Intro to Distributed Databases

Big Data Systems

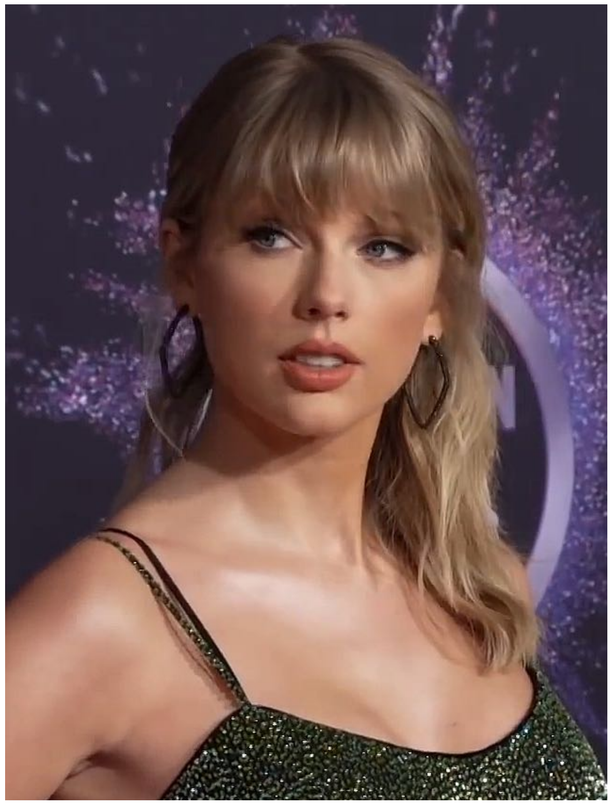
Dr. Rubi Boim

Agenda for today

- Motivation
- Distributed relational database?
- “Going distributed”

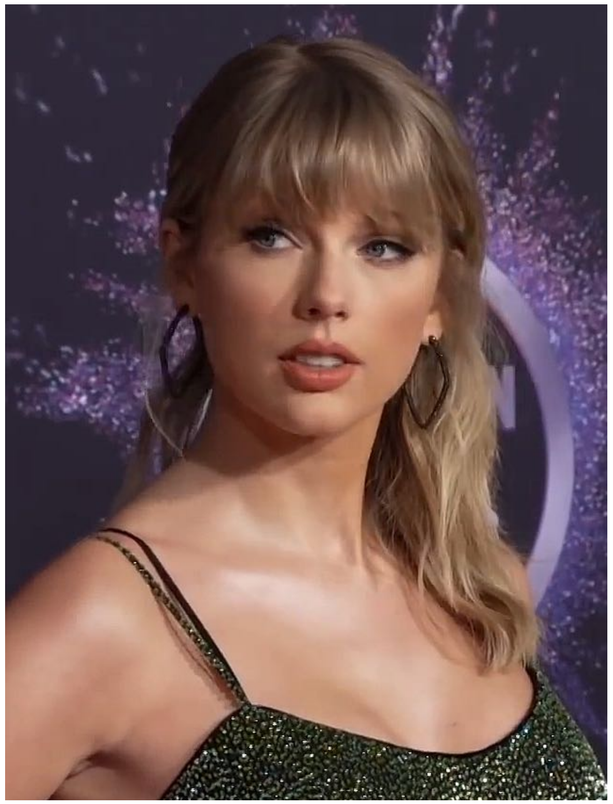
Example 1

Example 1



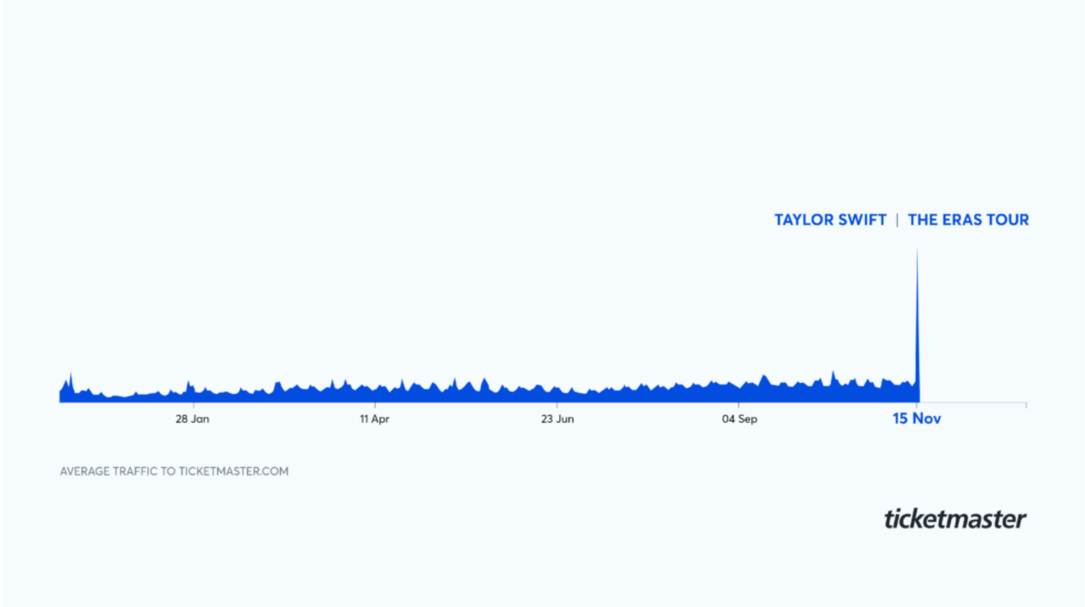
* image by Wikipedia

Example 1



HOME / BUSINESS / MUSIC

Taylor Swift Ticket Sales Crash Ticketmaster, Ignite Fan Backlash, Renew Calls To Break Up Service: “Ticketmaster Is A Monopoly”



14m users spike, 2m tickets sold, then Ticketmaster crashed

Example 2

* image by Wikipedia

Example 2

Elon Musk
@elonmusk

Reinstate former President Trump

Yes 51.8%

No 48.2%

15,085,458 votes · Final results

2:47 AM · Nov 19, 2022 · Twitter for iPhone

230.2K Retweets **76K** Quote Tweets **768.7K** Likes

Tweet your reply

Reply

Elon Musk @elonmusk · Nov 19
Replying to @elonmusk
Vox Populi, Vox Dei

27K 26.5K 296.1K

Elon Musk @elonmusk · 16h
134M people have seen this poll

15.4K 13.2K 229.3K

Example 2

Elon Musk
@elonmusk

Reinstate former President Trump

Yes 51.8%

No 48.2%

15,085,458 votes · Final results

2:47 AM · Nov 19, 2022 · Twitter for iPhone

230.2K Retweets **76K** Quote Tweets **768.7K** Likes

Tweet your reply

Reply

Elon Musk @elonmusk · Nov 19
Replying to @elonmusk
Vox Populi, Vox Dei

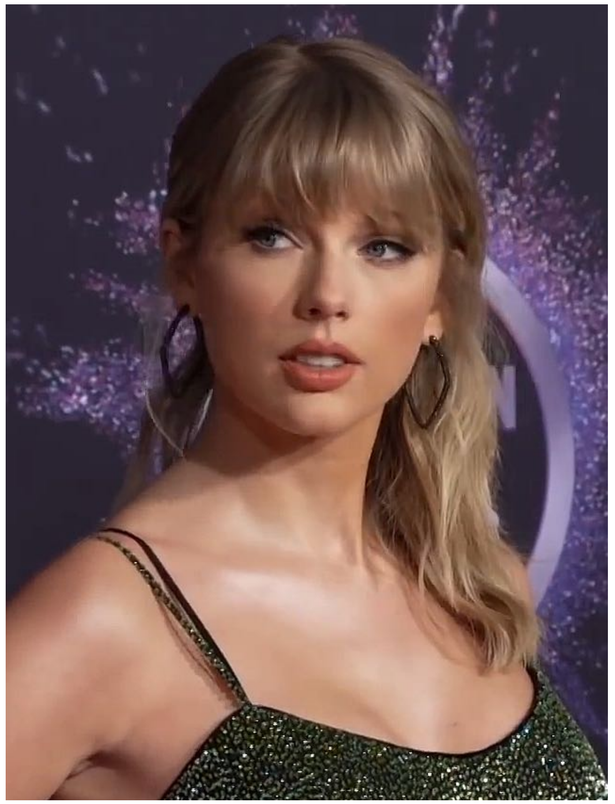
27K 26.5K 296.1K

Elon Musk @elonmusk · 16h
134M people have seen this poll

15.4K 13.2K 229.3K

Example

How Twitter managed to
handle 10x more traffic than
Ticketmaster without any crash?



Motivation

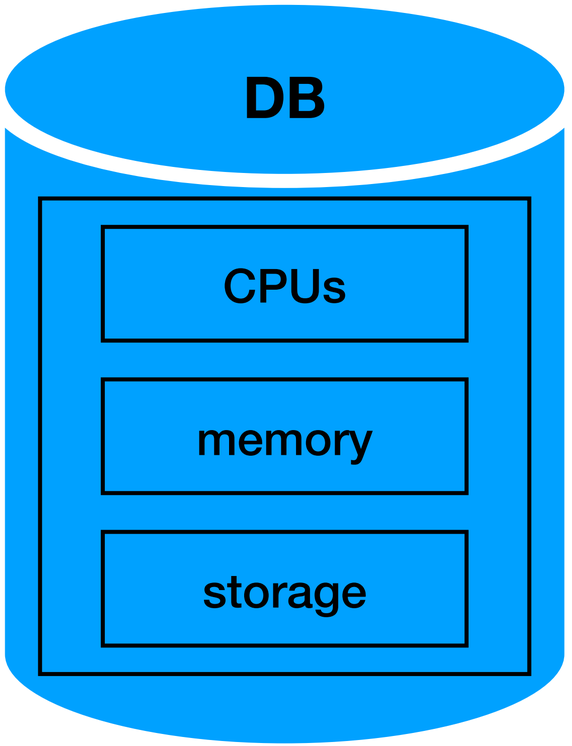
Centralized RDBMS

- Used everywhere
- Proven technology

- So why do we need anything else?

Motivation

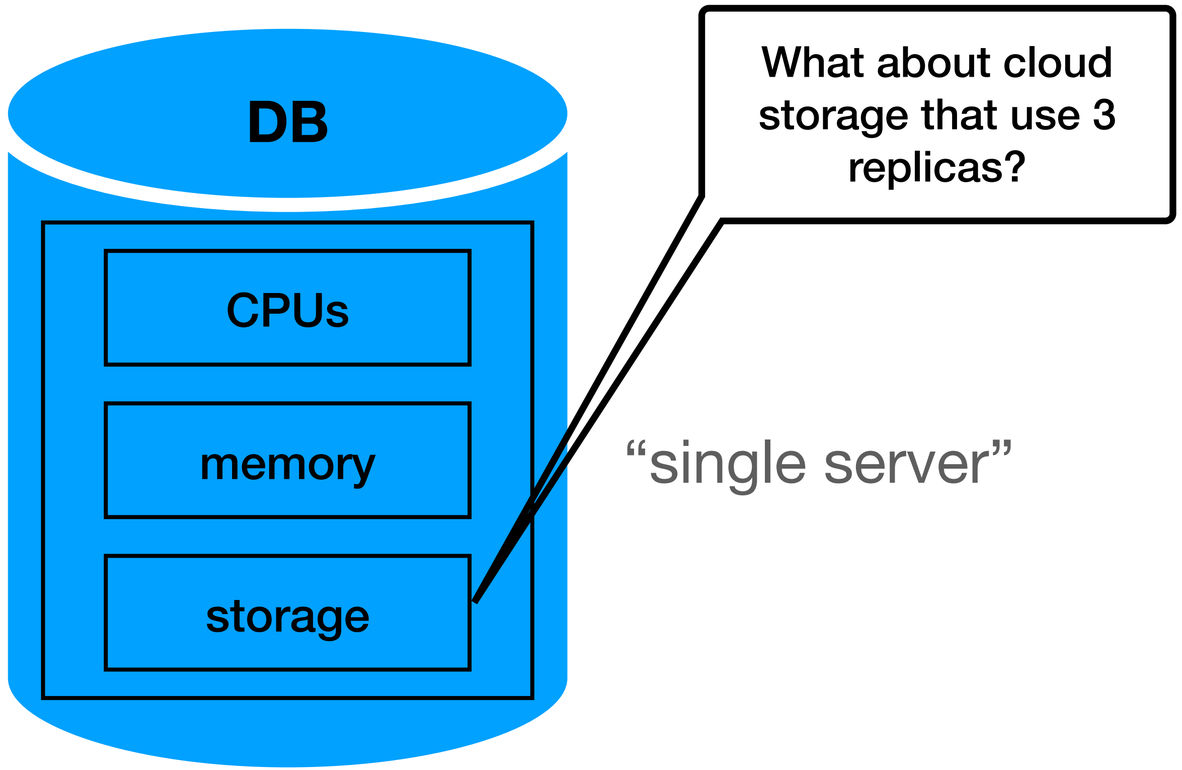
Centralized RDBMS (simplified)



“single server”

Motivation

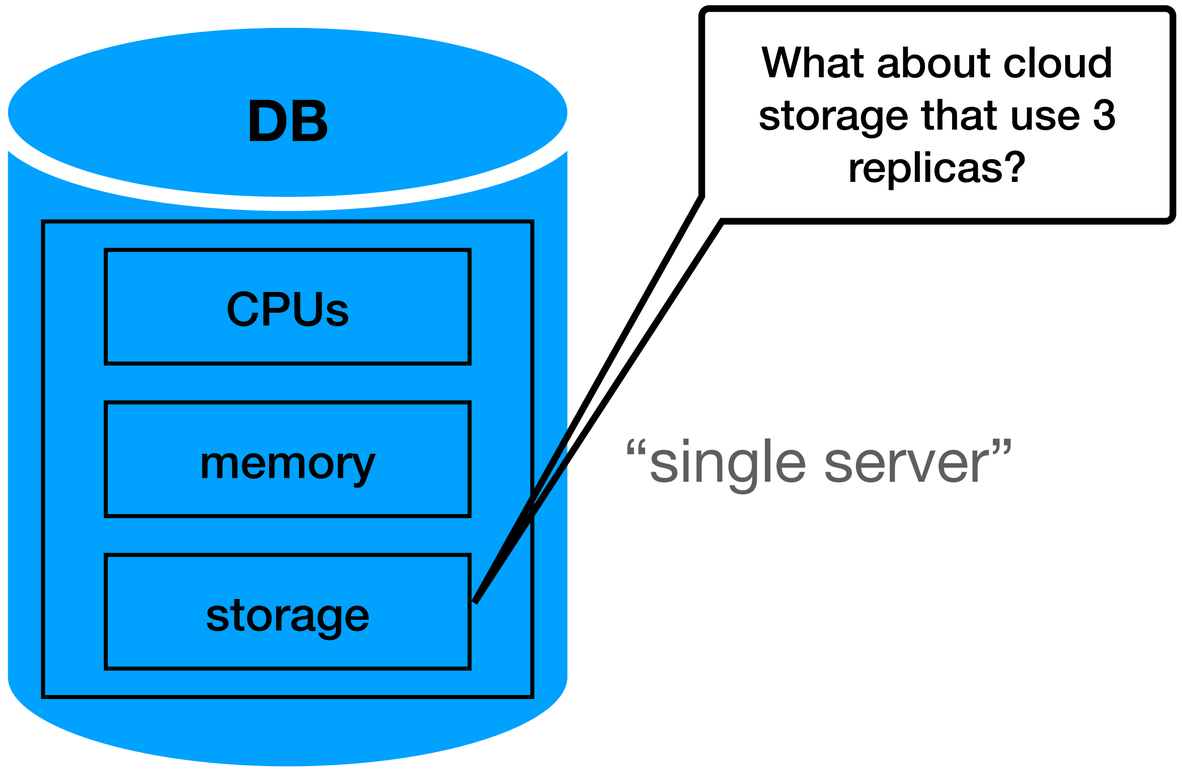
Centralized RDBMS (simplified)



Motivation

Centralized RDBMS (simplified)

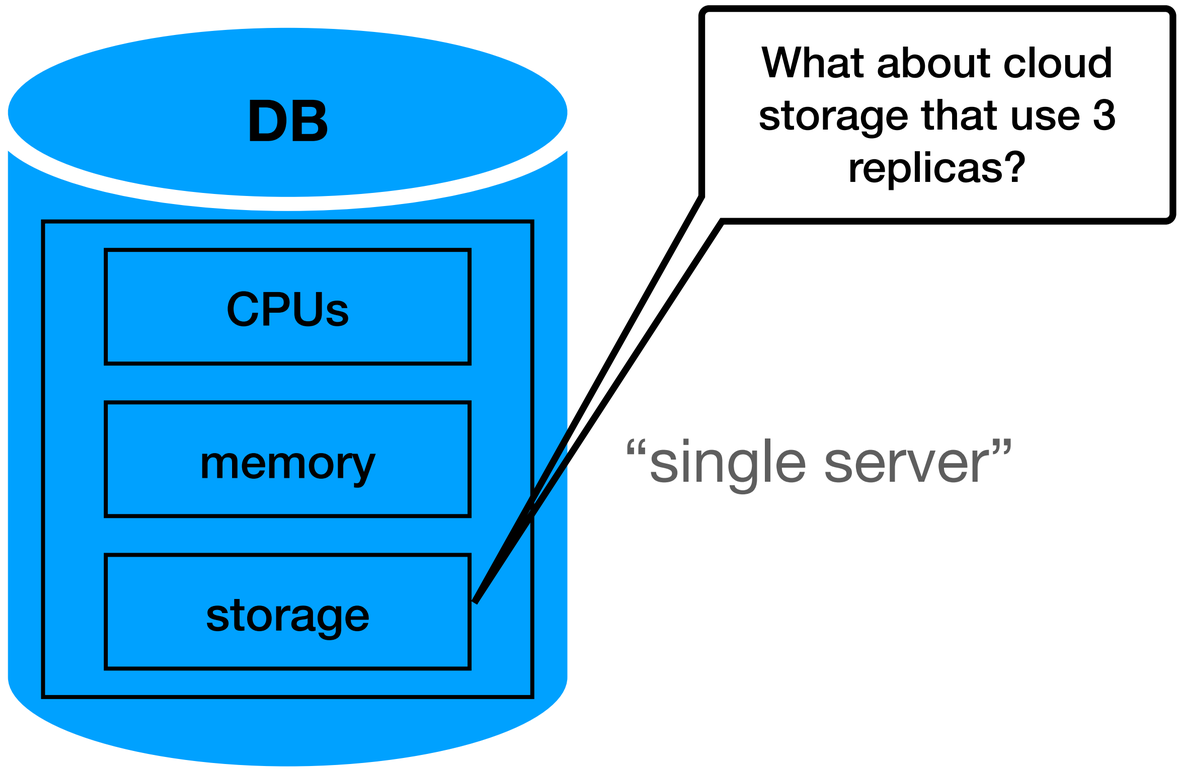
- What happens if we have more than 100m rows?
  - Storage
  - Query time



Motivation

Centralized RDBMS (simplified)

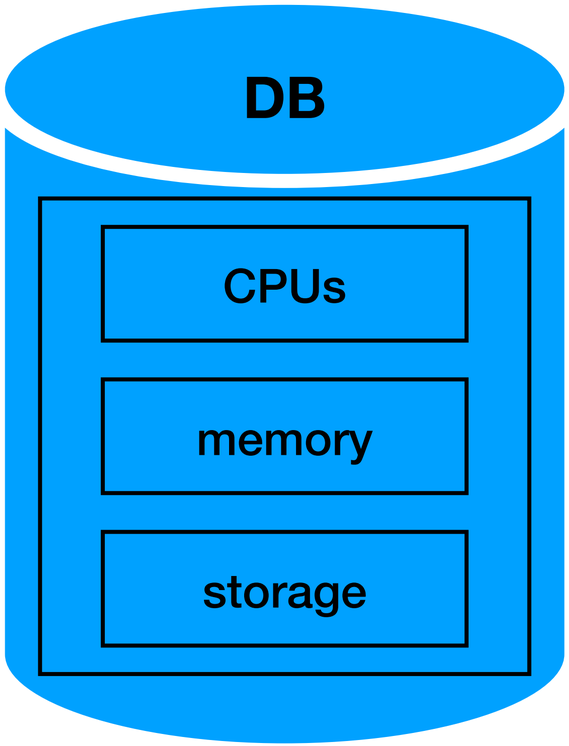
- What happens if we have more than 100m rows?
  - Storage
  - Query time
- The index is crucial



Motivation

Centralized RDBMS (simplified)

- What happens if we have more than 100m rows?
  - Storage
  - Query time
- What happens if the index is bigger than the memory?



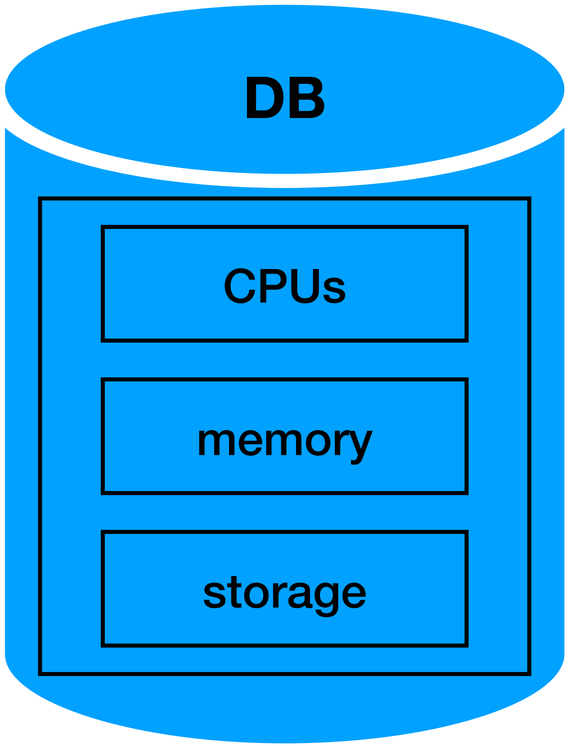
“single server”

Motivation

Centralized RDBMS (simplified)

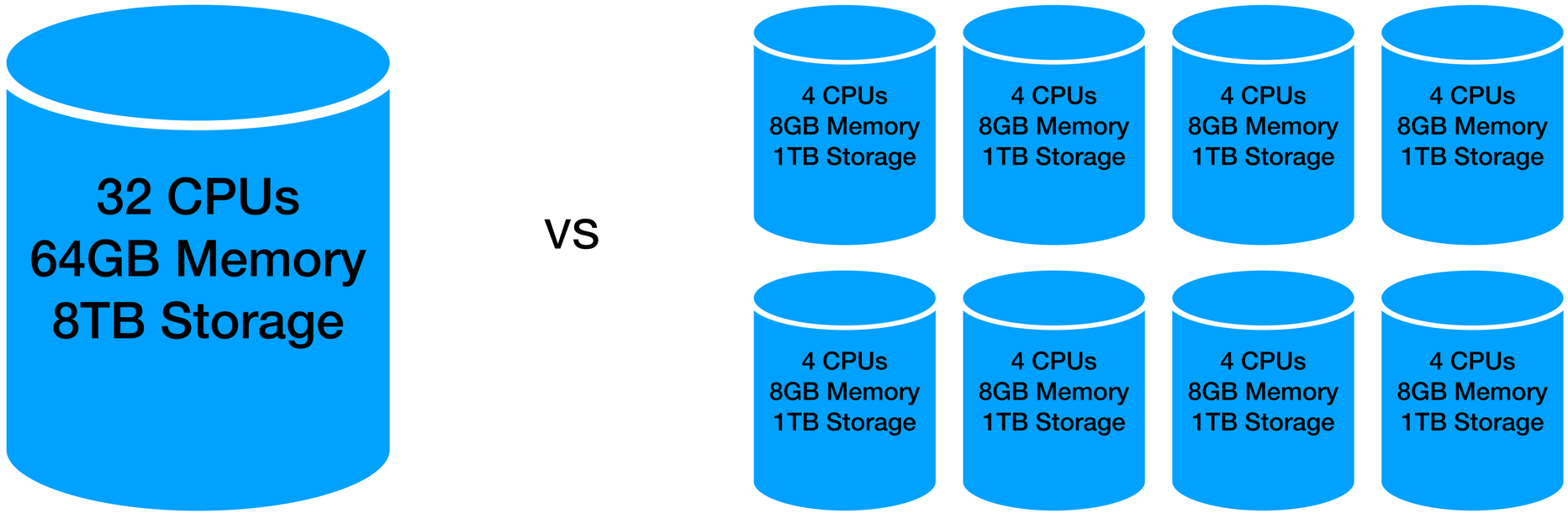
- What happens if we have more than 100m rows?
  - Storage
  - Query time
- What happens if the index is bigger than the memory?

Too many “page reads” can be slow

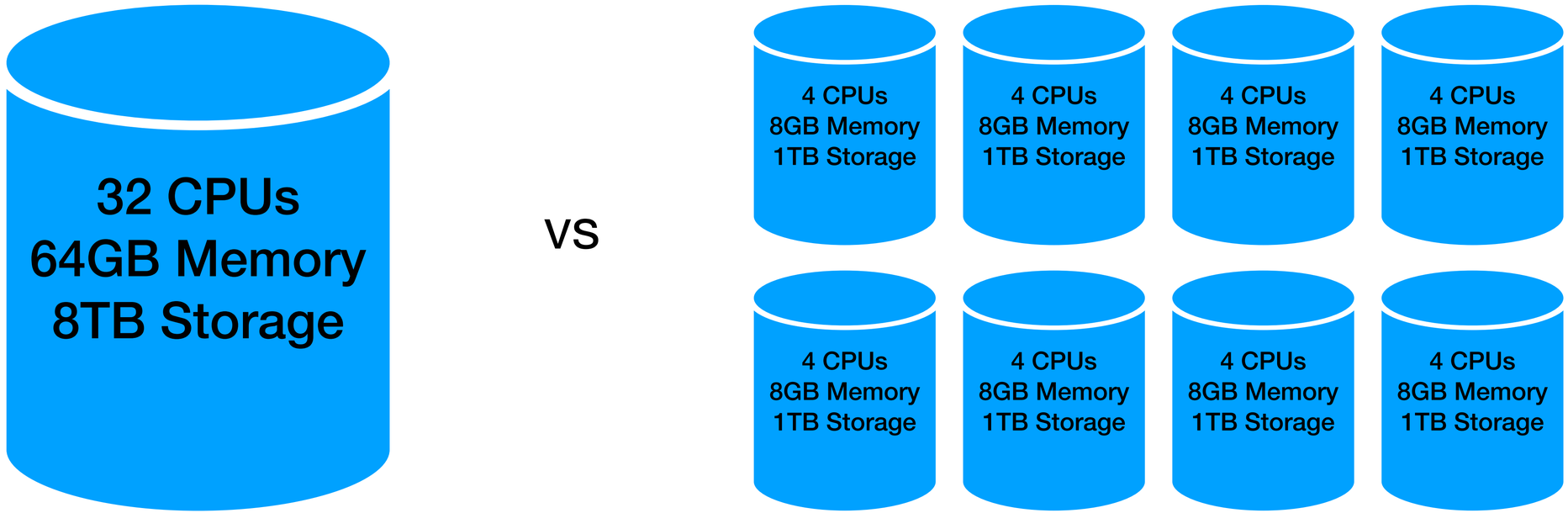


“single server”

Scale up vs Scale out



Scale up vs Scale out



Centralized / “Parallel” database

- scale has a limit
- “simple” management

Distributed database

- infinite scale
- “complex” management

RDBMS limitations

- Limited performance
  vertical scale vs horizontal scale
- Data protection
  no replication (*cloud storage layer with redundancy can be used)
- Up time
  downtime on upgrades / hardware problems
- Cost
  *resources are not elastic, pay even if unused

A note on distributed scaling (hot spots)

- Going distributed —> more problems
- In 2019 Jennifer Aniston joined Instagram and posted a single photo
- …
- Instagram crashed temporarily
  (much) more on that on “advance modeling”

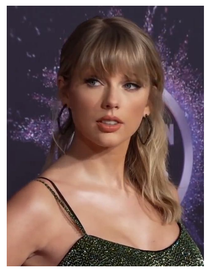
Distributed relational database?

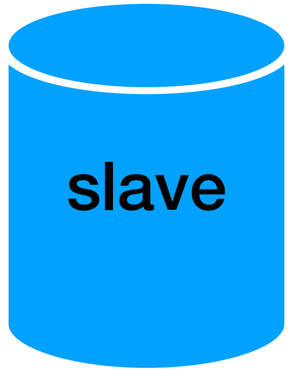
Distributed relational database

- Let’s distribute a relational database
- How would you do it?

RDBMS master - slave

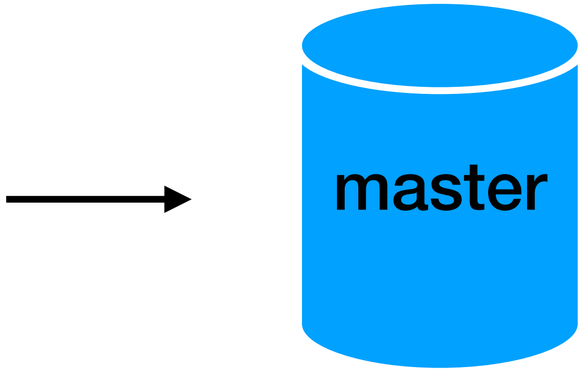
- On each write to the master we sync the slave
- If we detect an error, we transition to the slave





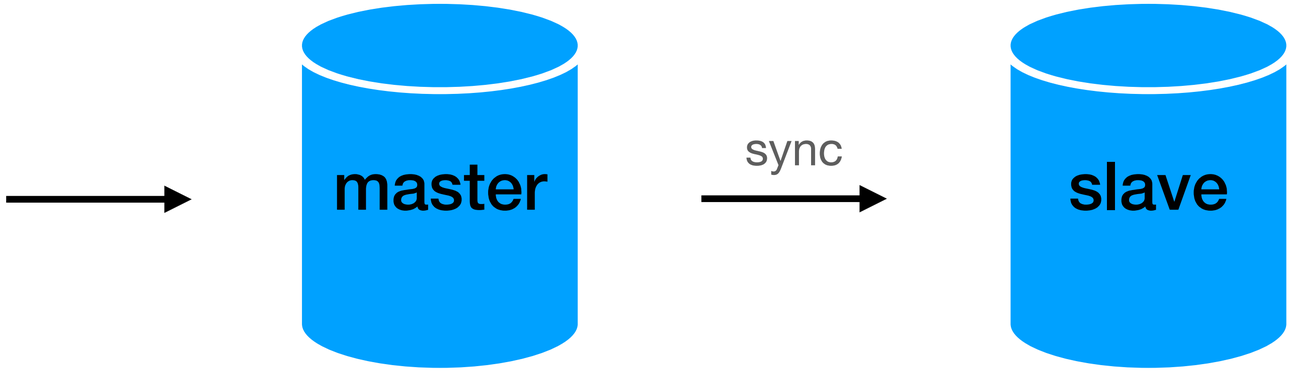
RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave



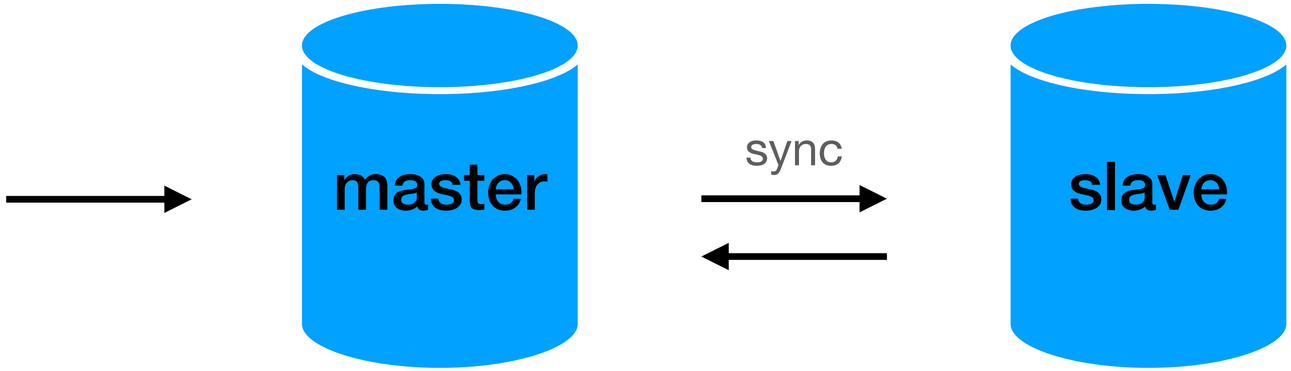
RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave



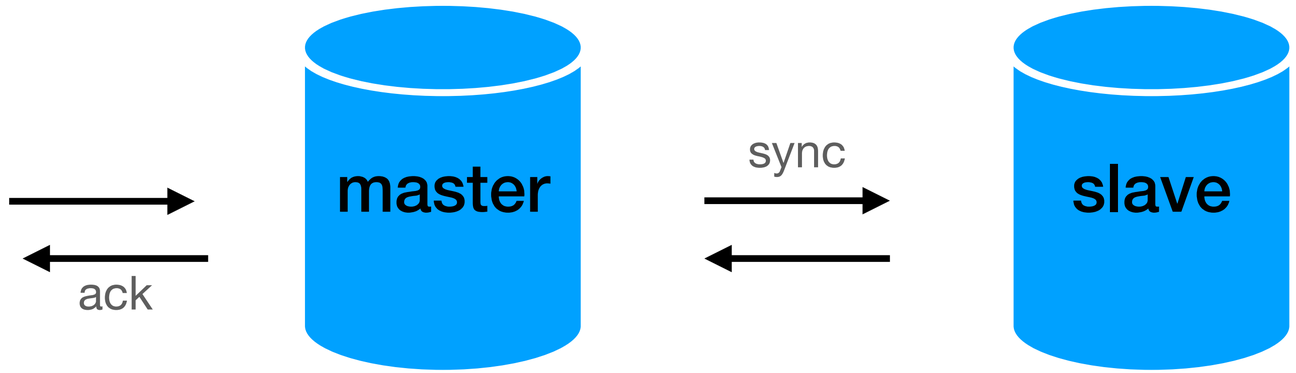
RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave



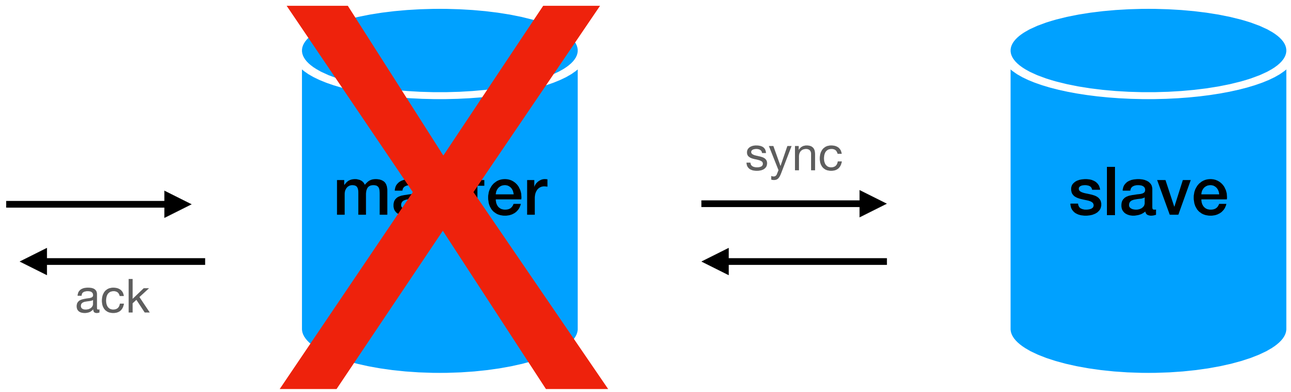
RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave



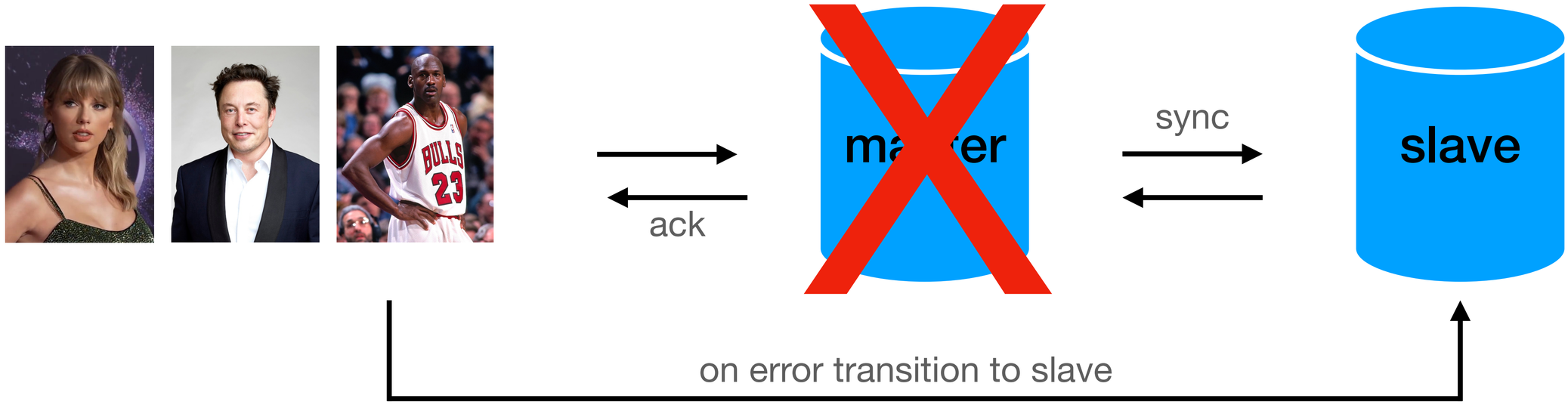
RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave



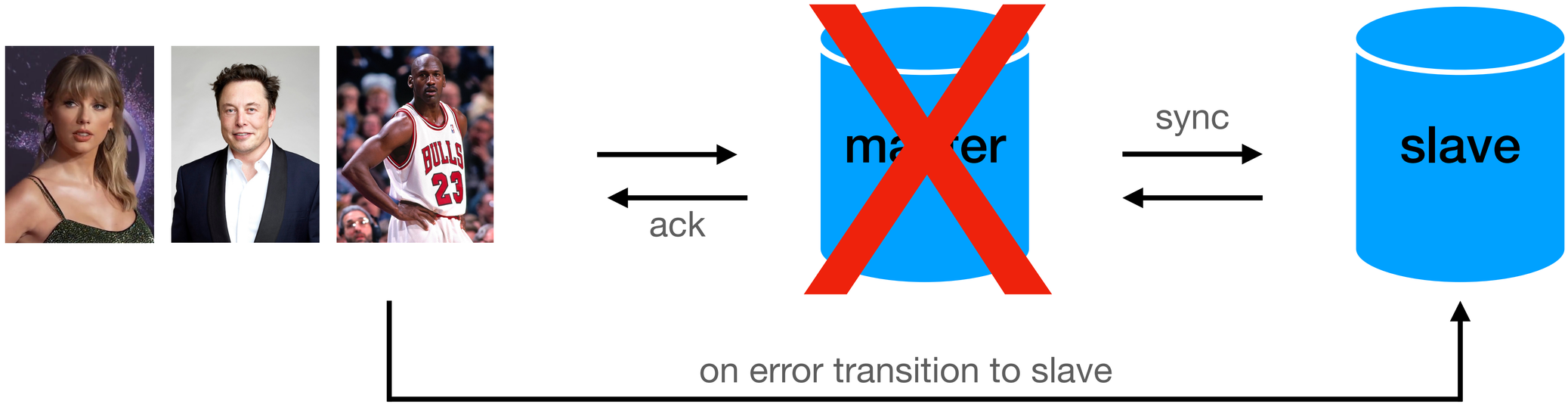
RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave



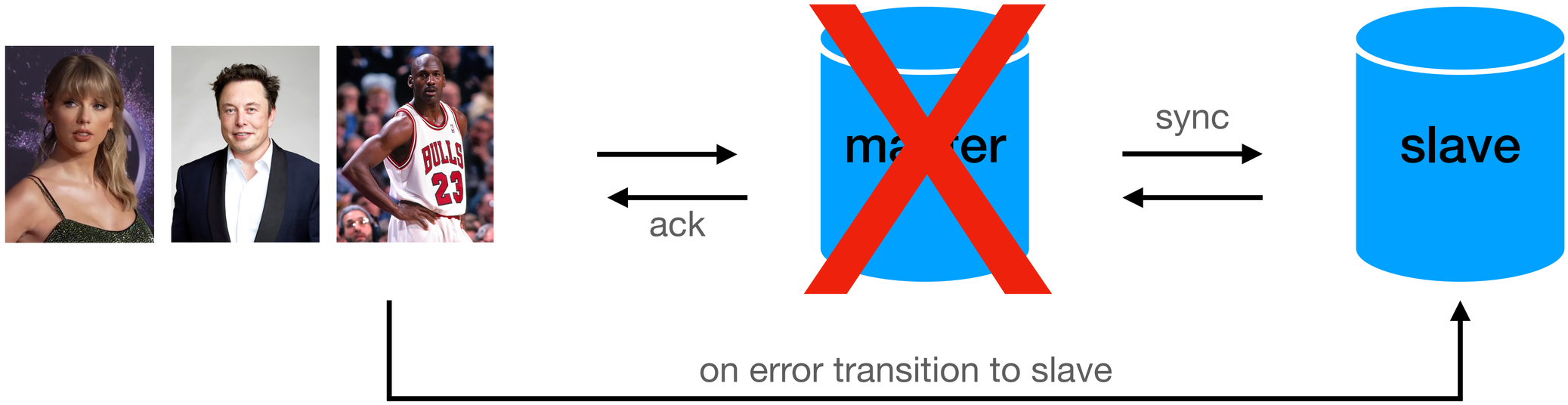
RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave



RDBMS master - slave

- On each write to the master we sync the slave
- If we detect an error, we transition to the slave

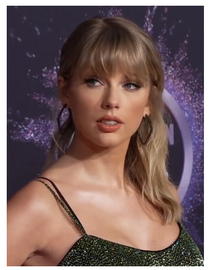


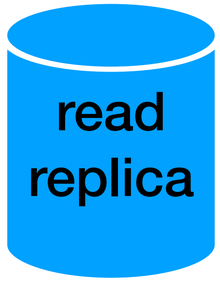
Data protection / up time

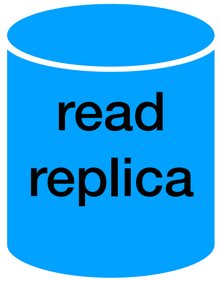
Which limitations did we addressed?

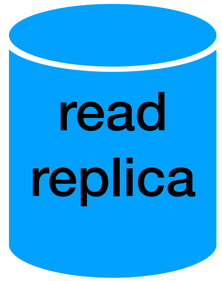
RDBMS read replicas

- On each write to the master we sync the replicas
- If we have a read query, we can use the replicas









RDBMS read replicas

- On each write to the master we sync the replicas
- If we have a read query, we can use the replicas

write

master

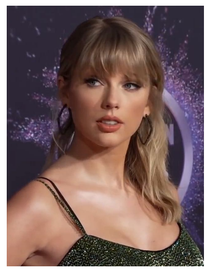
read replica

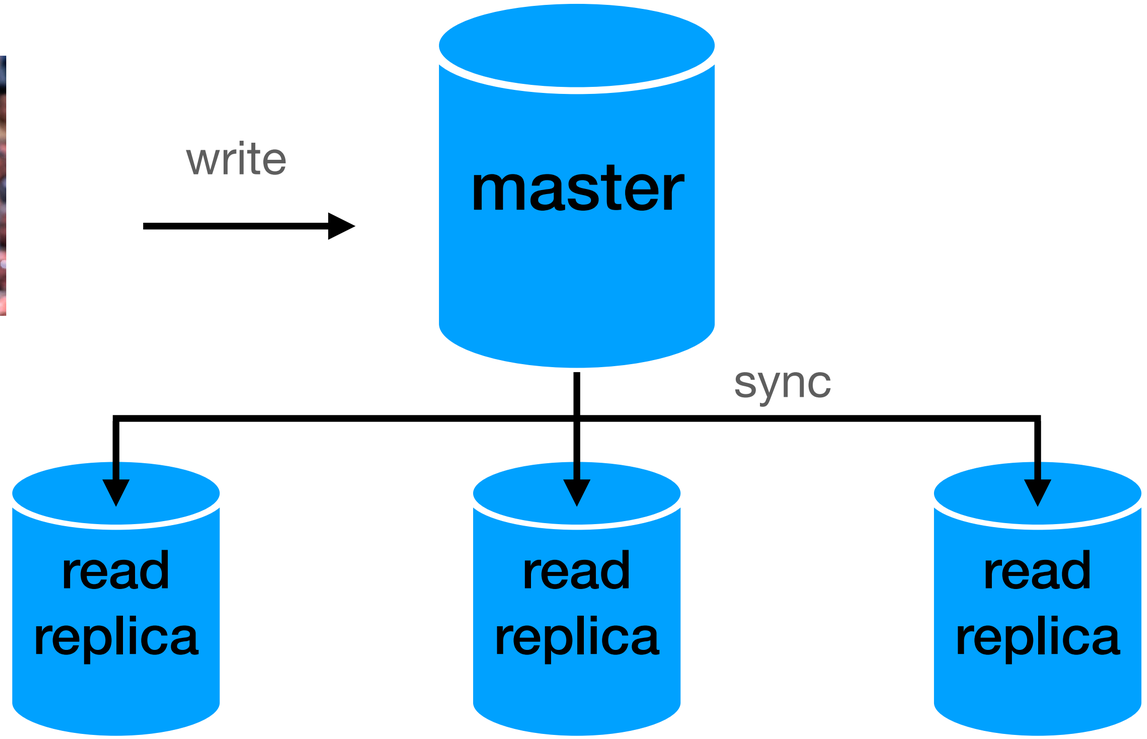
read replica

read replica

RDBMS read replicas

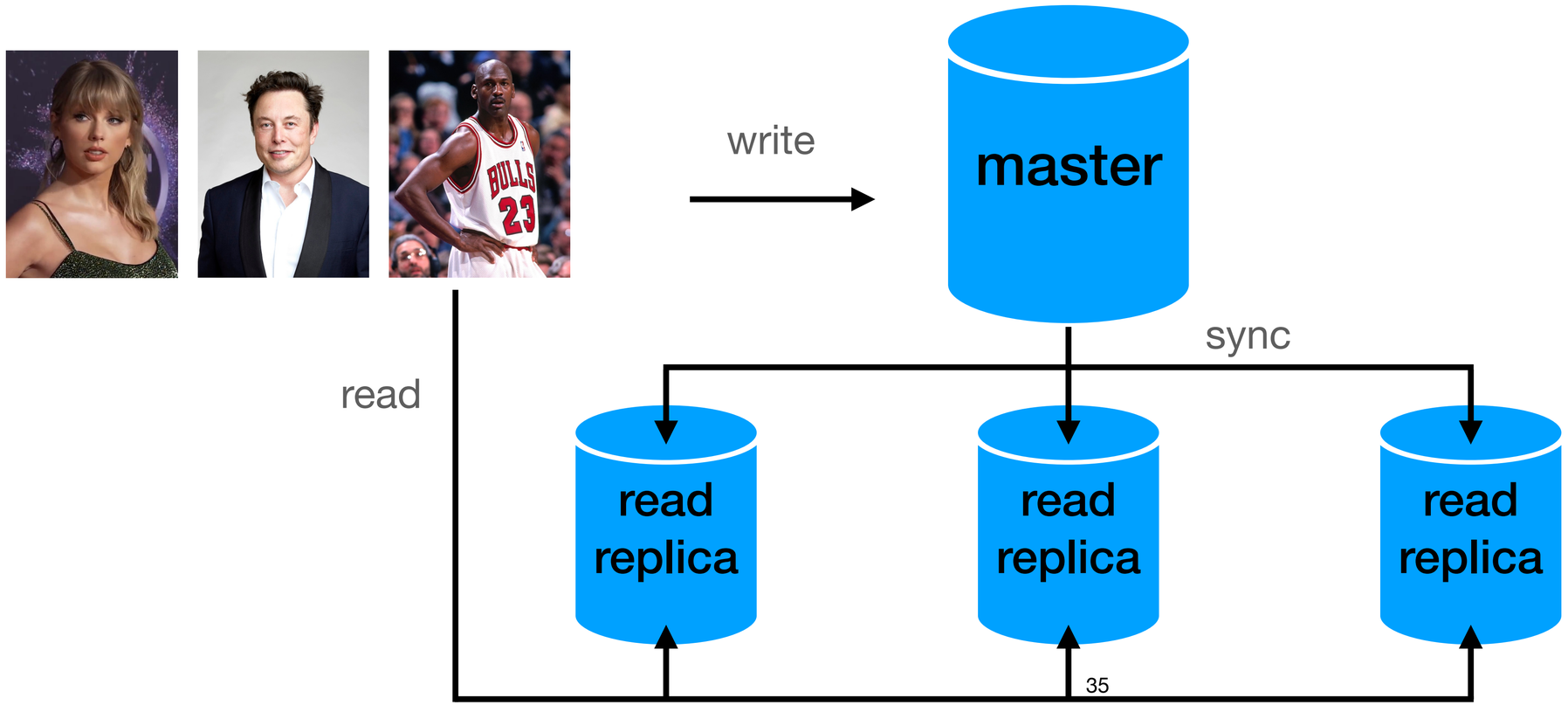
- On each write to the master we sync the replicas
- If we have a read query, we can use the replicas





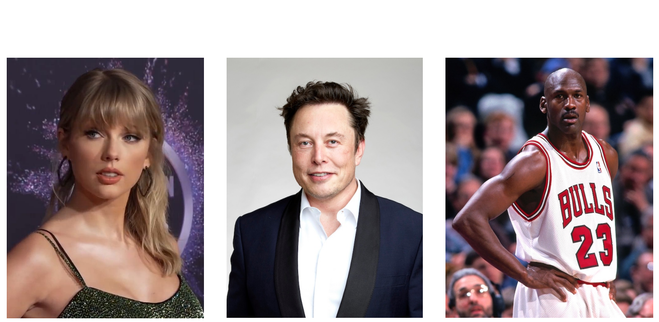
RDBMS read replicas

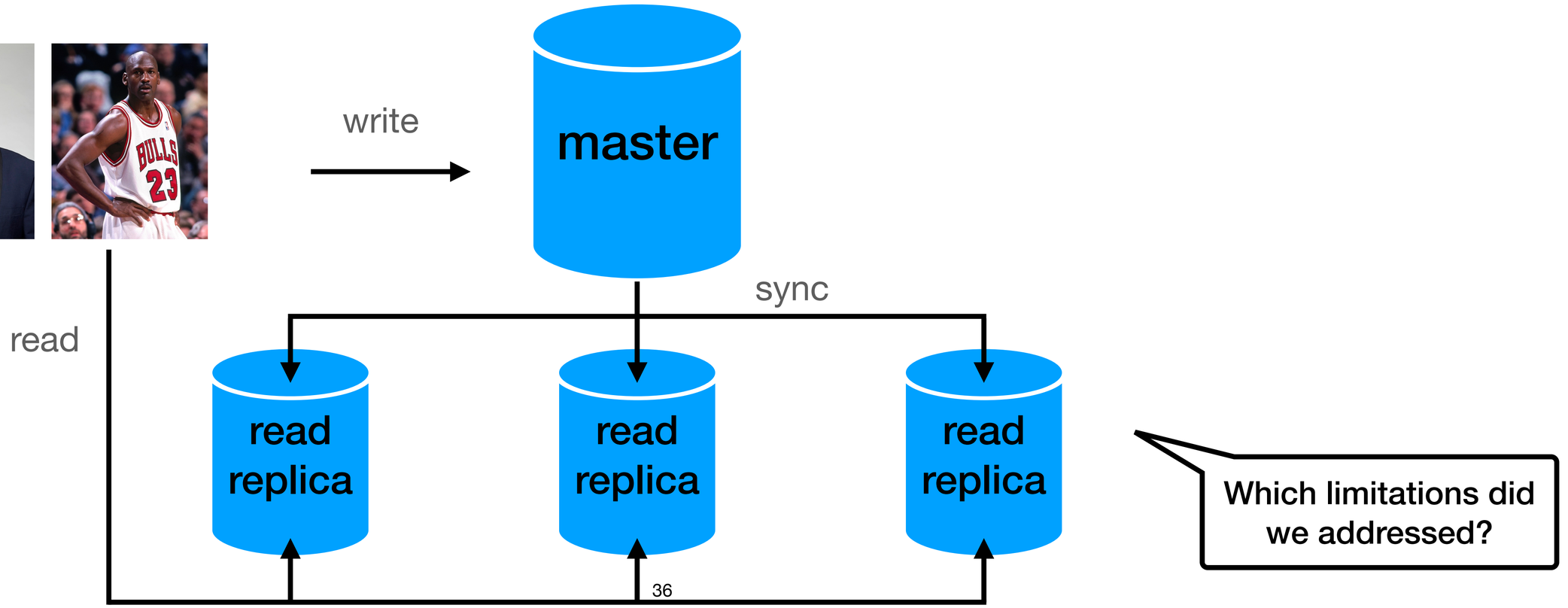
- On each write to the master we sync the replicas
- If we have a read query, we can use the replicas



RDBMS read replicas

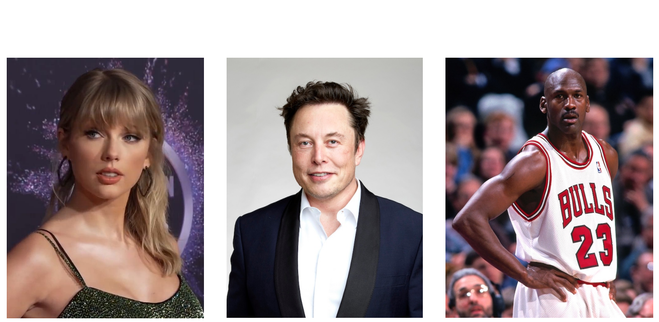
- On each write to the master we sync the replicas
- If we have a read query, we can use the replicas

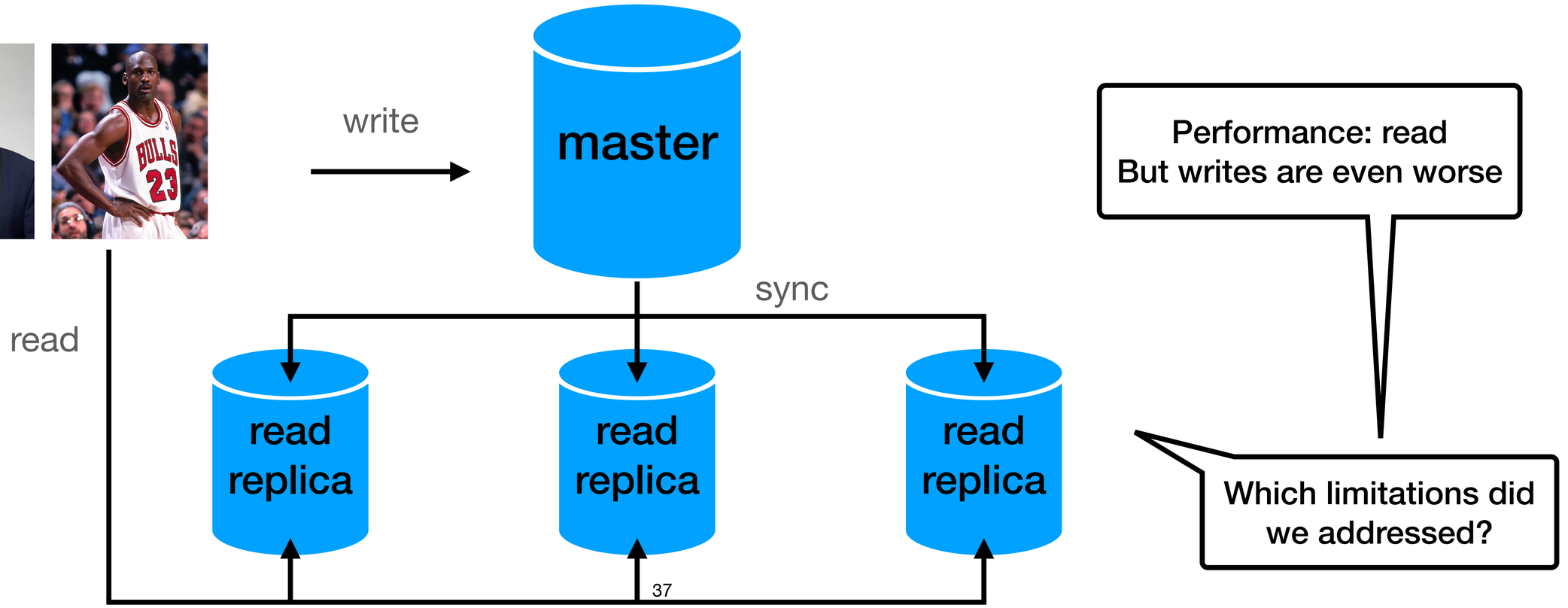




RDBMS read replicas

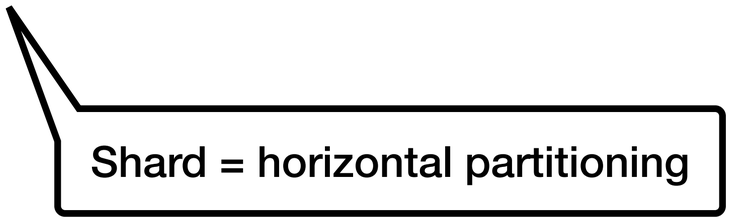
- On each write to the master we sync the replicas
- If we have a read query, we can use the replicas





Scaling RDBMS

- In practice, we use both methods
- Works great
  especially for data protection and up time
- Performance - until a point as we can't scale the master writes
  - * we can use manual data sharding / fragmentation to a limit
- So what do we do?



“Going distributed”

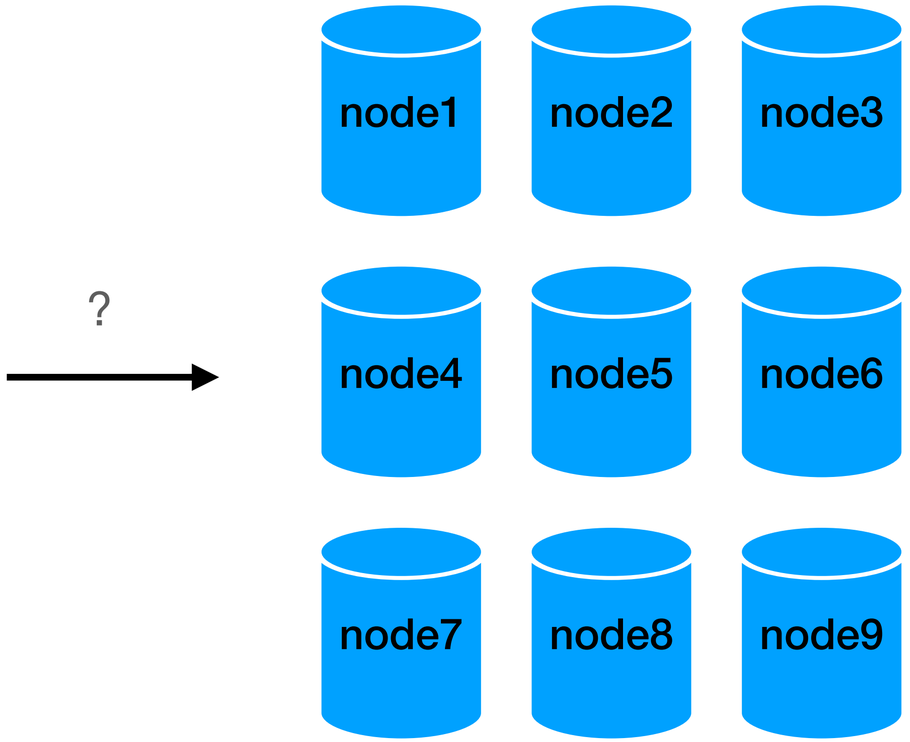
Disclaimer before going distributed

- RDBMS are great!
- Distributed databases do NOT replace RDBMS
- They are used for different use cases next to RDBMS

Going distributed

- **Not trivial**… :)

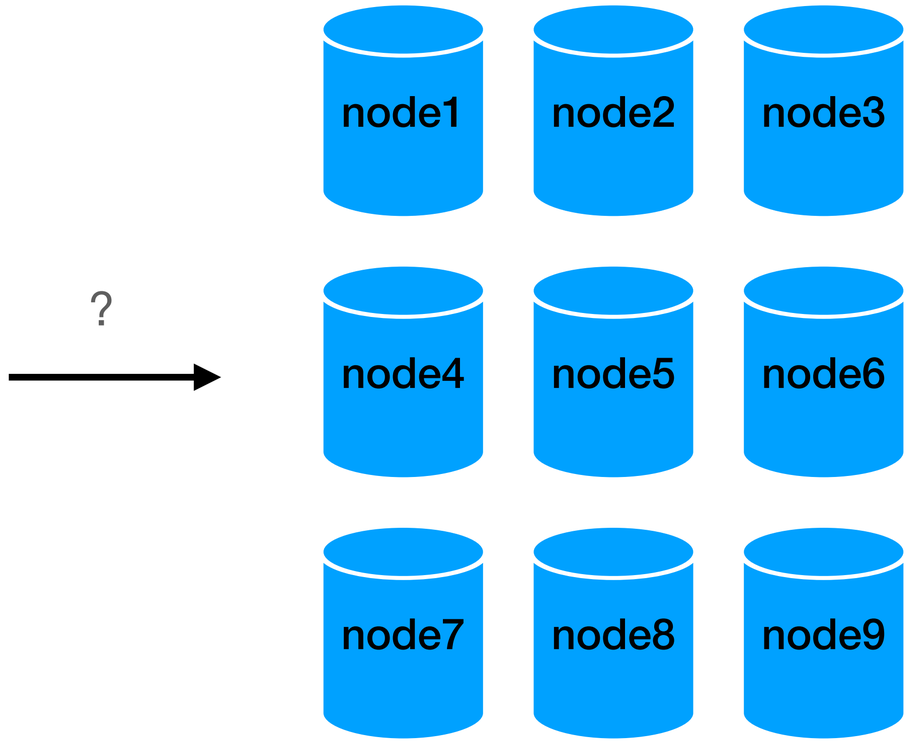
- Starting with:
  - Data fragmentation
  - Data distribution
  - Data replication



Going distributed

- **Not trivial**… :)

- Starting with:
  - **Data fragmentation**
  - Data distribution
  - Data replication



Data fragmentation

- How can we “break” a relation?
- How to implement reads / writes?

user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

Horizontal fragmentation

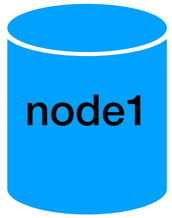
user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose an attribute
- Assign a “range” to each “node”

Horizontal fragmentation

user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose an attribute
- Assign a “range” to each “node”



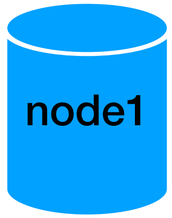
user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

user_id	fname	lname	city	country	account	brithdate
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984

Horizontal fragmentation

user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose an attribute
- Assign a “range” to each “node”



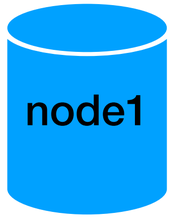
user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

user_id	fname	lname	city	country	account	brithdate
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984

Horizontal fragmentation

user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose an attribute
- Assign a “range” to each “node”



user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

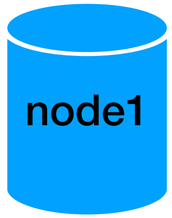
user_id	fname	lname	city	country	account	brithdate
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984

how do we implement reads and writes?

Horizontal fragmentation

user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose an attribute
- Assign a “range” to each “node”



user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

user_id	fname	lname	city	country	account	brithdate
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984

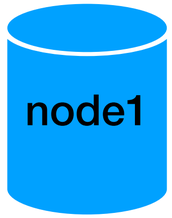
how do we implement reads and writes?

Is “SELECT * FROM users WHERE user_id = 101” valid?
How the DB finds the right node?

Horizontal fragmentation

user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose an attribute
- Assign a “range” to each “node”



user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

user_id	fname	lname	city	country	account	brithdate
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984

what will be the range if we choose `user_id` or `birthdate`?

Vertical fragmentation

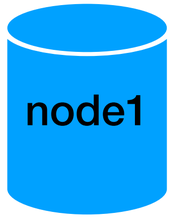
- Choose attributes
- Assign a “attributes” to each “node”

user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

Vertical fragmentation

user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose attributes
- Assign a “attributes” to each “node”



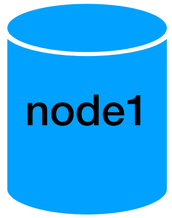
user_id	fname	lname
101	Rubi	Boim
102	Tova	Milo
103	Lebron	James
104	Michael	Jordan

user_id	city	country	account	brithdate
101	Tel Aviv	Israel	Normal	<null>
102	Tel Aviv	Israel	Premium	<null>
103	Los Angeles	USA	Premium	30/12/1984
104	Chicago	USA	Normal	17/02/1963

Vertical fragmentation

user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose attributes
- Assign a “attributes” to each “node”



user_id	fname	lname
101	Rubi	Boim
102	Tova	Milo
103	Lebron	James
104	Michael	Jordan

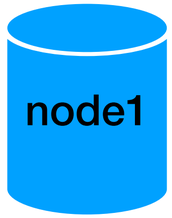
user_id	city	country	account	brithdate
101	Tel Aviv	Israel	Normal	<null>
102	Tel Aviv	Israel	Premium	<null>
103	Los Angeles	USA	Premium	30/12/1984
104	Chicago	USA	Normal	17/02/1963

how do we implement reads and writes?

Vertical fragmentation

user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

- Choose attributes
- Assign a “attributes” to each “node”



user_id	fname	lname
101	Rubi	Boim
102	Tova	Milo
103	Lebron	James
104	Michael	Jordan

What happen if the key is defined by 3 different attributes?

user_id	city	country	account	brithdate
101	Tel Aviv	Israel	Normal	<null>
102	Tel Aviv	Israel	Premium	<null>
103	Los Angeles	USA	Premium	30/12/1984
104	Chicago	USA	Normal	17/02/1963

Vertical fragmentation

user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

SIDE NOTE

- There is a class of databases that is called “**columnar database**”
- Used mainly in data warehouse

user_id	fname
101	Rubi
102	Tova
103	Lebron
104	Michael

node1

user_id	lname
101	Boim
102	Milo
103	James
104	Jordan

node2

...

user_id	birthdate
101	<null>
102	<null>
103	30/12/1984
104	17/02/1963

nodeN

Vertical fragmentation

user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

SIDE NOTE

- There is a class of databases that is called “**columnar database**”
- Used mainly in data warehouse

Can be highly compressible.
Question - for which attributes?

user_id	fname
101	Rubi
102	Tova
103	Lebron
104	Michael

node1

user_id	lname
101	Boim
102	Milo
103	James
104	Jordan

node2

...

user_id	birthdate
101	<null>
102	<null>
103	30/12/1984
104	17/02/1963

nodeN

Vertical fragmentation

user_id	fname	lname	city	country	account	brithdate
101	Rubi	Boim	Tel Aviv	Israel	Normal	<null>
102	Tova	Milo	Tel Aviv	Israel	Premium	<null>
103	Lebron	James	Los Angeles	USA	Premium	30/12/1984
104	Michael	Jordan	Chicago	USA	Normal	17/02/1963

SIDE NOTE

- There is a class of databases that is called “**columnar database**”
- Used mainly in data warehouse

Popular databases:
Redshift, BigQuery, Apache Druid, Maria DB

user_id	fname
101	Rubi
102	Tova
103	Lebron
104	Michael

node1

user_id	lname
101	Boim
102	Milo
103	James
104	Jordan

node2

...

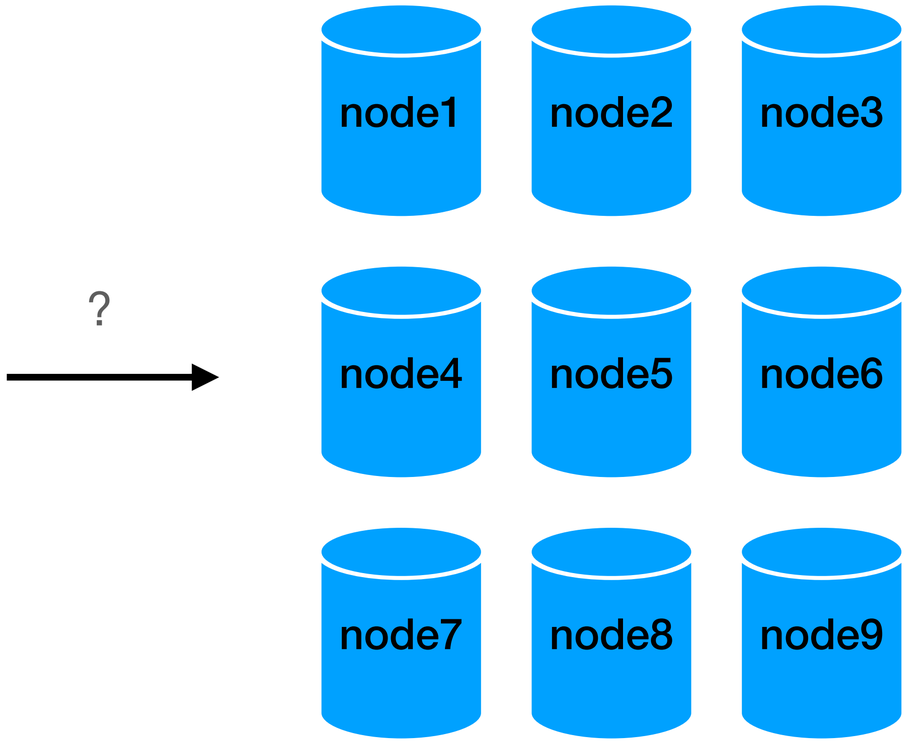
user_id	birthdate
101	<null>
102	<null>
103	30/12/1984
104	17/02/1963

nodeN

Going distributed

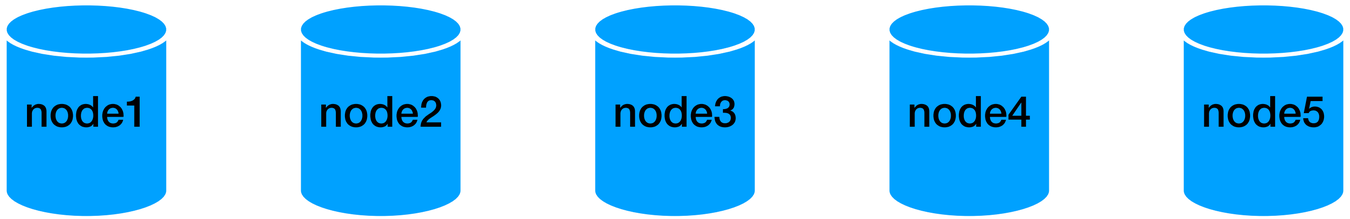
- **Not trivial**… :)

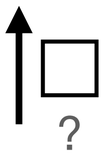
- Starting with:
  - Data fragmentation
  - **Data distribution**
  - Data replication



Data distribution

- How can the DB decide where the data is located?



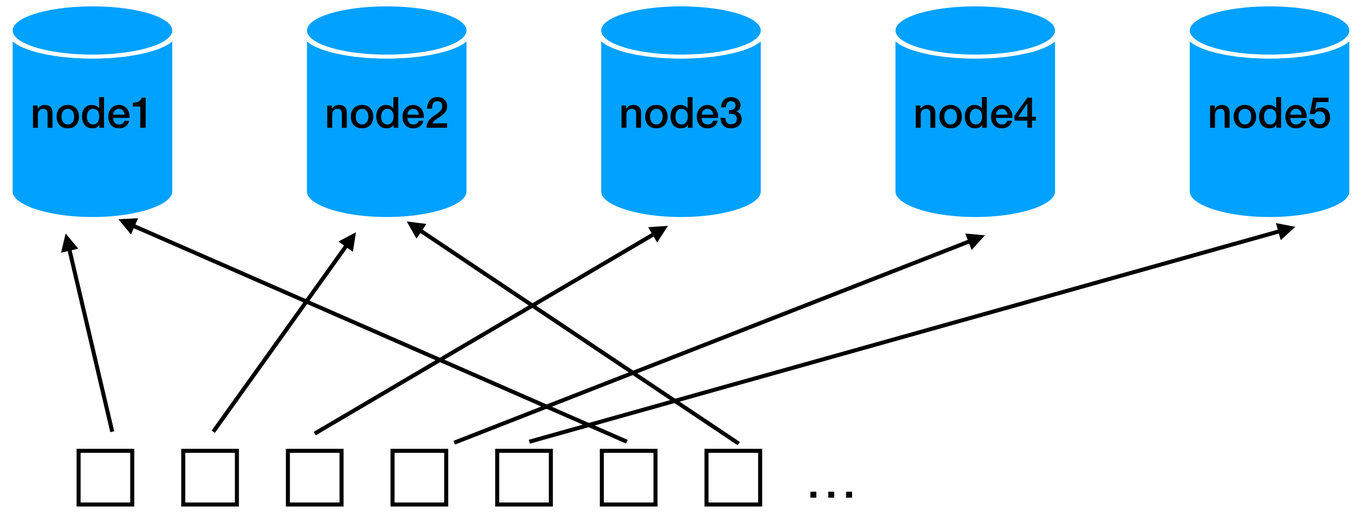


INSERT INTO users VALUES(x,y,z)

add new data /
query existing data

Data distribution (1) - Round robin

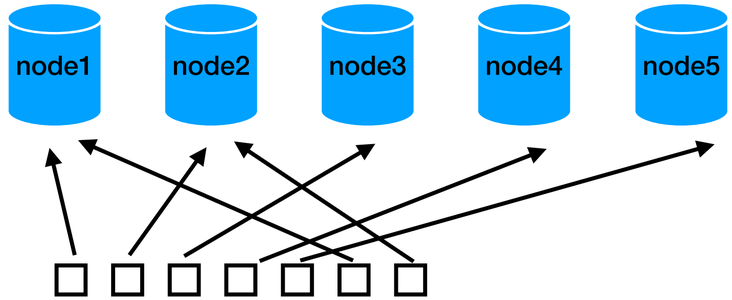
- Or random…



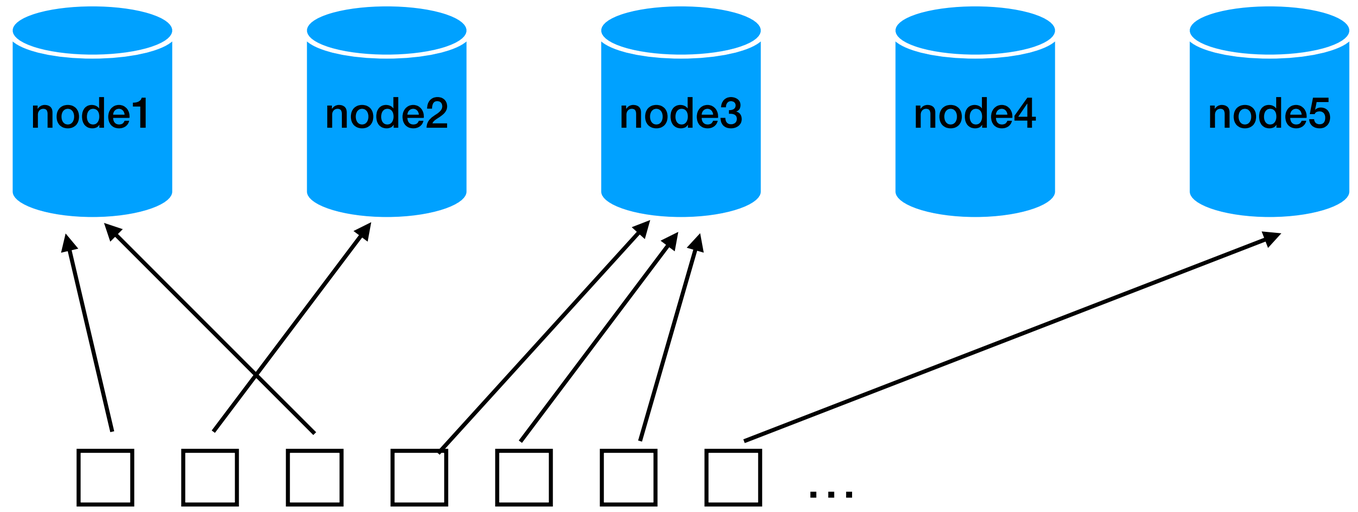
pros / cons?

Data distribution (1) - Round robin

- ☑ “Uniform” distribution
- ☐ The “driver” needs to save the last node for each different query
- ☐ To query we need to save the node “used” for each “package"
  - we need a DB to implement a DB…

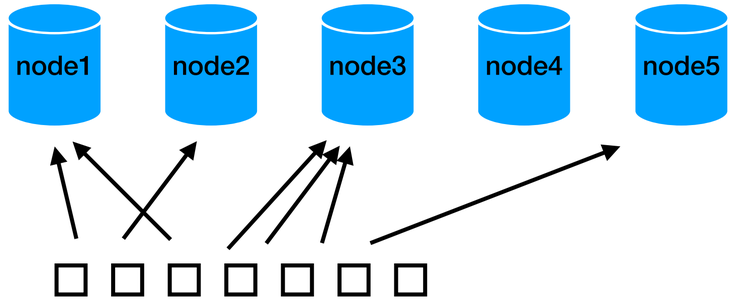


Data distribution (2) - Hashing



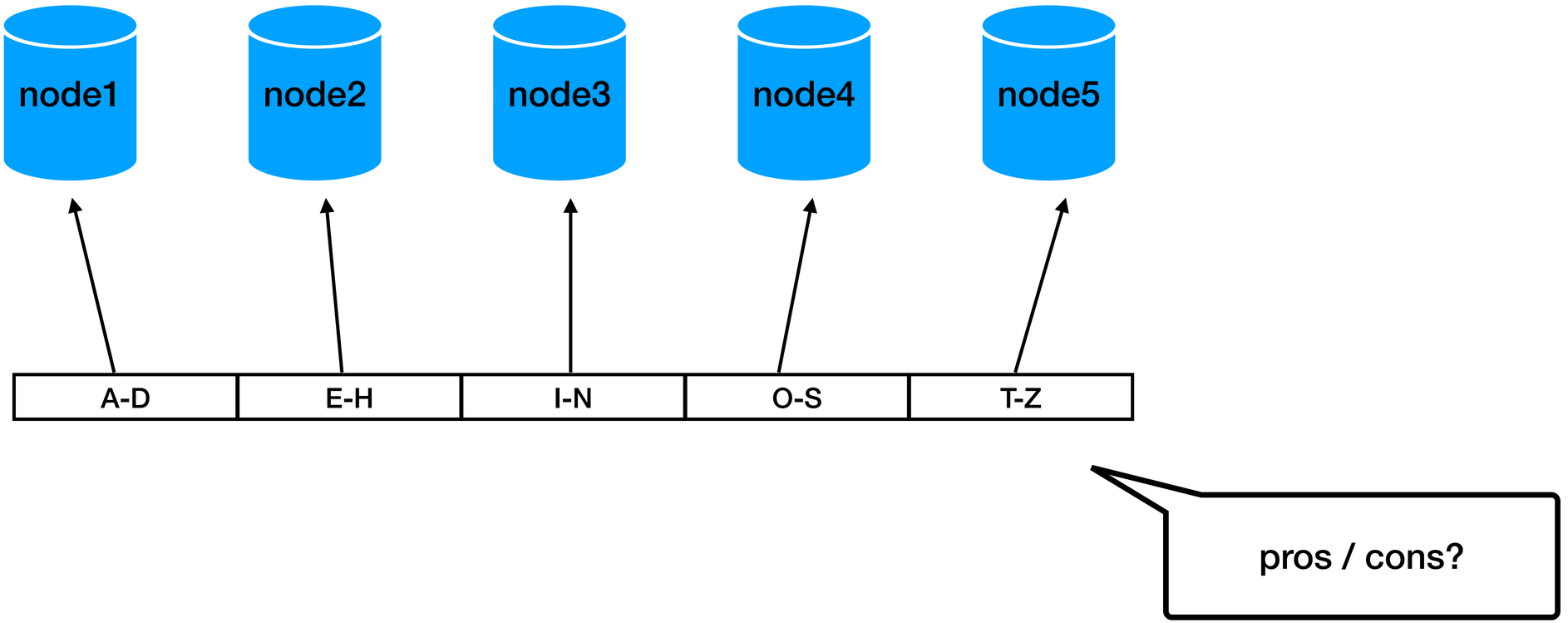
pros / cons?

Data distribution (2) - Hashing

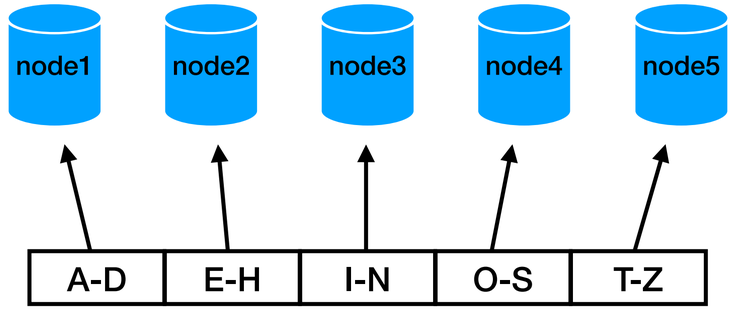


- Similar to Round robin:
  To query we need to save the node “used” for each “package” (or hash)
  - we need a DB to implement a DB…

Data distribution (3) - Range

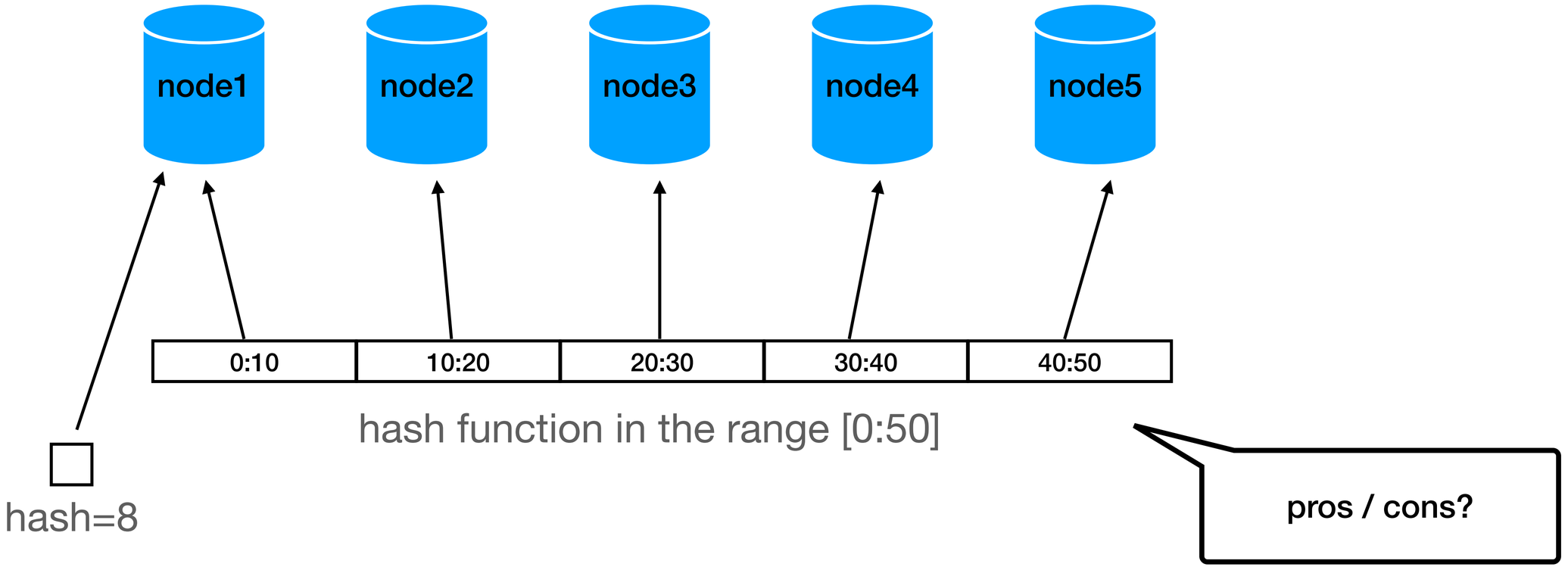


Data distribution (3) - Range

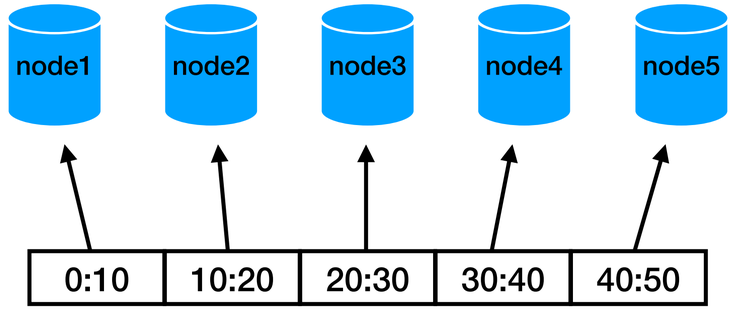


- Only the range mapping is needed to be saved
  - No need for an “extra DB”
    but we still need to store them
    (very little information - can be cached on the driver)
- Works different for strings, dates, int…
- Data distribution
  most names starts with “A” for example

Data distribution (4) - Range on hashes



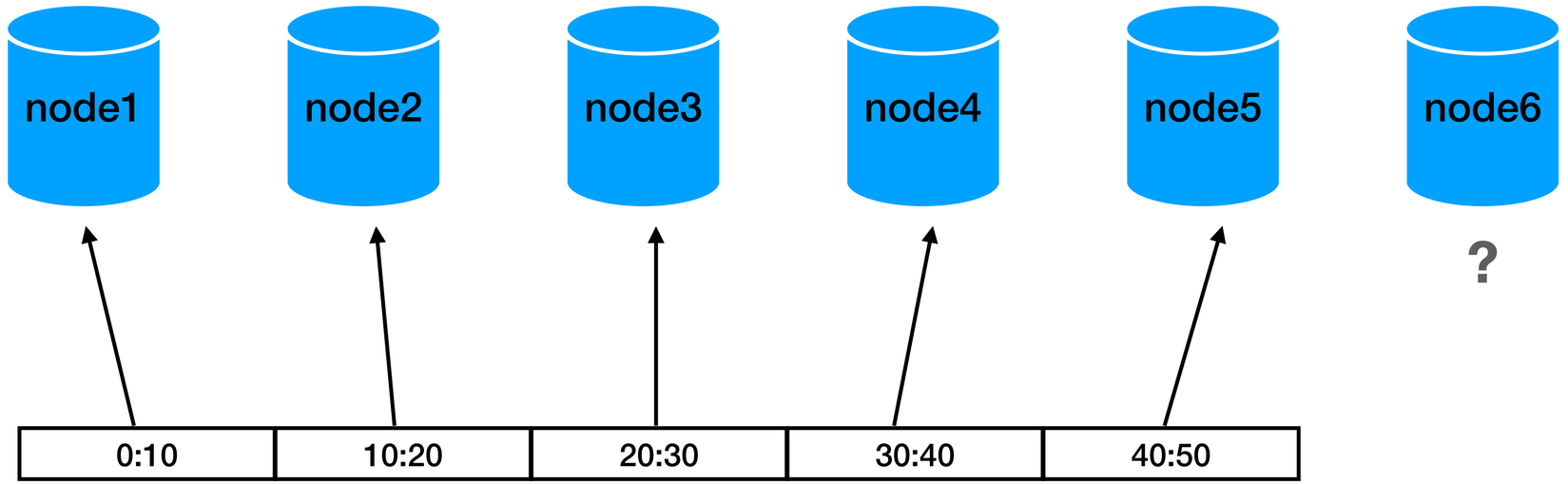
Data distribution (4) - Range on hashes



- Only the range mapping is needed to be saved
  - No need for an “extra DB”
    but we still need to store them
    (very little information - can be cached on the driver)
- Data agnostic
  assuming the hash function works properly
- Data distribution
  assuming the hash function works properly

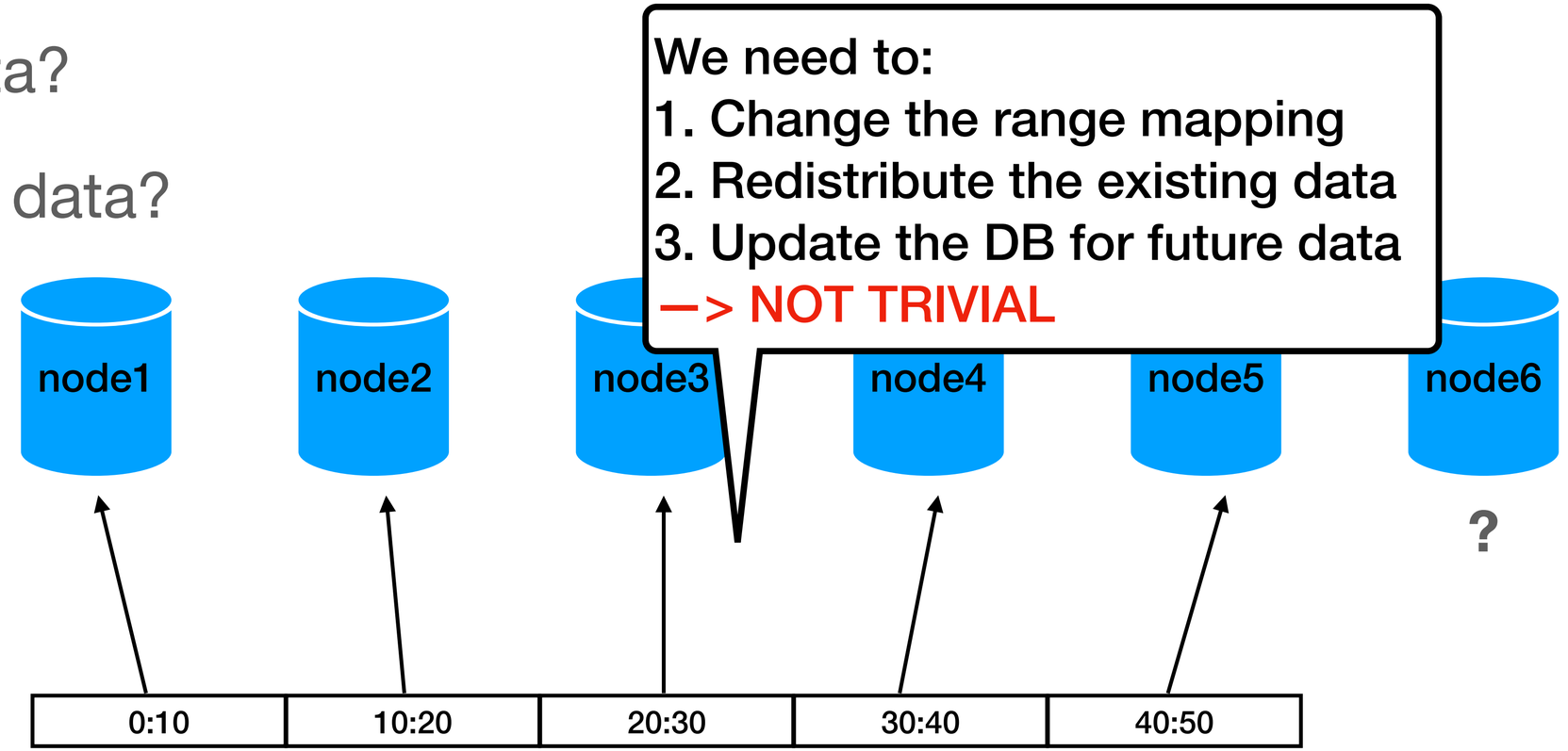
Data distribution - scaling

- What happens if we want to add a node?
  - new data?
  - existing data?



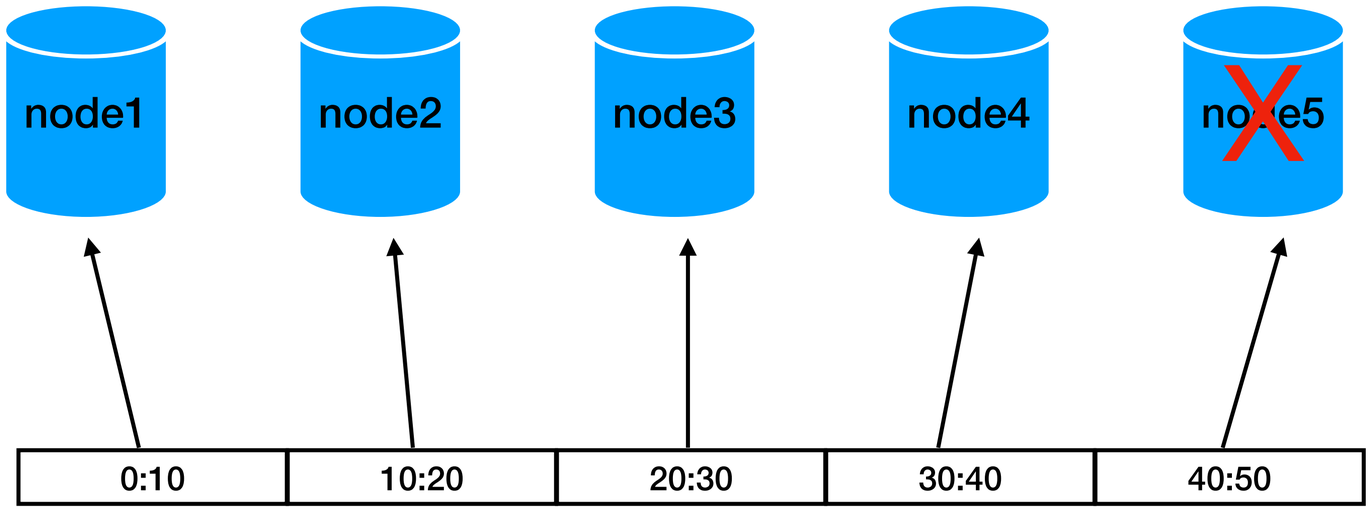
Data distribution - scaling

- What happens if we want to add a node?
  - new data?
  - existing data?



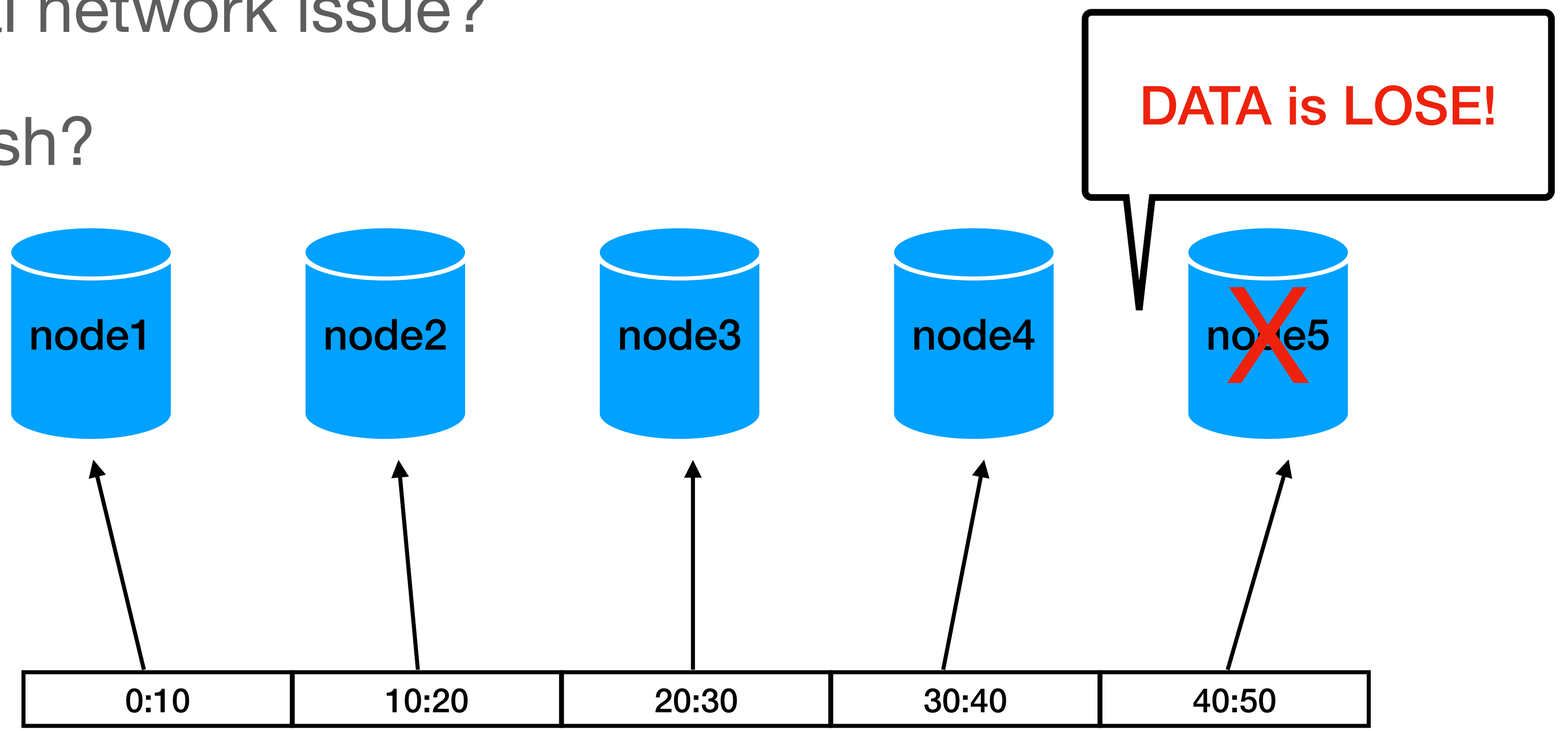
Stuff happens

- What happens if a node fails?
  - temporal network issue?
  - disk crash?



Stuff happens

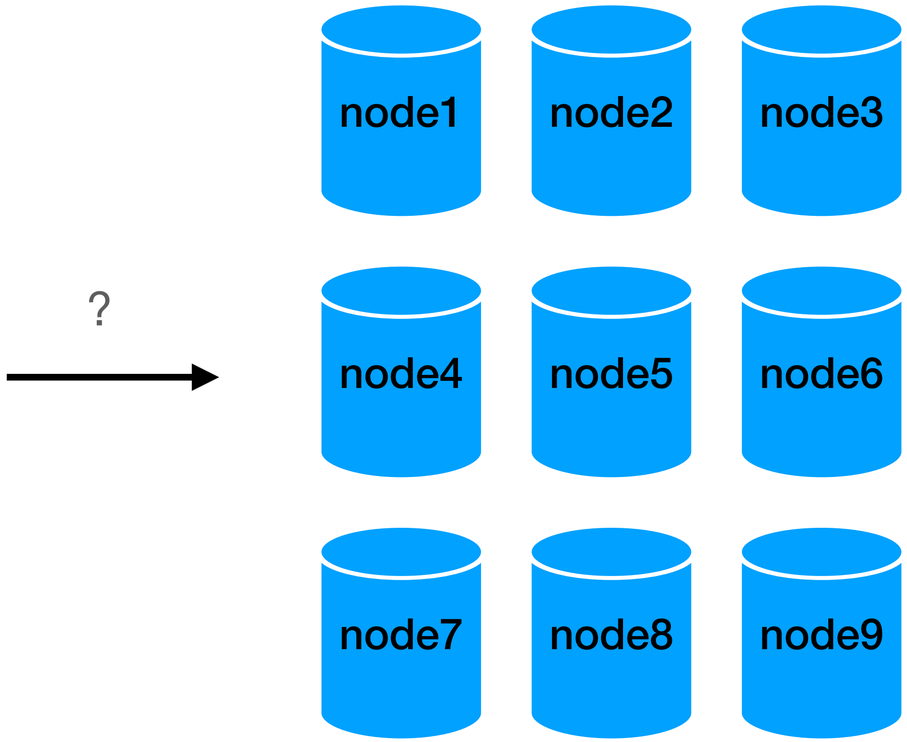
- What happens if a node fails?
  - temporal network issue?
  - disk crash?



Going distributed

- **Not trivial**… :)

- Starting with:
  - Data fragmentation
  - Data distribution
  - **Data replication**



Data replication

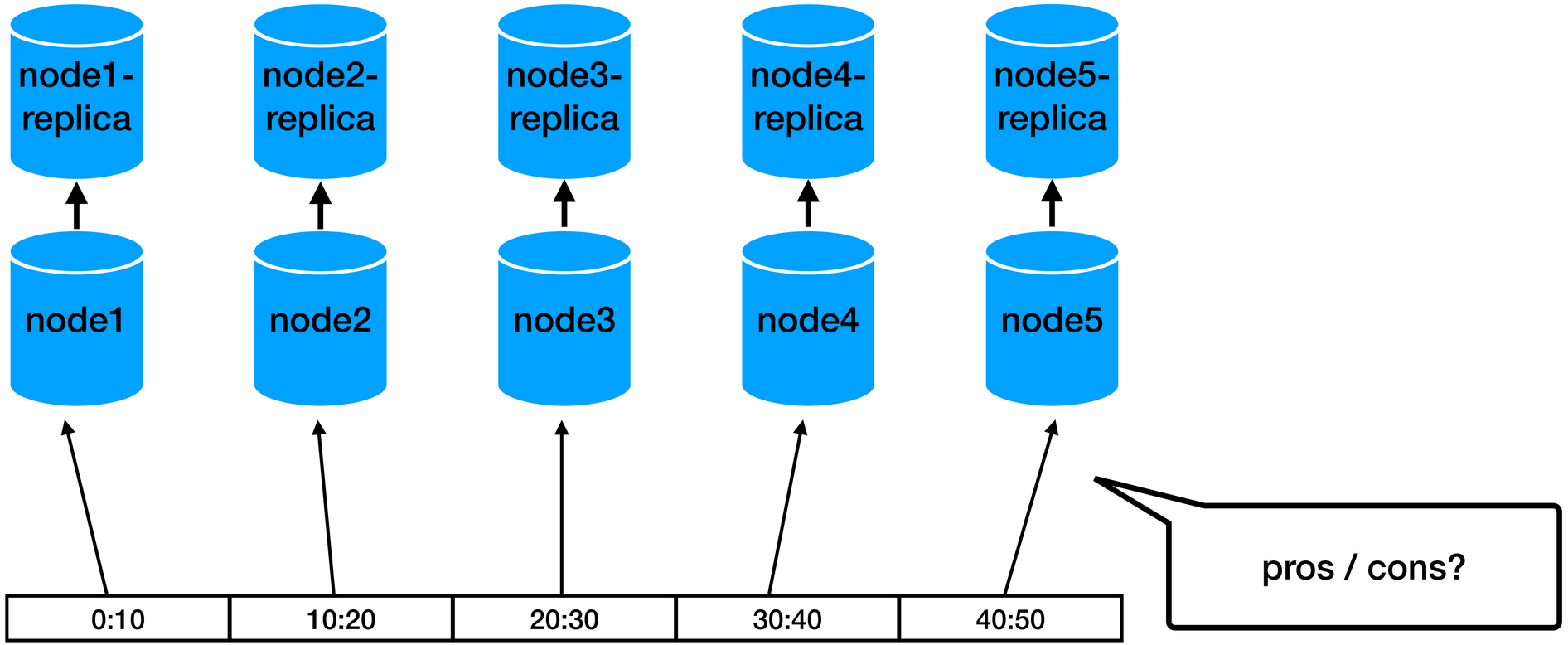
- Replicate the data on more than one node
  and more than one data center

Solves 2 issues

- On errors (network / hardware) we have a backup
- On Spikes we can utilize more than 1 node

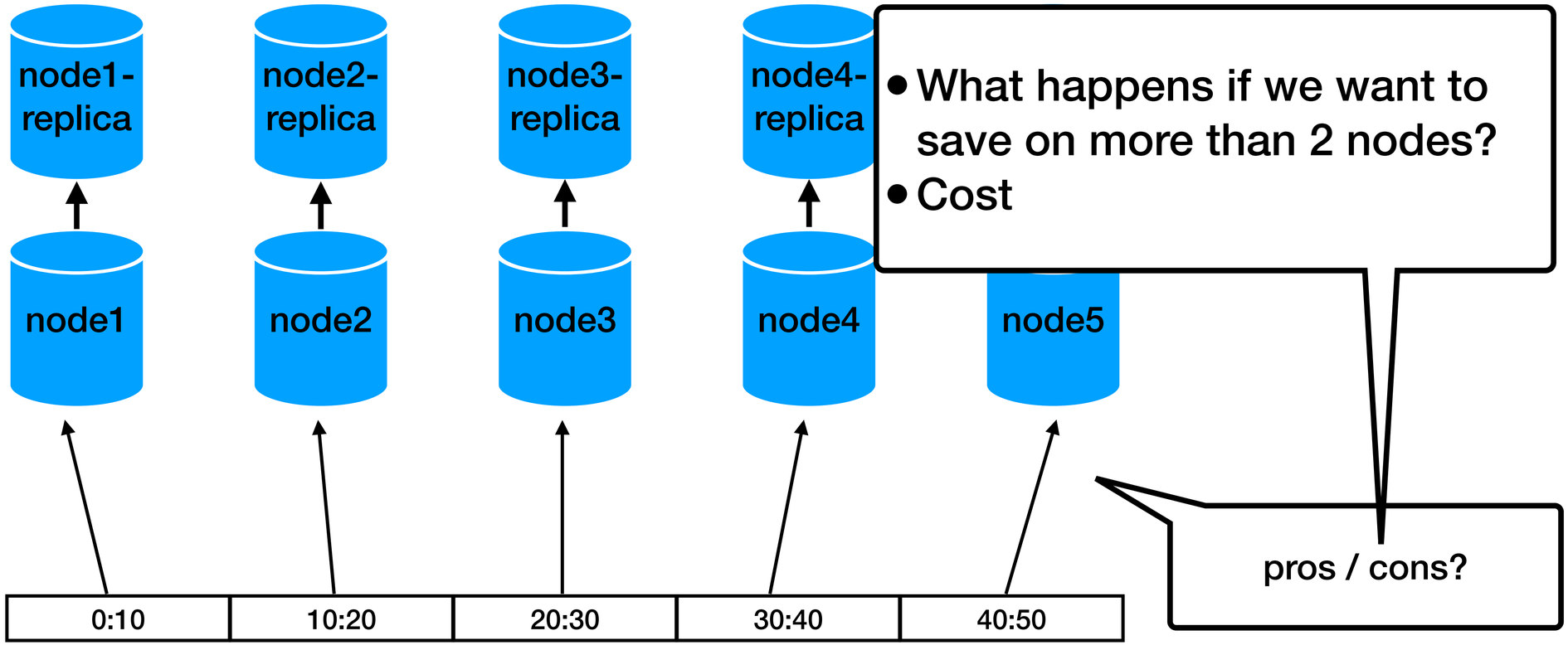
Data replication

- Read replicas + master slave?



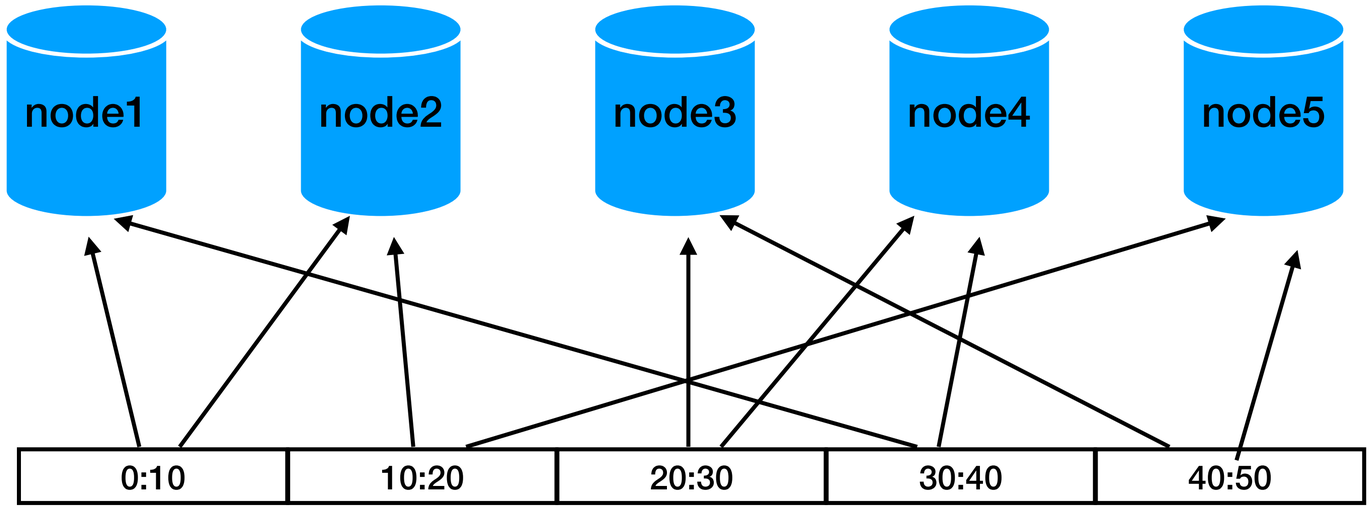
Data replication

- Read replicas + master slave?



Data replication

- (re)distribute among all nodes

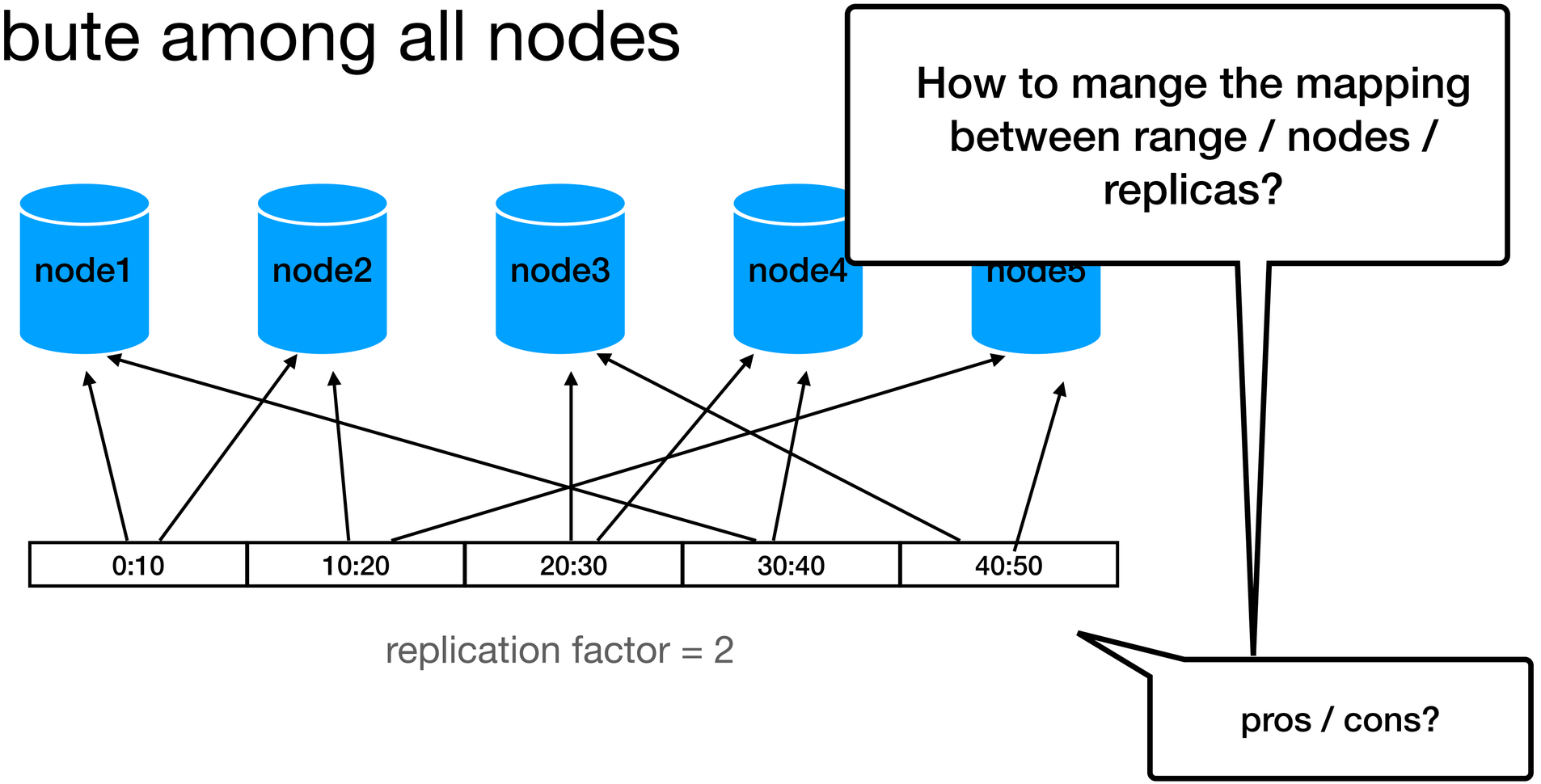


replication factor = 2

pros / cons?

Data replication

- (re)distribute among all nodes



Spoiler alert

- Get ready for the “Dynamo" lecture

We are just getting started

- Managing the fragmentation, distribution and replication of the data is a hard problem
- We also need to support:
  - **C**onsistency
    Every read receives the most recent write or an error
  - **A**vailability
    Every request receives a (non-error) response,
    without the guarantee that it contains the most recent write
  - **P**artition tolerance
    The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network