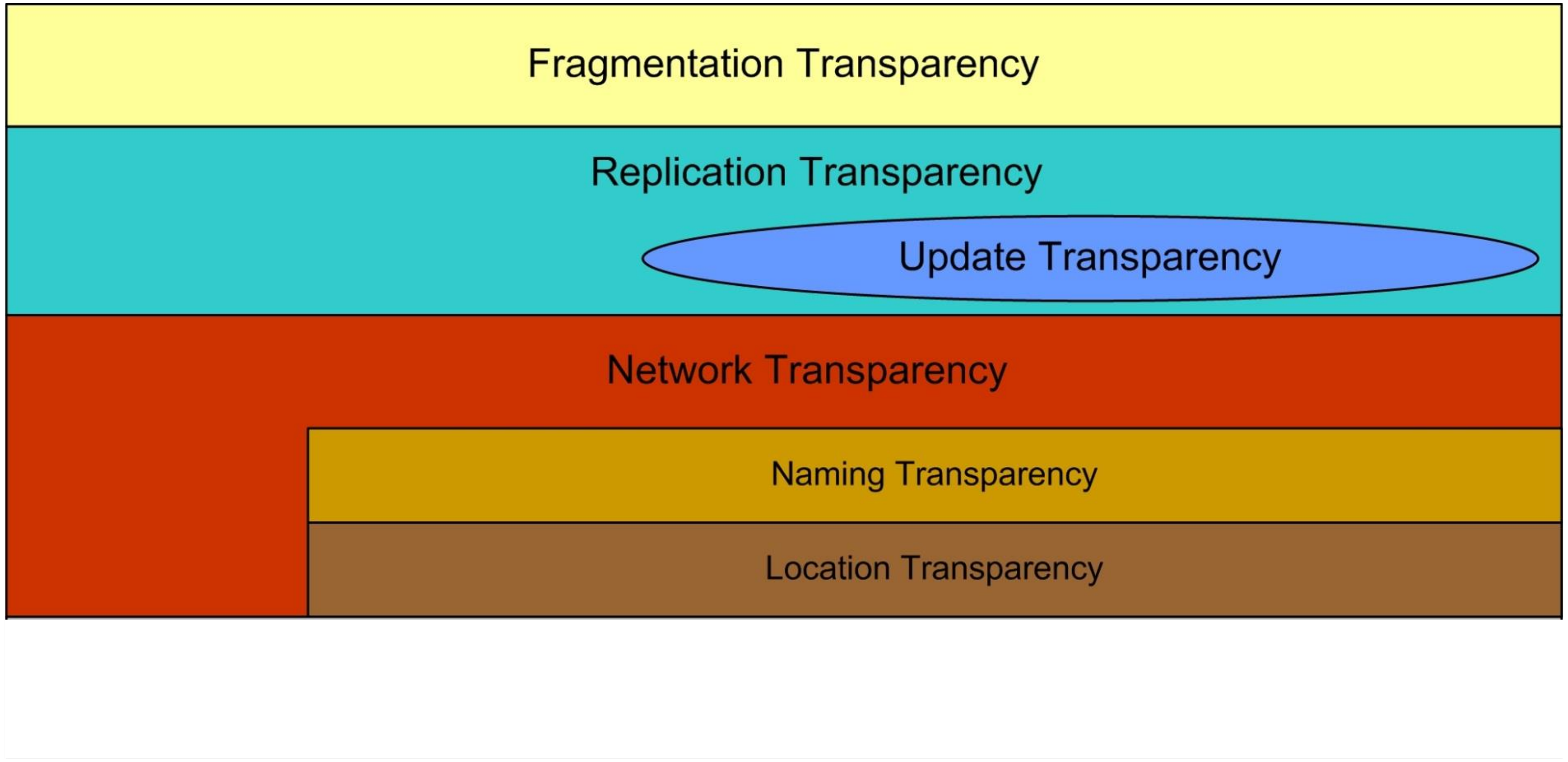
[**Desiderata**: Data independency at the logical and physical level must be guaranteed to avoid accessing directly to physical structures]

- As seen in centralized DBs (ANSI SPARC)

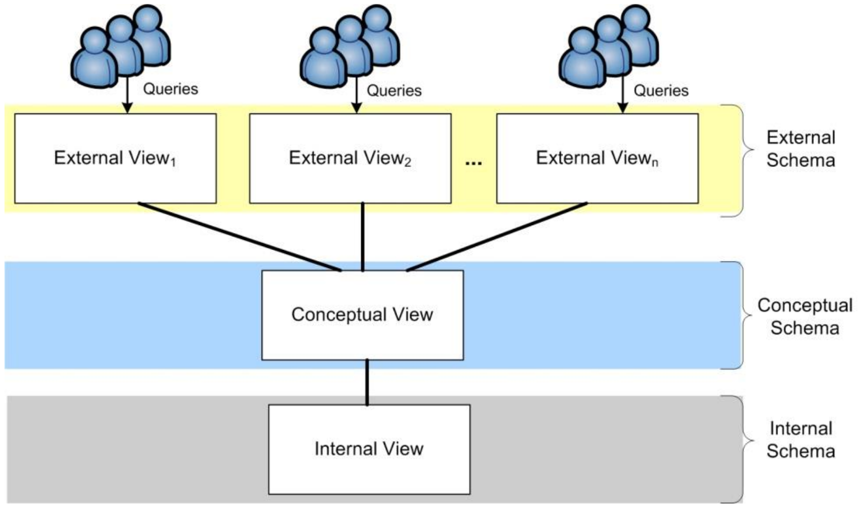
Distributed Architectures

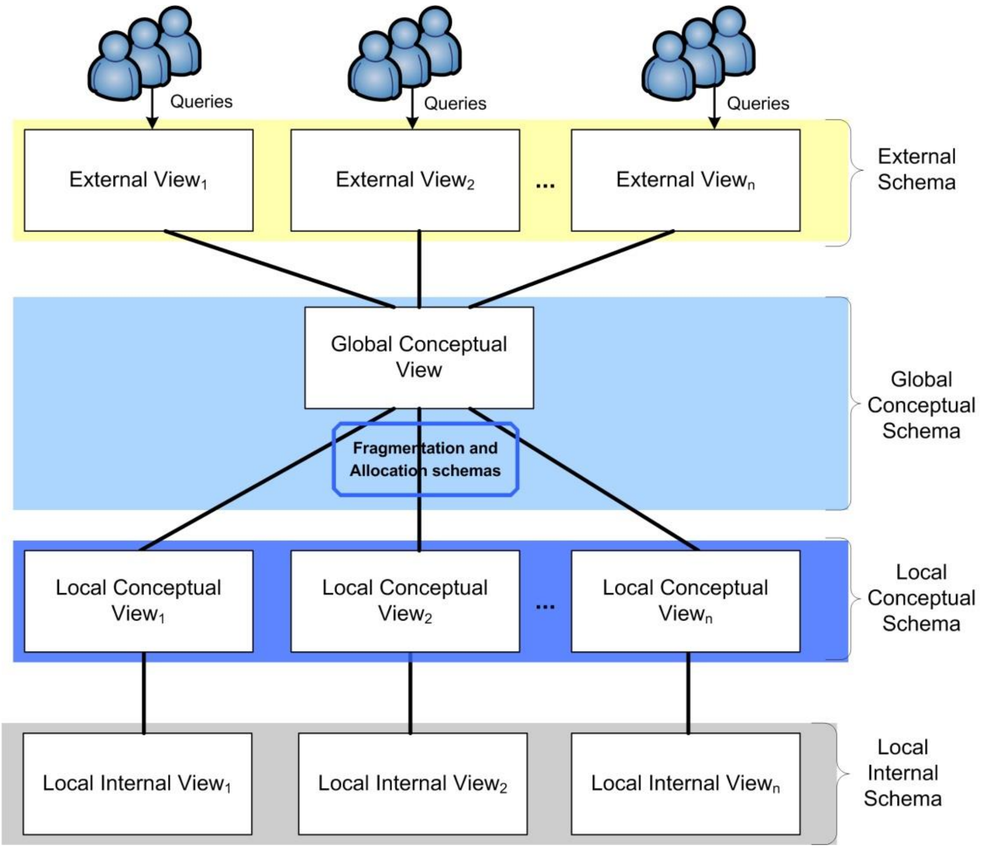


Distributed Architectures



Extended ANSI-SPARC Architecture





- Global catalog
  - Mappings between ESs – GCS and GCS – LCSs
- Each node has a local catalog
  - Mappings between LCS_i – IS_i

Distributed DBMS Architecture

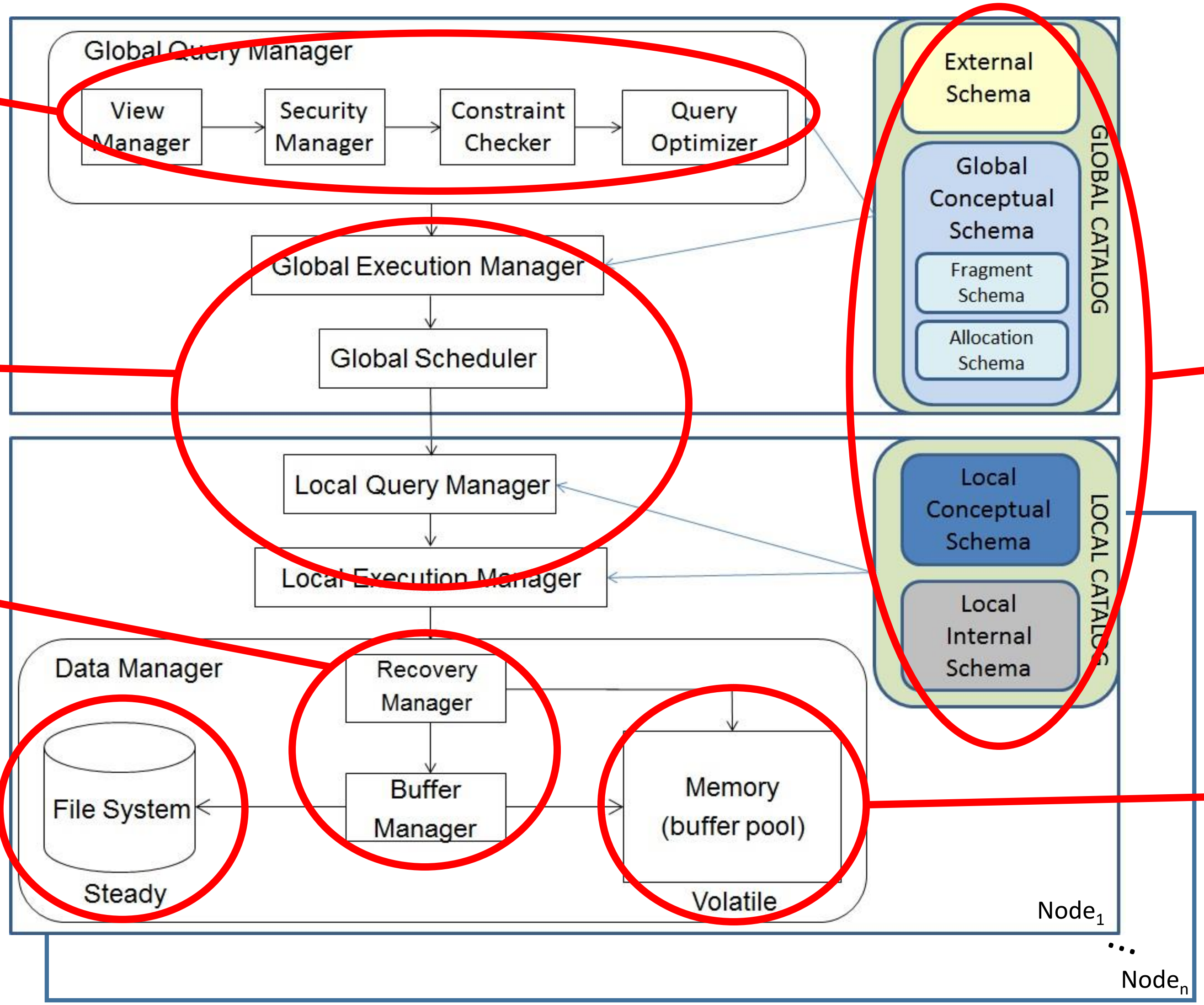
A distributed database has one global layer but many local layers (or nodes). Each node stores and process data locally

Query Manager: unfolds views, checks permissions and constraints and generates (and optimizes) a physical access plan (procedural) to data in disk from the declarative SQL query

Concurrency manager and scheduler: guarantees CI (from ACID) and schedules the execution of subqueries over the distributed fragments. They are responsible for briding between the global layer and each node

Recovery manager: interacts with the buffer and memory to guarantee AD (from ACID)

Disk: physical data storage (persistence)



Global and local catalog: needed to make distribution transparent to the end-user

Main memory: memory assigned to the database in that node

Challenges

Principles of Distributed Databases

Challenges in Distributed Databases

I. Distributed DB design
- Node distribution
- Data fragments
- Data allocation (replication)

II. Distributed DB catalog
- Fragmentation trade-off: Where to place the DB catalog
  - Global or local for each node
  - Centralized in a single node or distributed
  - Single-copy vs. Multi-copy

III. Distributed query processing
- Data distribution / replication
- Communication overhead

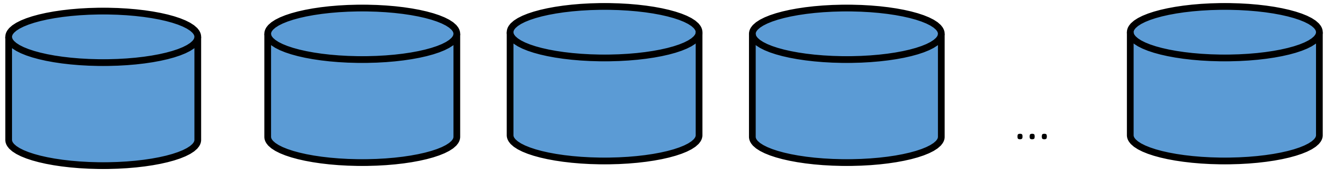
IV. Distributed transaction management
- How to enforce the ACID properties
  - Replication trade-off: Queries vs. Data consistency between replicas (updates)
  - Distributed recovery system
  - Distributed concurrency control system

V. Security issues
- Network security

Challenges in Data Distribution

A	B	C	D	E

Conceptual View



Physical View

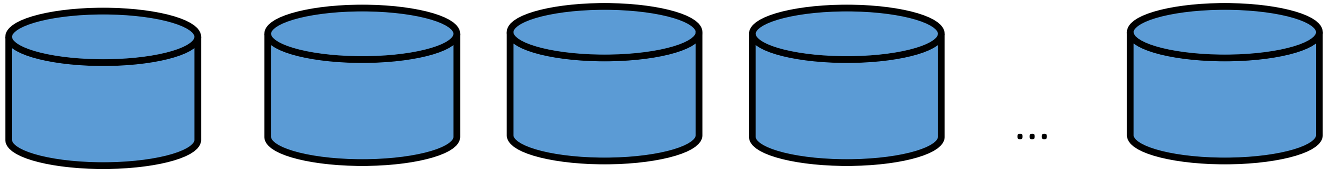
Challenges in Data Distribution

Conceptual View

A	B	C	D	E

?

How to physically store this table in a distributed manner (making it transparent for the user)?



Physical View

Challenge I

Database Design: Fragmentation and Allocation (Replication)

Challenge I: DDB Design

- Given a DB and its workload, how should the DB be split and allocated to sites as to optimize certain objective functions
  - For example, minimize resource consumption for query processing
- Two main issues:
  - Data fragmentation
  - Data allocation (data replication)

Activity: Fragmentation Strategies

- *Objective: Understand the principles behind fragmentation*
- *Tasks:*
  1. *(10') By pairs, answer the following questions:*
     1. *Briefly explain which fragmentation strategy has been applied for the database below.*
     2. *Is this fragmentation strategy complete and disjoint? Can we reconstruct the global relations? Explicit the algebraic operation you would use.*
  2. *(5') Discussion*

Global Relations

```
Kids(kidId, name, address, age)

Toys(toyId, name, price)

Requests(kidId, toyId, willingness)
```

`Note that requests(kidId) is a foreign key to kids(kidId) and similarly, requests(toyId) refers to toys(toyId).`

Fragments

```
K1= Kids[kidId, name]

K2= Kids[kidId, address, age]
```

```
T1= Toys(price >= 150)

T2= Toys(price < 150)
```

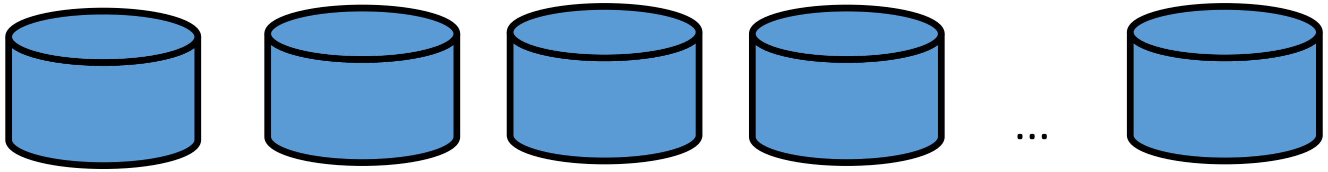
```
R1 = Requests ⋉ T1

R2 = Requests ⋉ T2
```

Challenge I: Data Fragments

Conceptual View

A	B	C	D	E



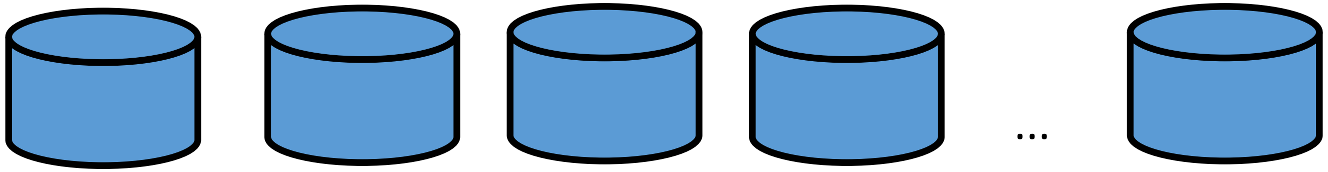
Physical View

Challenge I: Data Fragments

Conceptual View

A	B	C	D	E

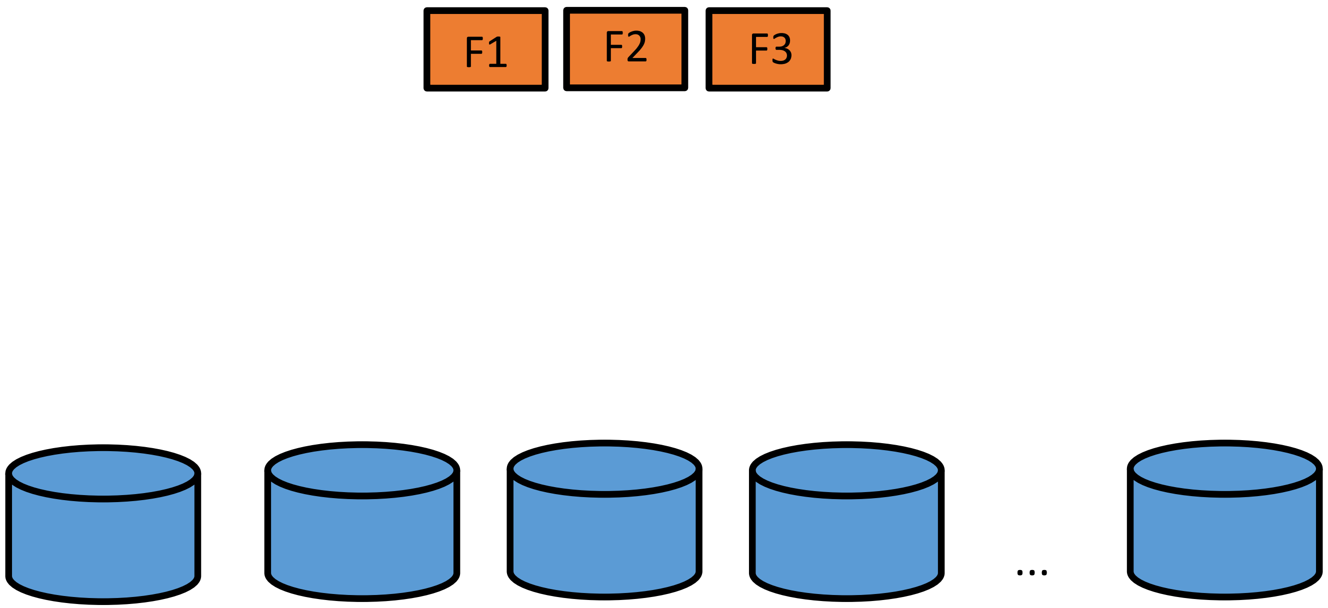
Apply a fragmentation strategy



Physical View

Challenge I: Data Fragments

Conceptual View



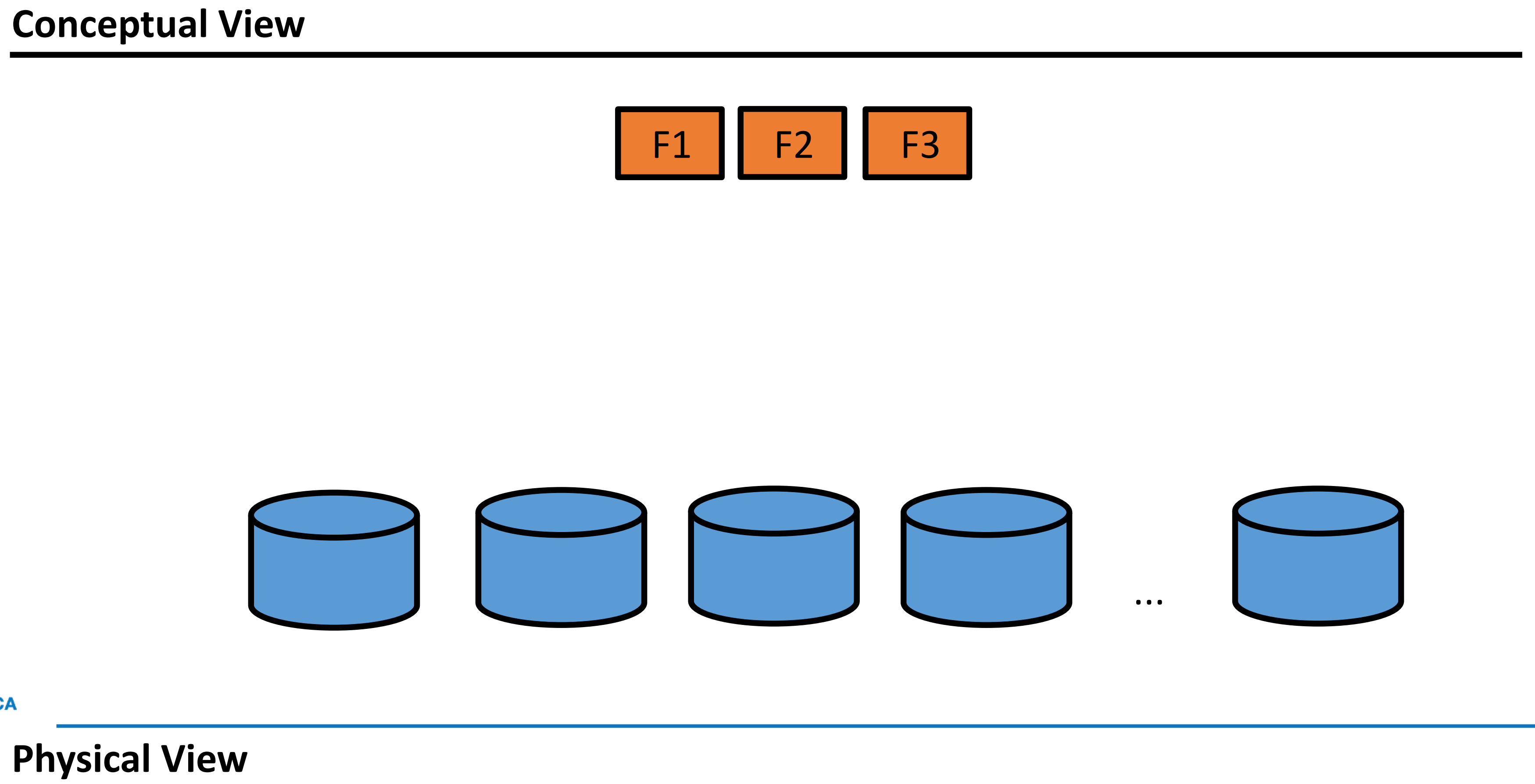
Secondary servers

Physical View

Challenge I: Data Allocation

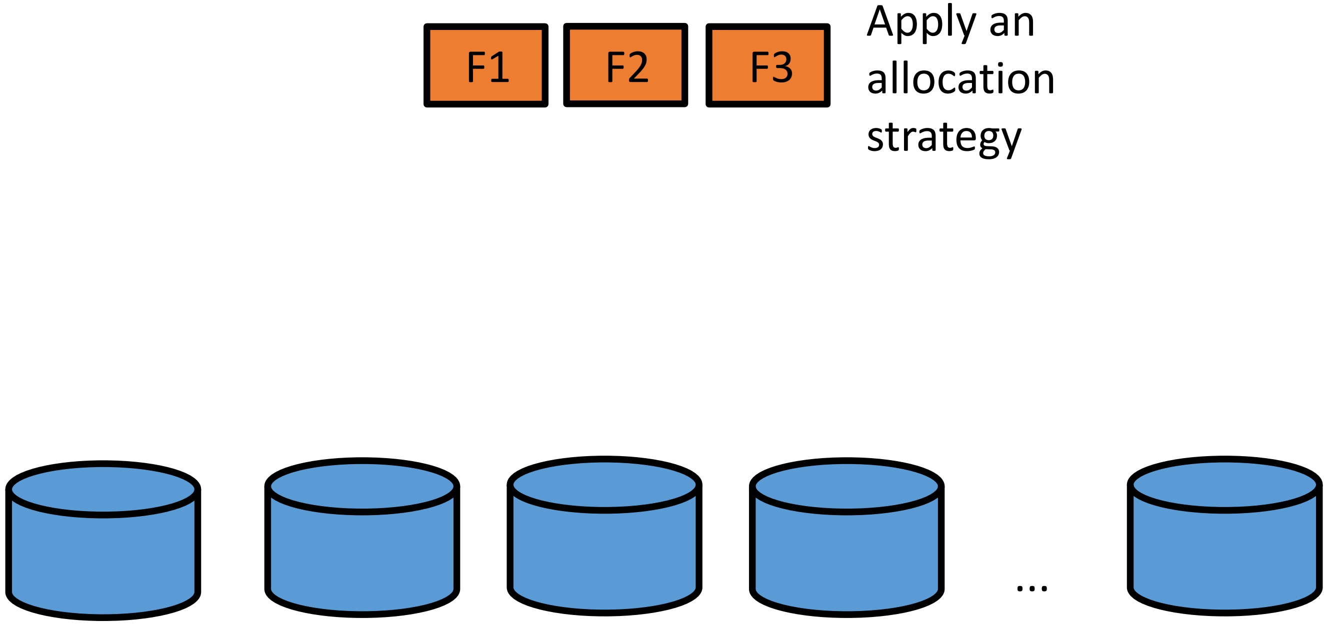
- Given a set of fragments, a set of sites on which a number of applications are running, **allocate** each fragment such that some optimization criterion is met (subject to certain constraints)
- It is known to be a NP-hard problem
  - The optimal solution depends on many factors
    - Location in which the query originates
    - The query processing strategies (e.g., join methods)
  - Furthermore, in a dynamic environment the workload and access pattern may change
- Most advanced approaches build *cost models* and any optimization algorithm can be adapted to solve it
  - Simply the problema with assumptions (e.g., only communication cost considered)
  - Heuristics are also available: (e.g., best-fit for non-replicated fragments)
  - Sub-optimal solutions (i.e., applied per fragment individually)

Challenge I: Data Allocation



Challenge I: Data Allocation

Conceptual View

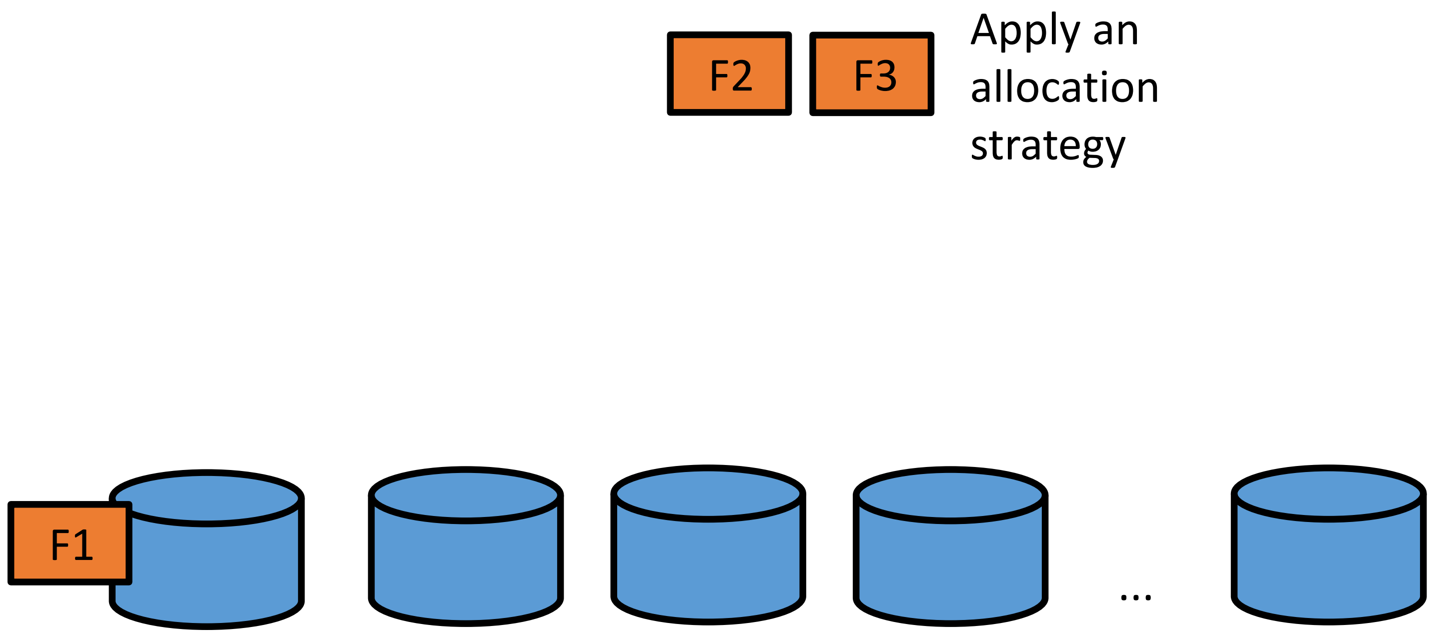


Apply an allocation strategy

Physical View

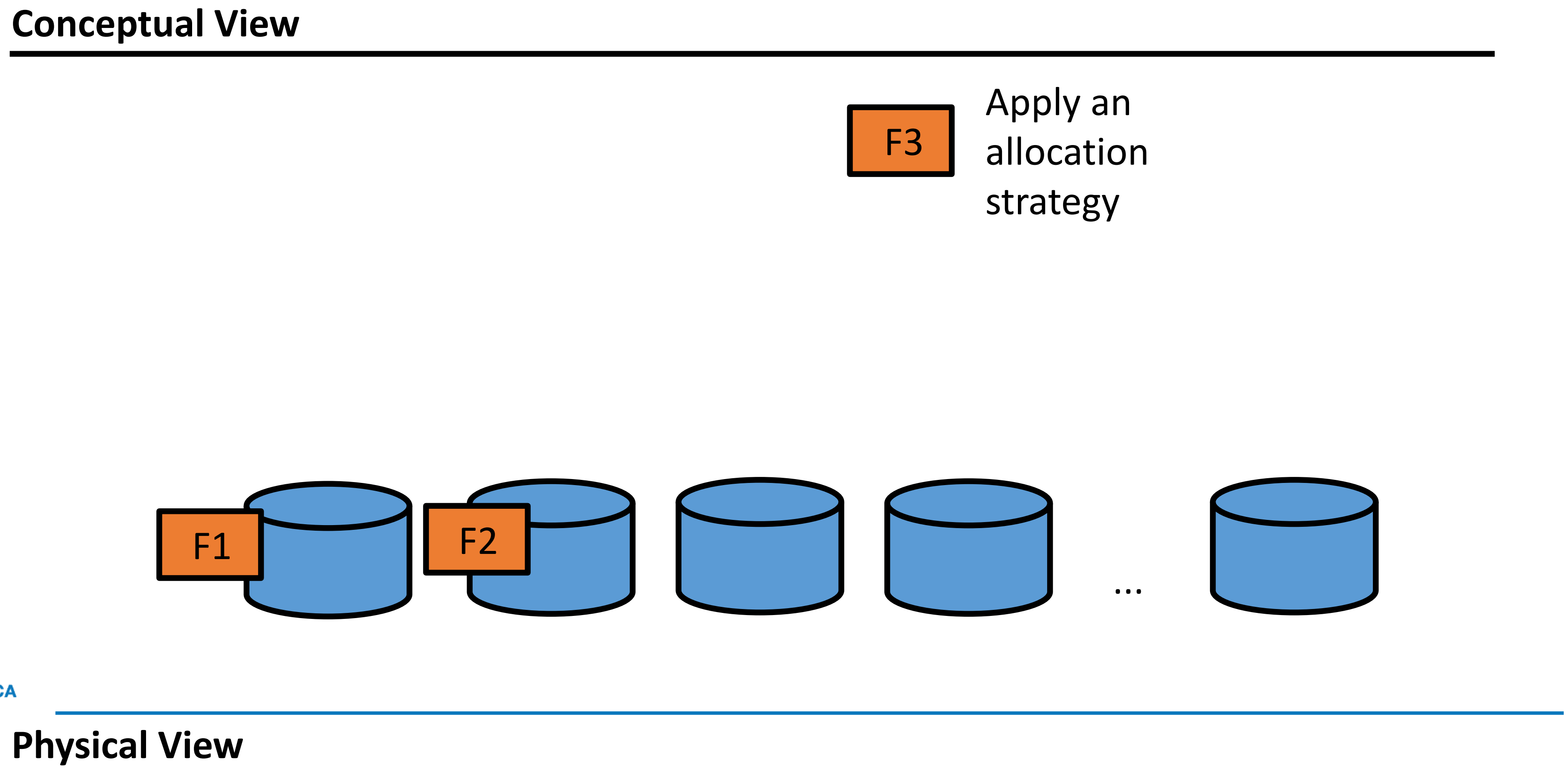
Challenge I: Data Allocation

Conceptual View



Physical View

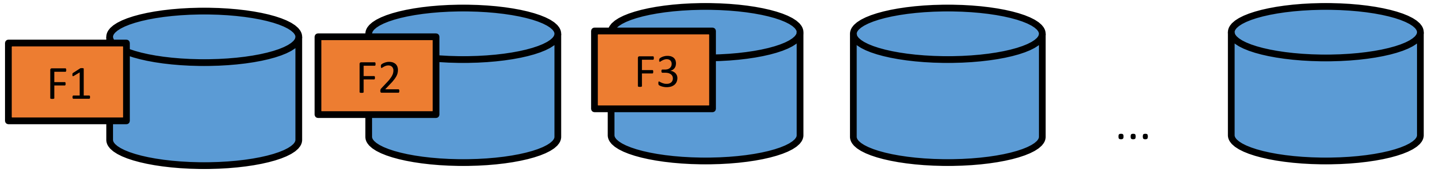
Challenge I: Data Allocation



Challenge I: Data Allocation

Conceptual View

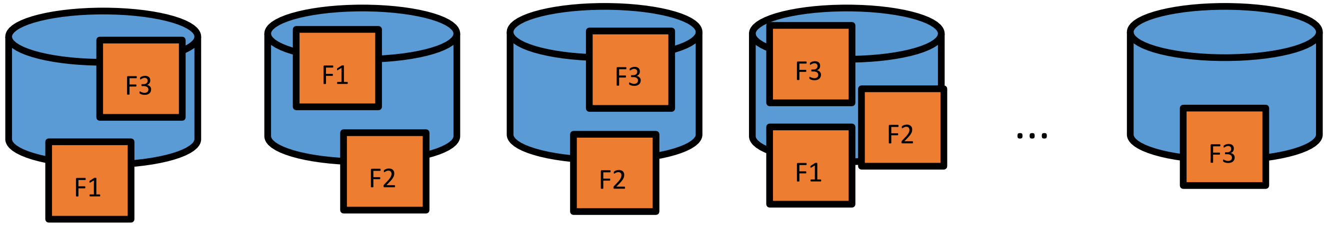
Apply an allocation strategy



Physical View

Challenge I: Data Allocation

Conceptual View

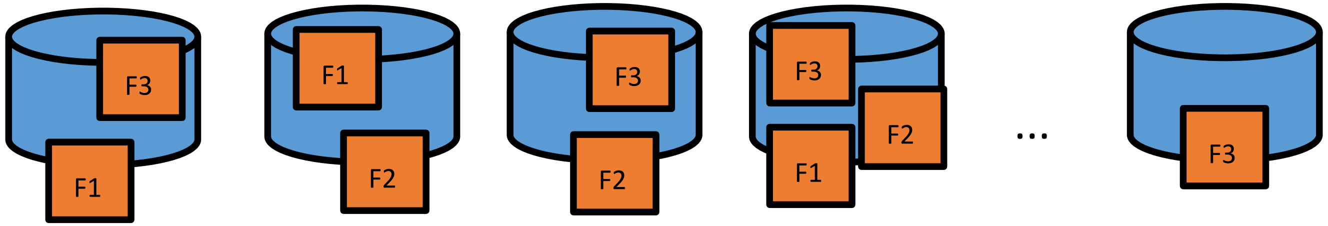


Physical View

Challenge I: Data Allocation

Conceptual View

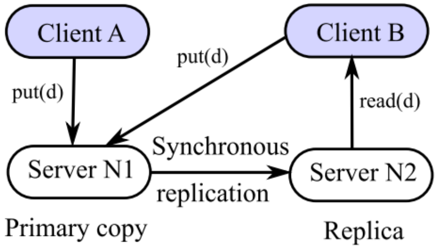
If a fragment is placed in more than one server, then, we are **replicating** it



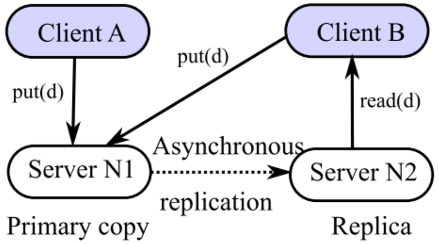
Physical View

Challenge I: Data Replication Management

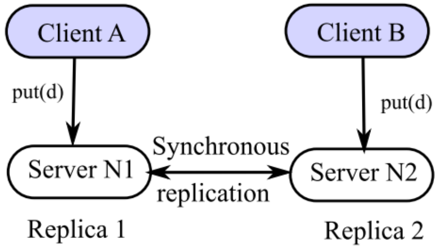
- Replicating fragments improves the system throughput but raises some other issues:
  - Consistency
  - Update performance
- Most used replication protocols
  - Eager – Lazy replication
  - Primary – Secondary versioning



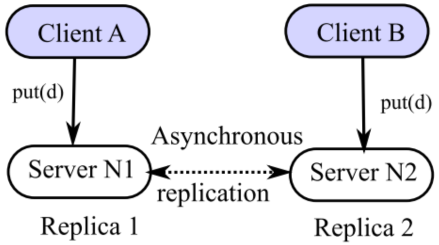
a) Eager replication with primary copy



b) Lazy replication with primary copy (a.k.a Master-Slave replication)



c) Eager replication, distributed

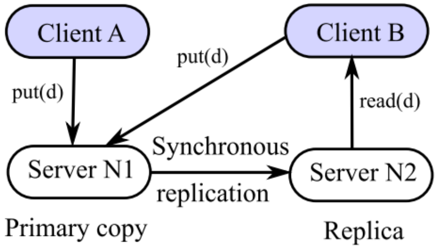


d) Lazy replication, distributed (a.k.a. Master-Master replication)

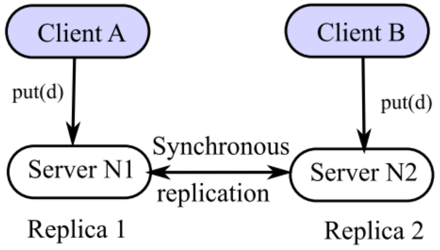
Challenge I: Data Replication Management

- Replicating fragments improves the system throughput but raises some other issues:
  - Consistency
  - Update performance
- Most used replication protocols
  - Eager – Lazy replication
  - Primary – Secondary versioning

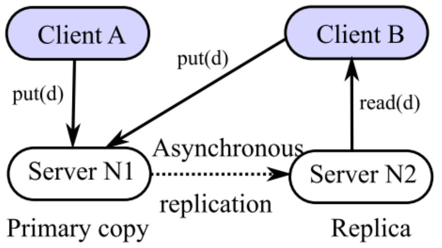
Strong Consistency



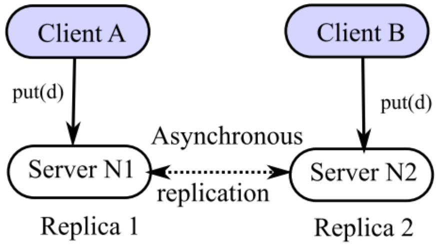
a) Eager replication with primary copy



c) Eager replication, distributed



b) Lazy replication with primary copy (a.k.a Master-Slave replication)



d) Lazy replication, distributed (a.k.a. Master-Master replication)

Eventually Consistent

Activity: Data Replication Issues

- *Objective: Understand the consequences behind each data replication strategy*
- *Tasks:*
  1. *(10') By pairs, answer the following questions:*
     1. *Discuss the questions below with your peer*
     2. *What is the most important feature for each scenario?*
  2. *(5') Discussion*

- You are a customer using an e-commerce based on heavy replication (e.g., Amazon):
  a) Show a database replication strategy (e.g., sketch it) where:
     1. You buy an item, but this item does not appear in your basket.
     2. You reload the page: the item appears.

     What happened?

  b) Show a database replication strategy (e.g., sketch it) where:
     1. You delete an item from your command, and add another one: the basket shows both items.

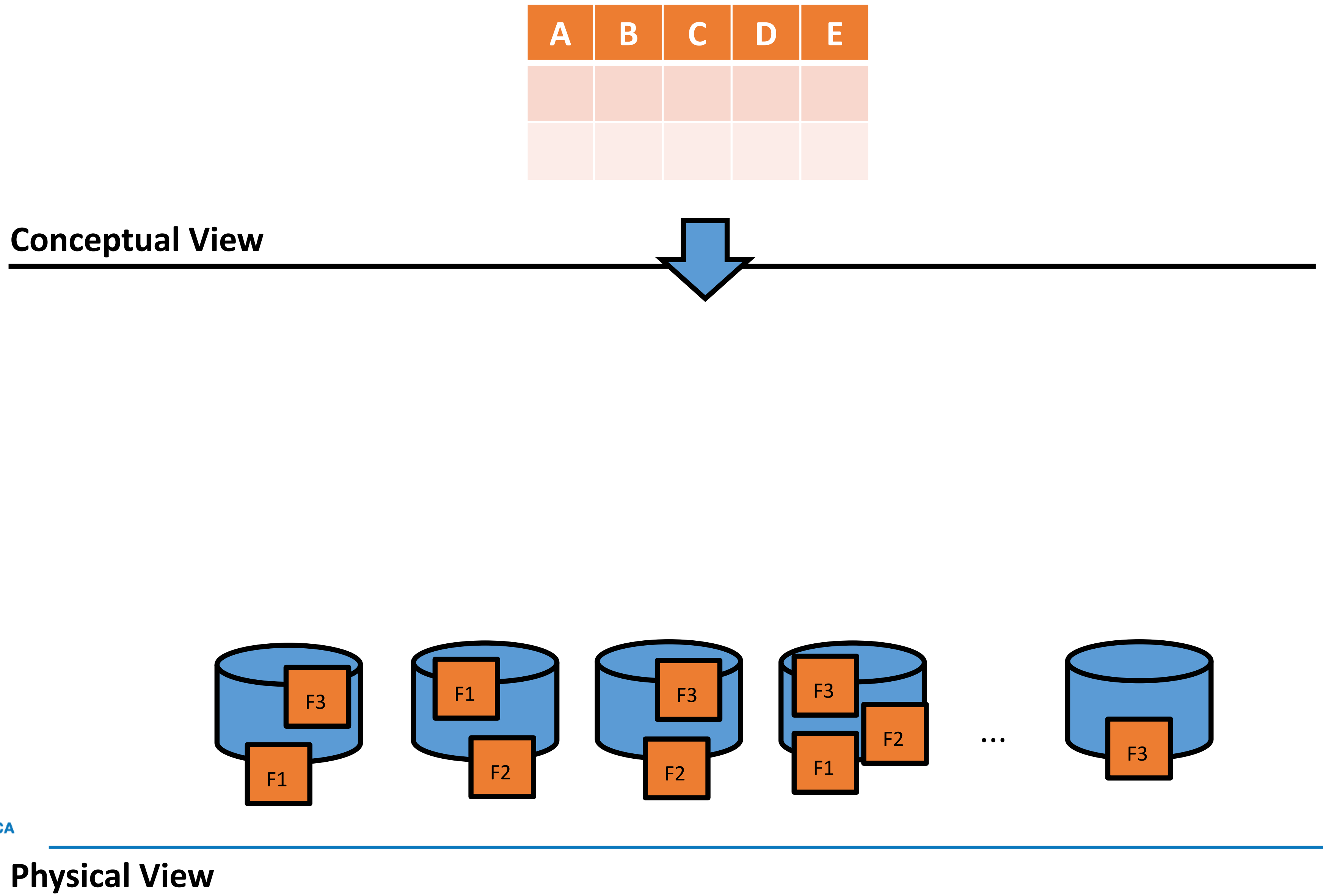
     What happened?

     Will the situation change if you reload the page?

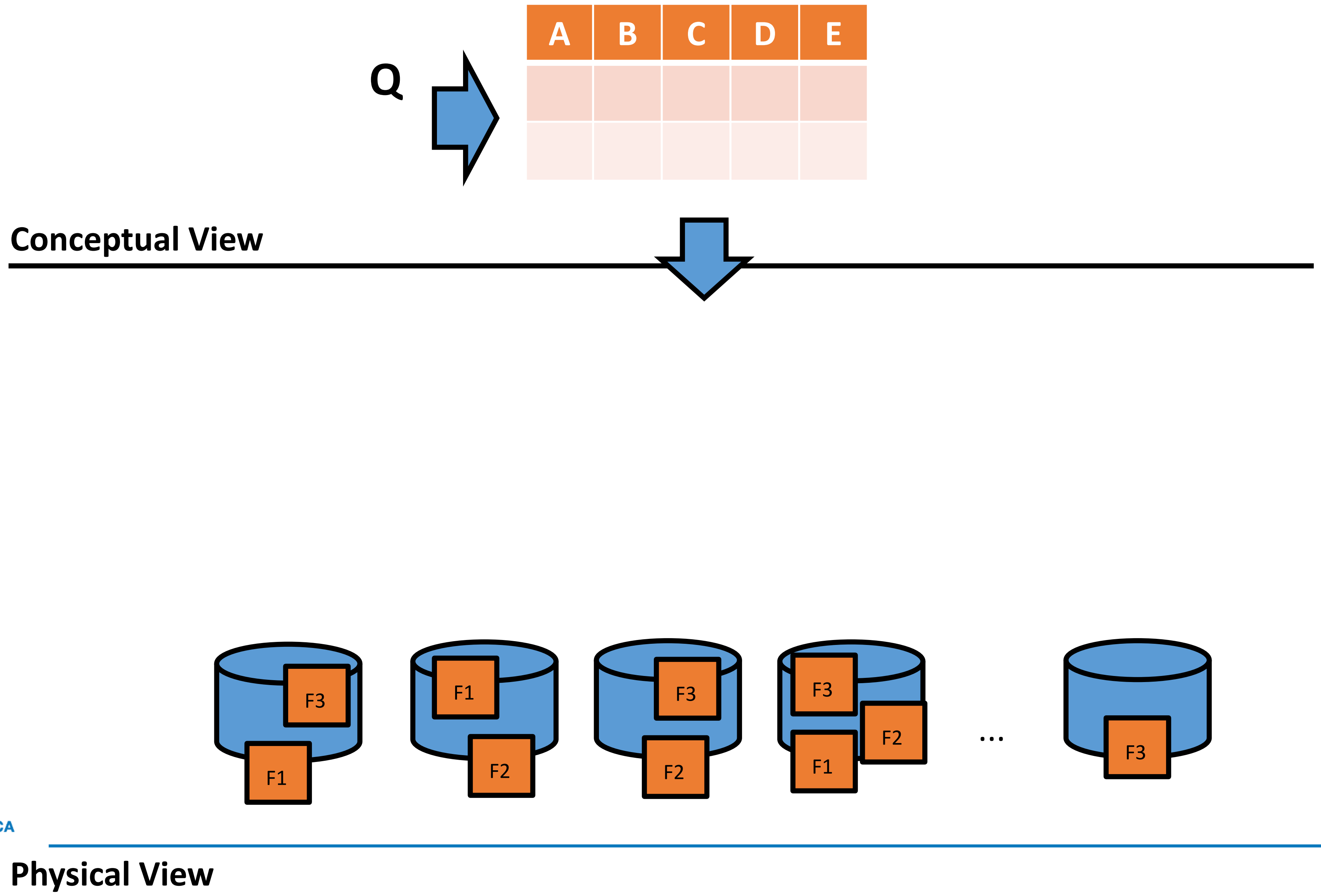
Challenge II

The Global Catalog

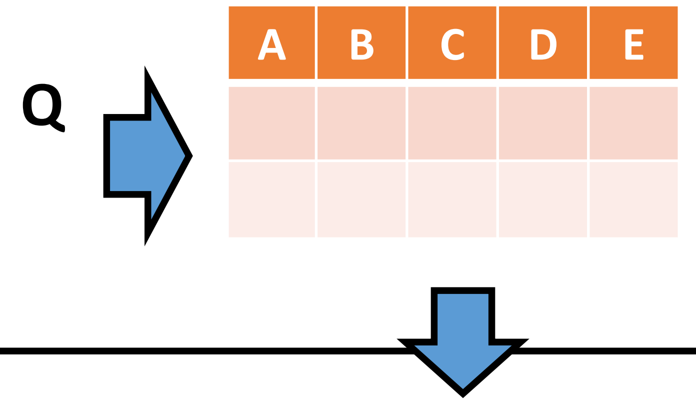
Challenge II: Global Catalog



Challenge II: Global Catalog



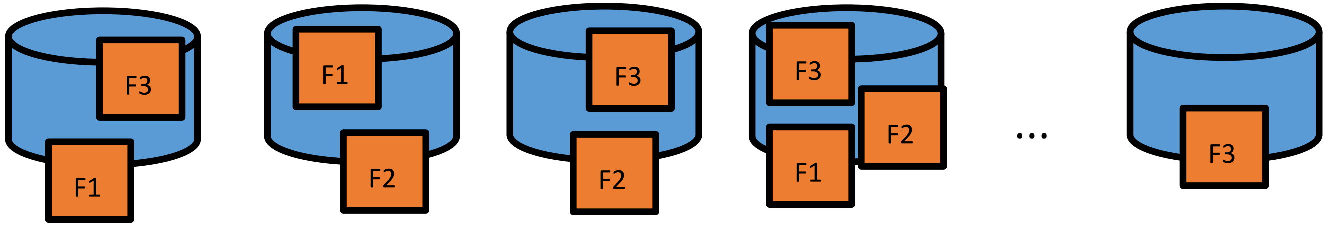
Challenge II: Global Catalog



Conceptual View

?

How to know this table is fragmented and where the fragments are?

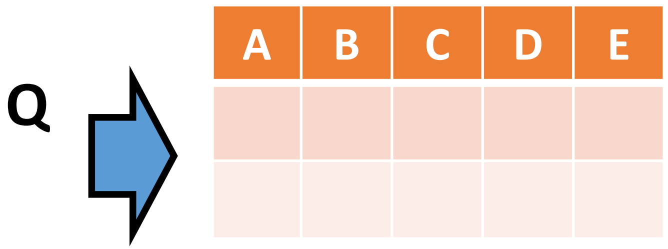


Physical View

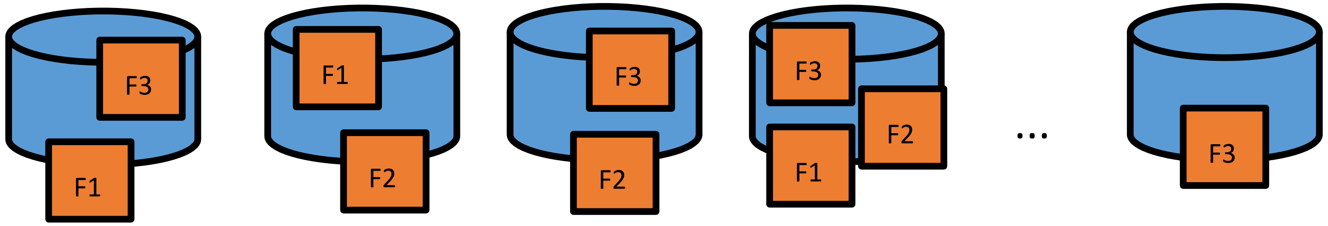
Challenge II: Global Catalog

- Centralized version (@master)
  - Accessing it is a bottleneck
    - Single-point failure
      - May add a mirror
    - Poorer performance
- Distributed version (several *masters*)
  - Replica synchronization
    - Potential inconsistencies

Challenge II: Global Catalog

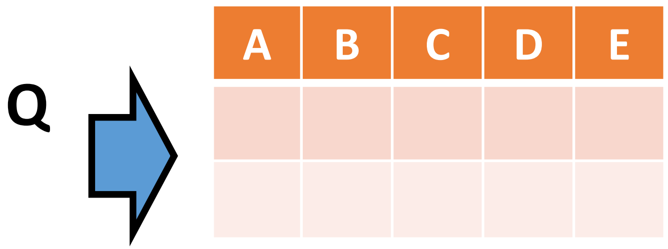


Conceptual View



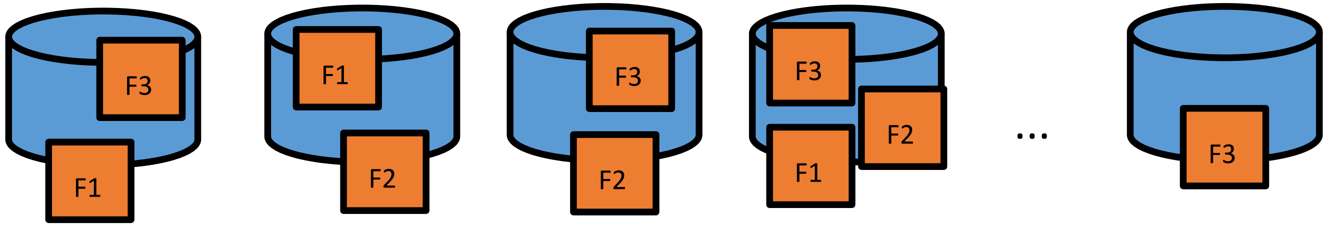
Physical View

Challenge II: Global Catalog



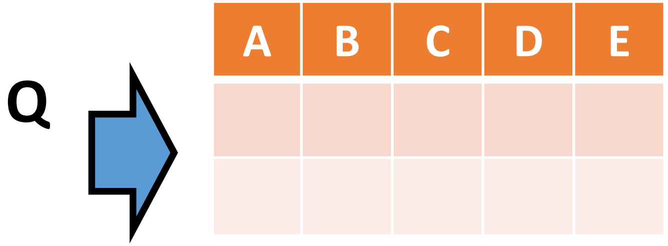
Conceptual View

Catalog:
T <<fragmentation strategy>>
F1: @S1, @S2, @S4
F2: @S2, @S3, @S4
F3: @S1, @S3, @S4, @Sn



Physical View

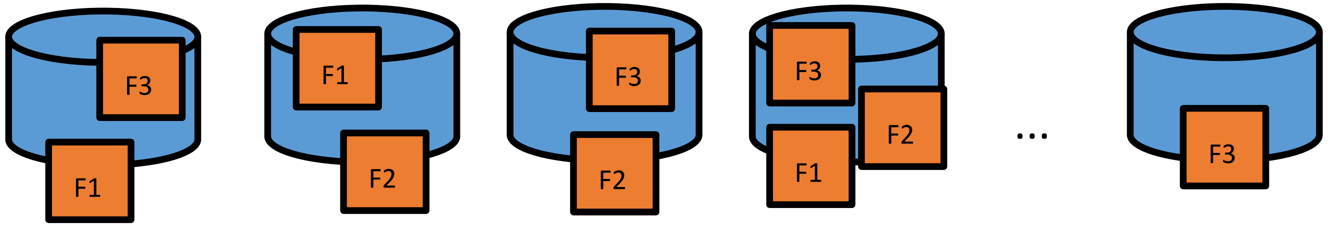
Challenge II: Global Catalog



Conceptual View

Catalog:
T <<fragmentation strategy>>
F1: @S1, @S2, @S4
F2: @S2, @S3, @S4
F3: @S1, @S3, @S4, @Sn

This information is typically stored in a distributed index

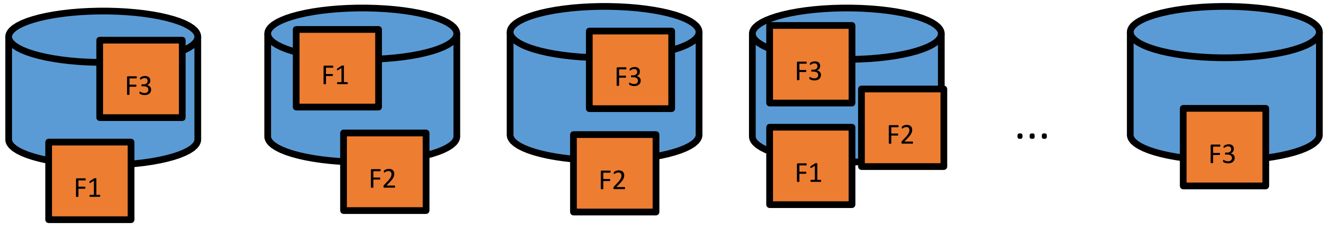


Physical View

Challenge II: Global Catalog

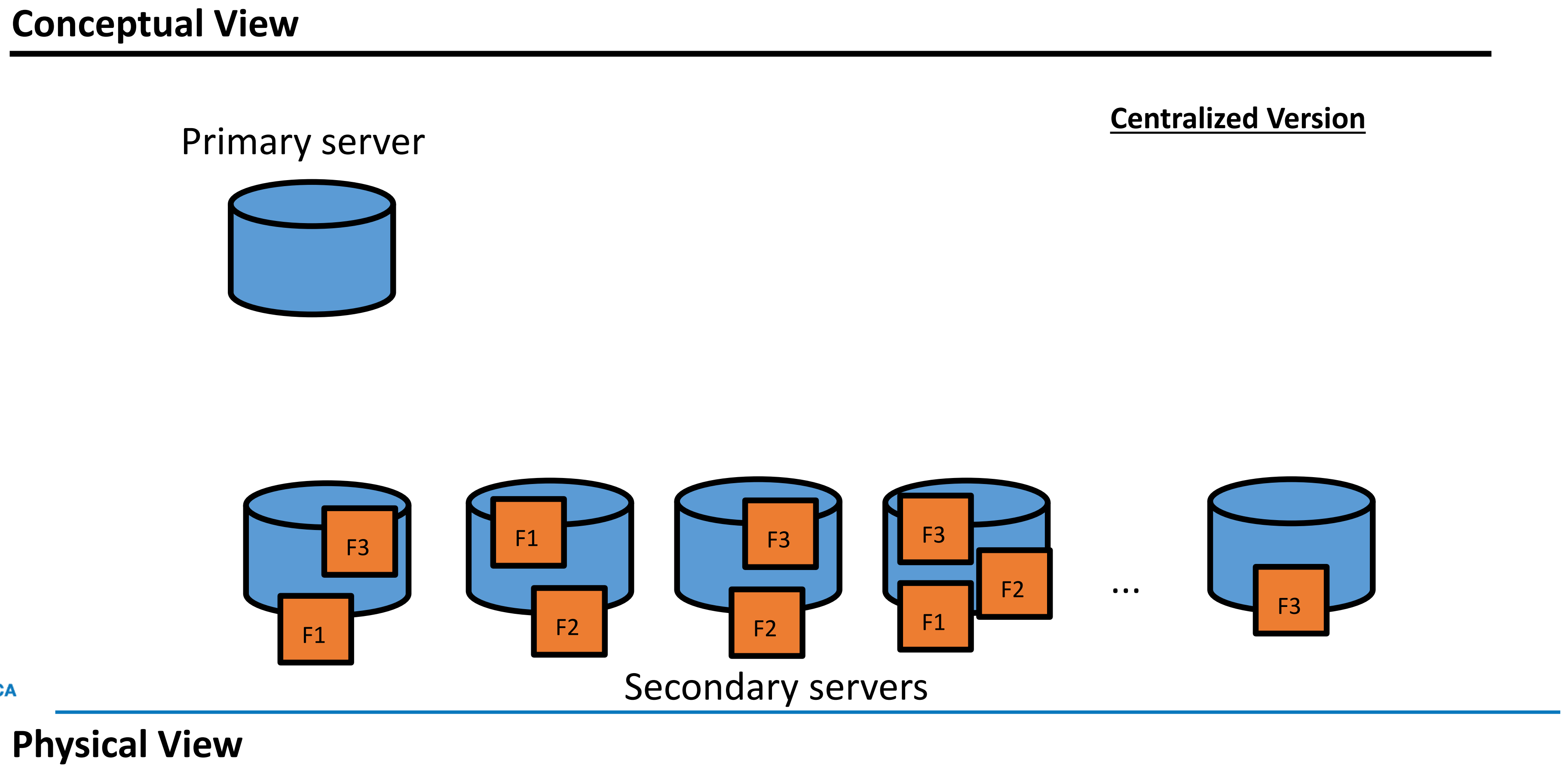
Conceptual View

Centralized Version

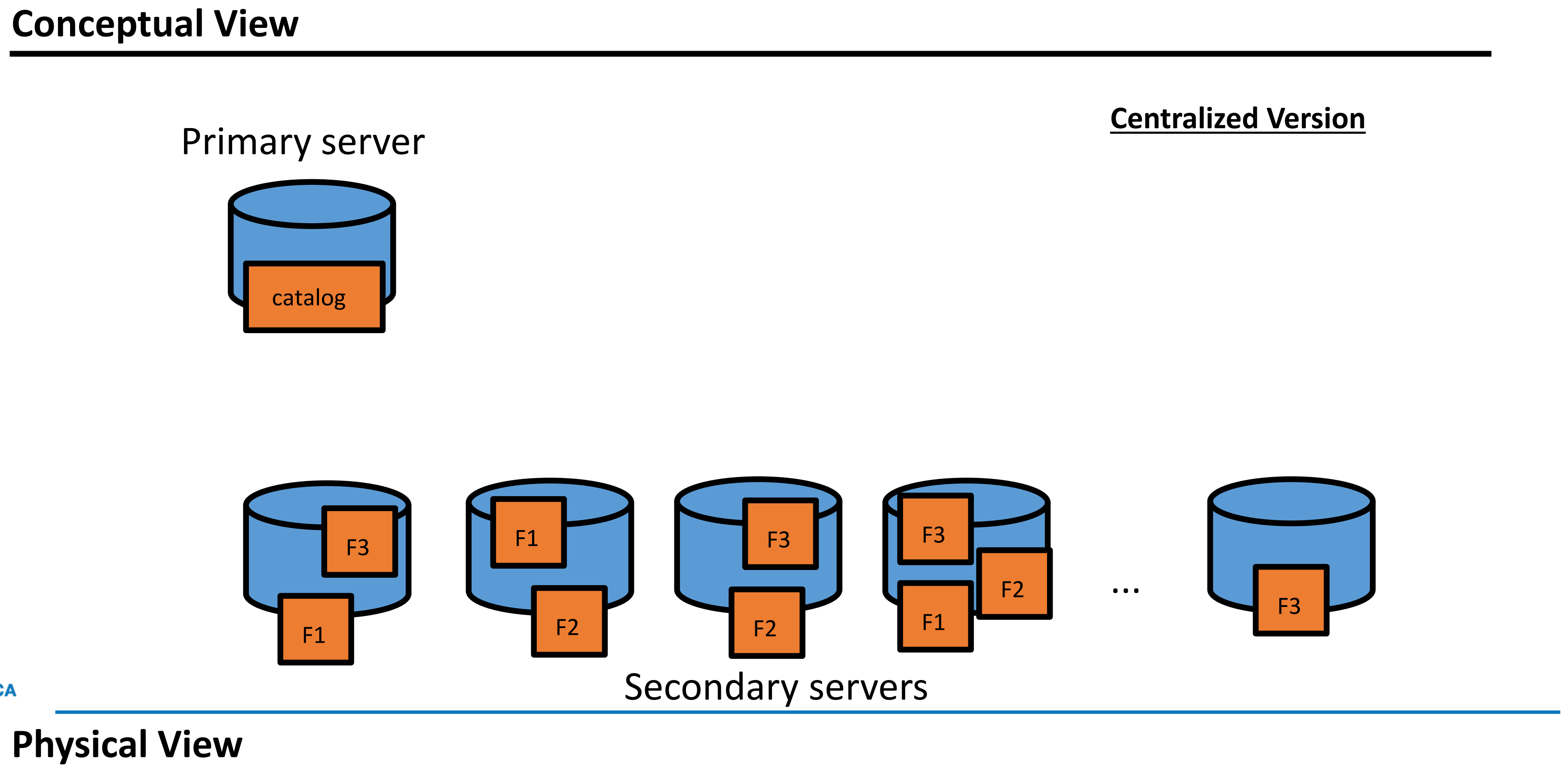


Physical View

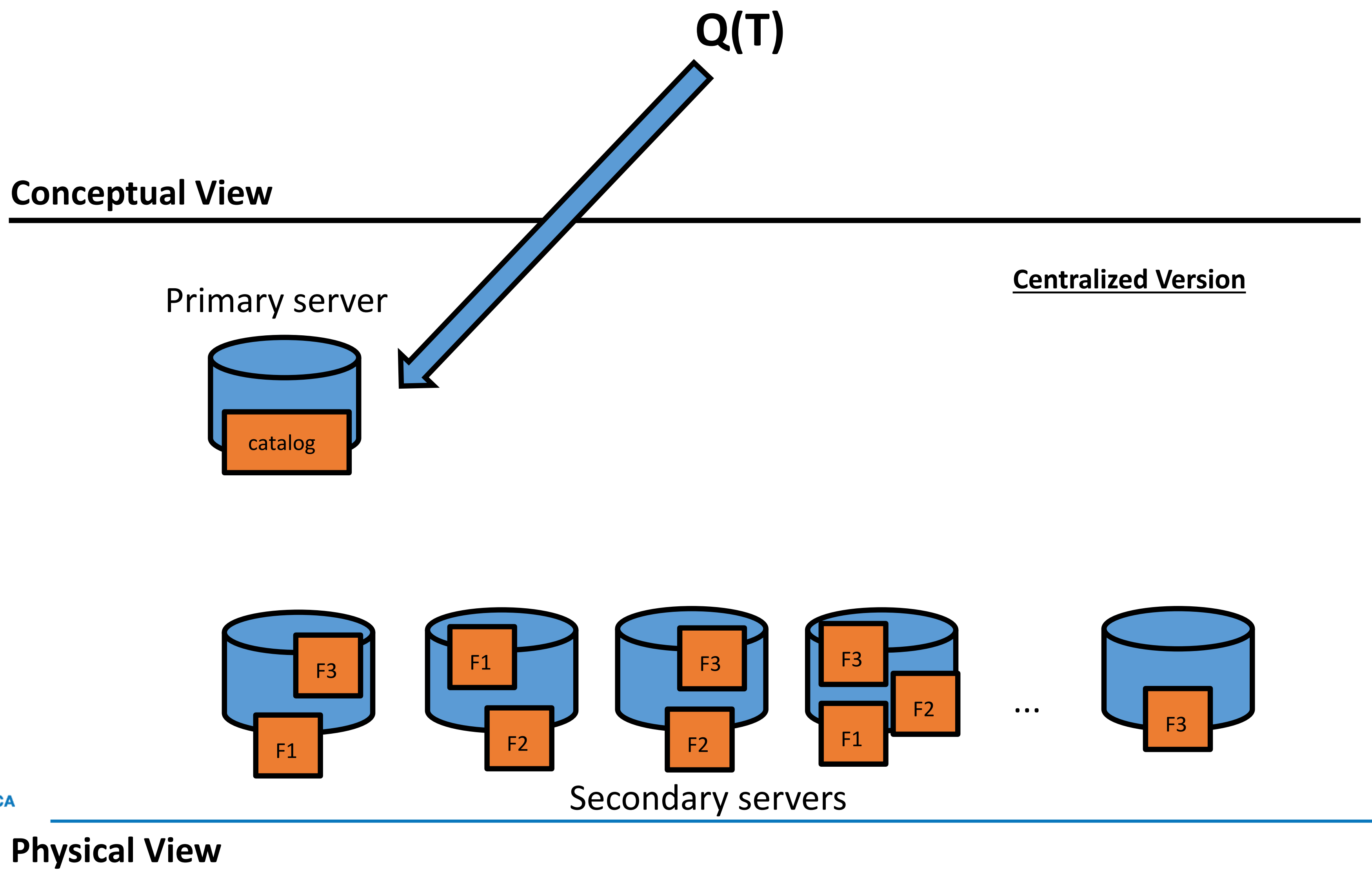
Challenge II: Global Catalog



Challenge II: Global Catalog



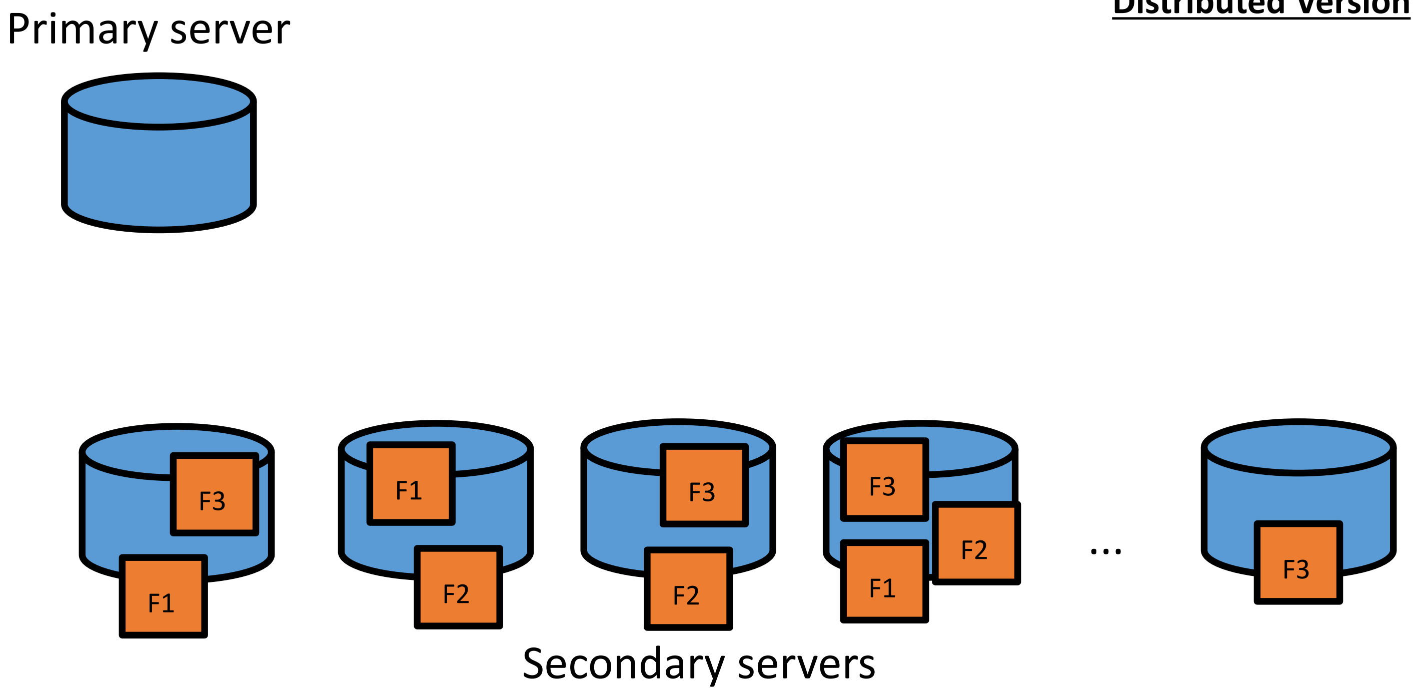
Challenge II: Global Catalog



Challenge II: Global Catalog

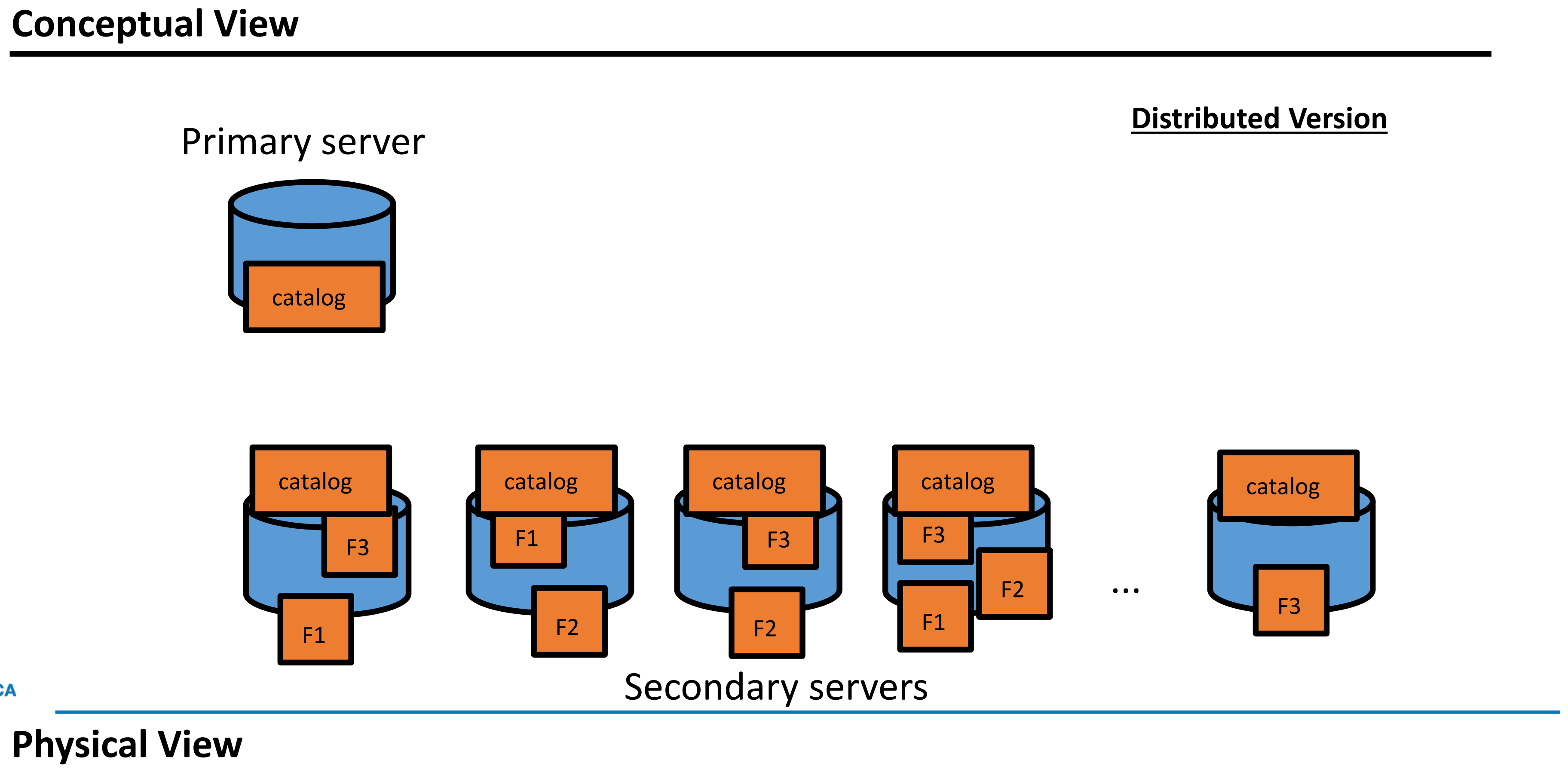
Conceptual View

Distributed Version

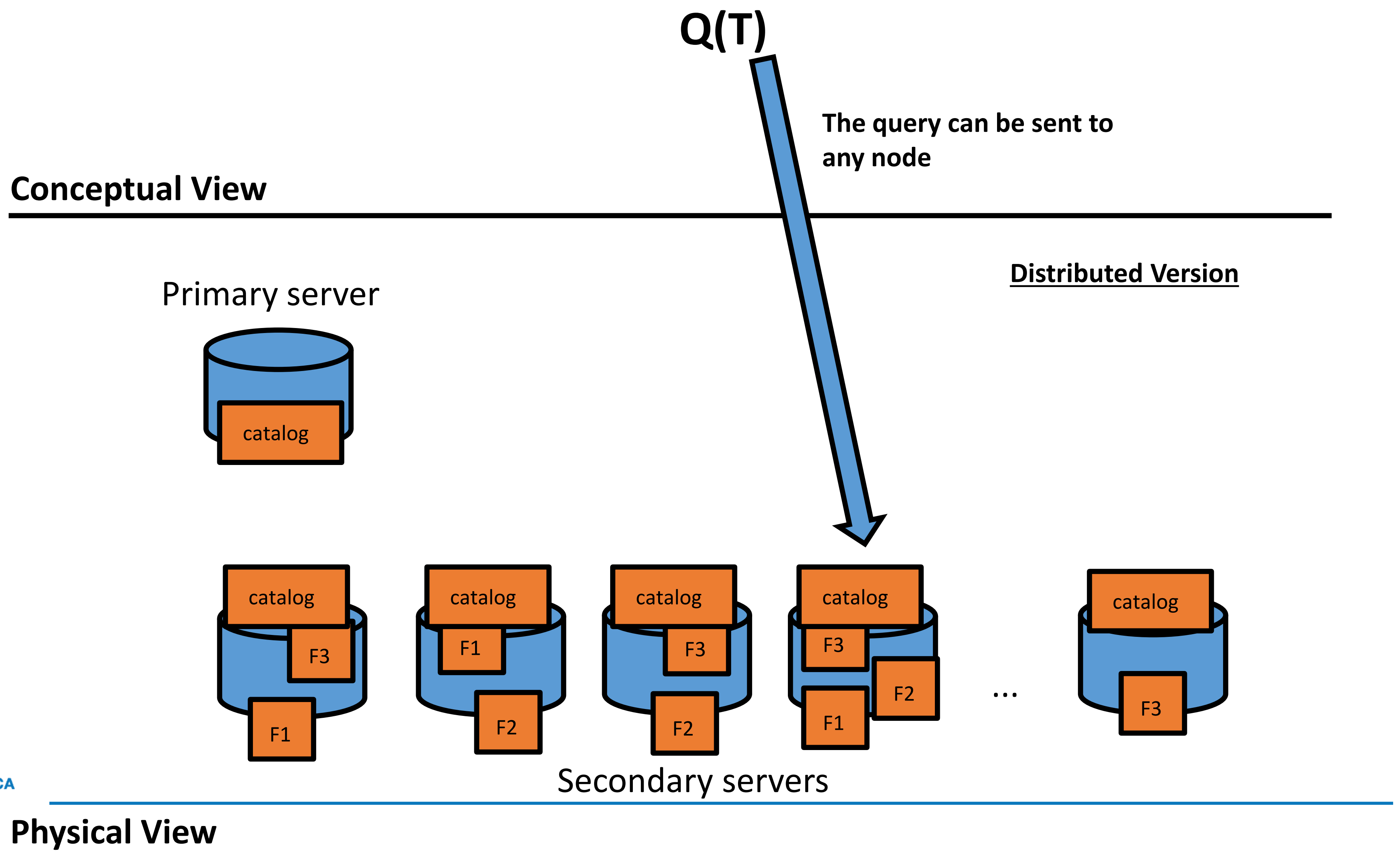


Physical View

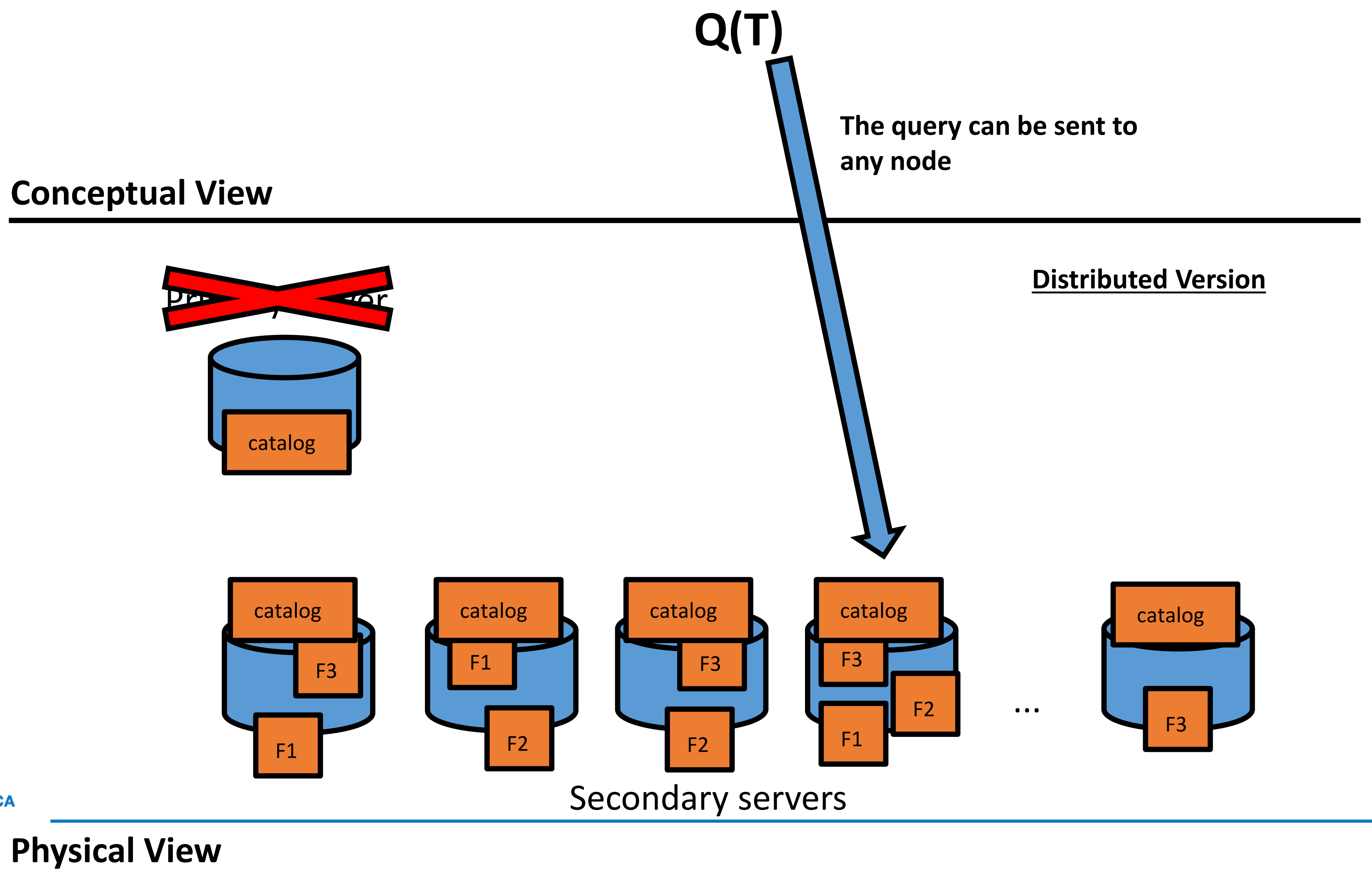
Challenge II: Global Catalog



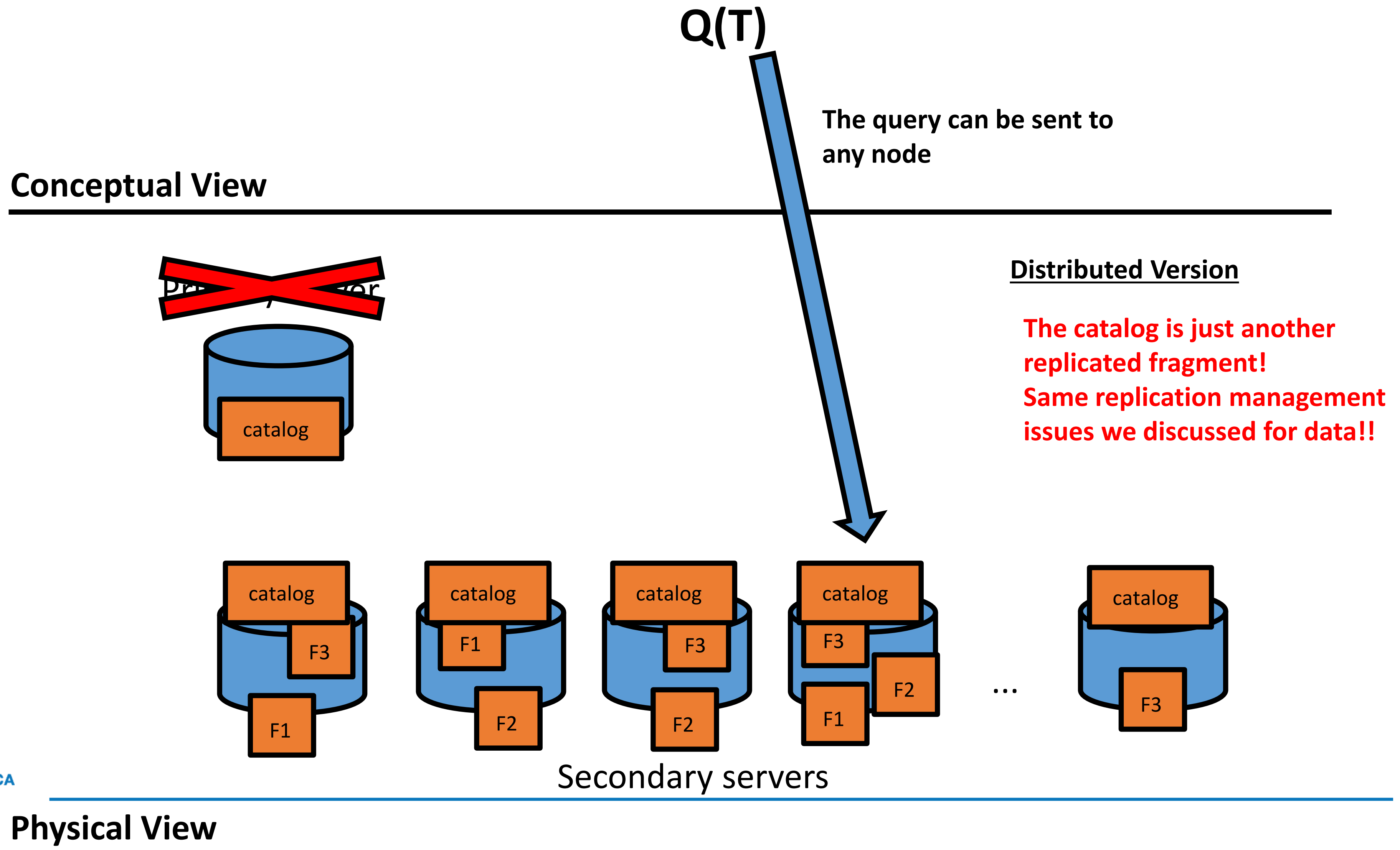
Challenge II: Global Catalog



Challenge II: Global Catalog



Challenge II: Global Catalog



Summary

- Benefits of distributed systems and their relevance for NOSQL
- What is a distributed DBMS
- System architecture of a DDBMS
  - Distribution transparency
  - Replication transparency
  - Fragmentation transparency
- Distributed Database design
  - Data fragmentation
  - Data allocation
    - Data replication
- Global catalog management

Bibliography

- M.T. Özsu and P. Valduriez. *Principles of Distributed Database Systems*. Second edition. Prentice Hall, 1999
- Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart. *Web Data Management*. Cambridge Press, 2011.
- L. Liu, M.T. Özsu (Eds.). *Encyclopedia of Database Systems*. Springer, 2009