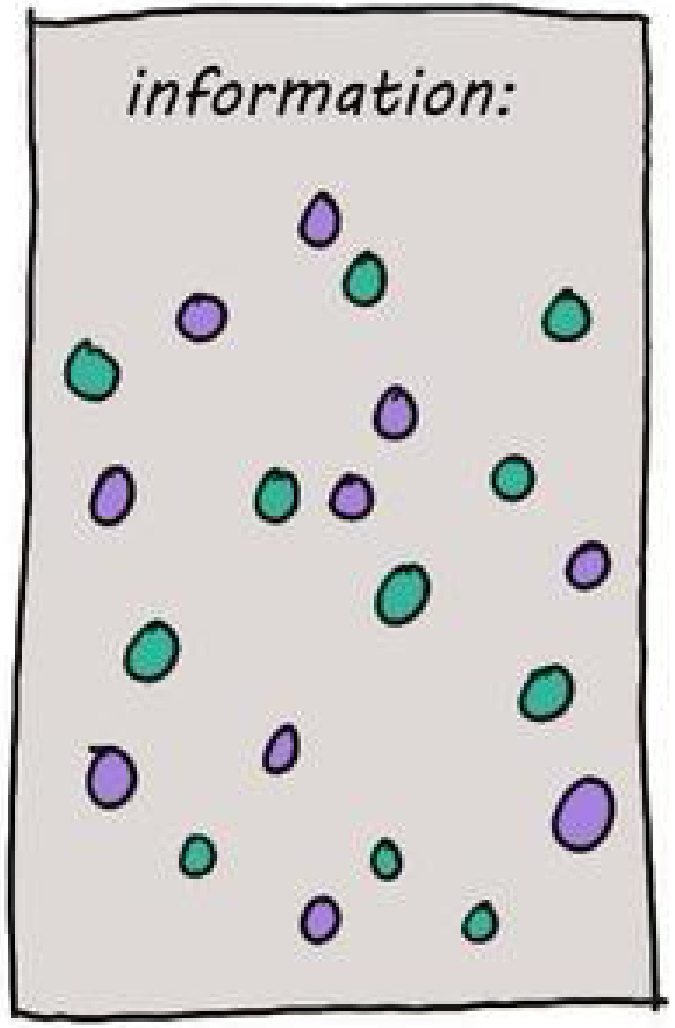
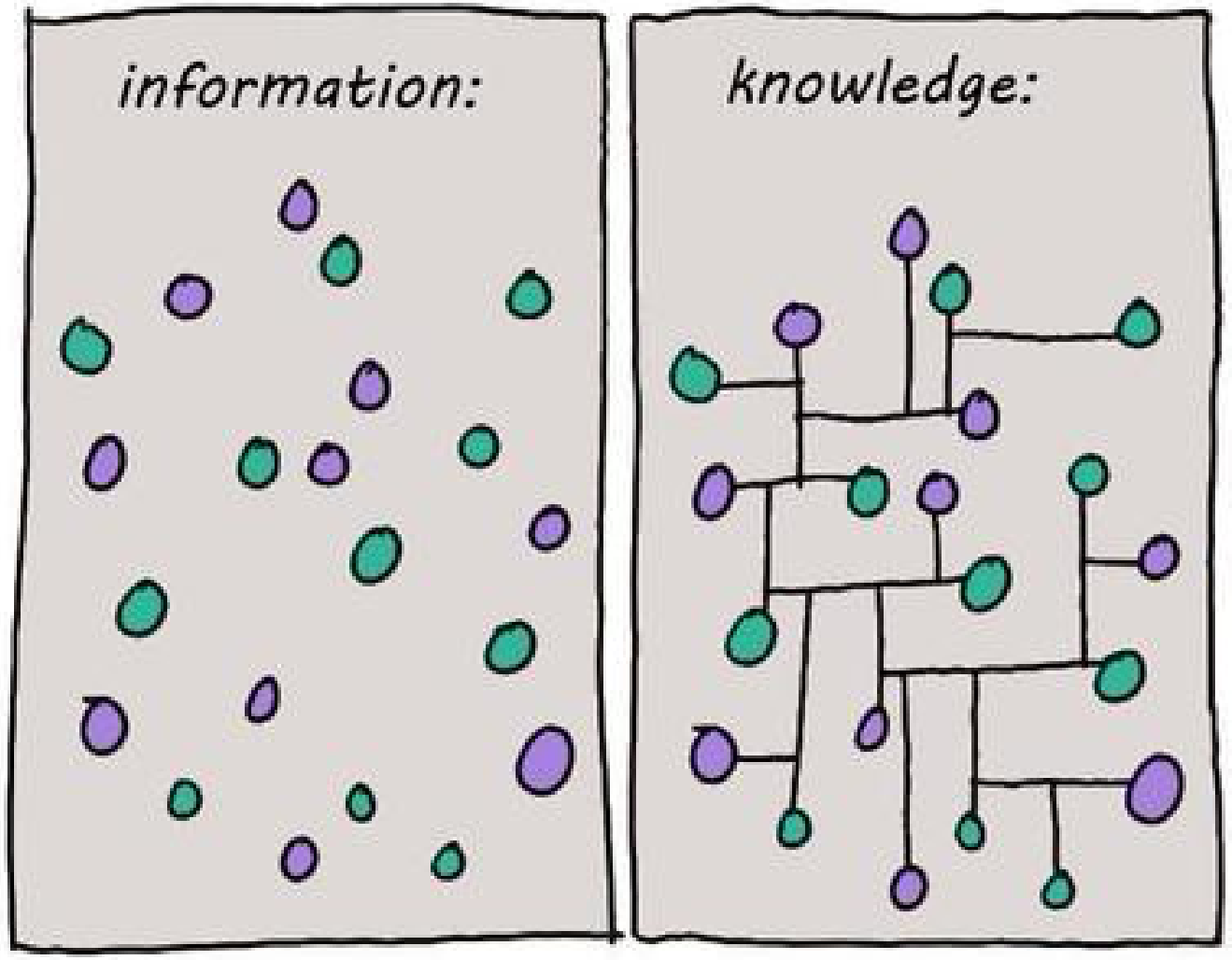
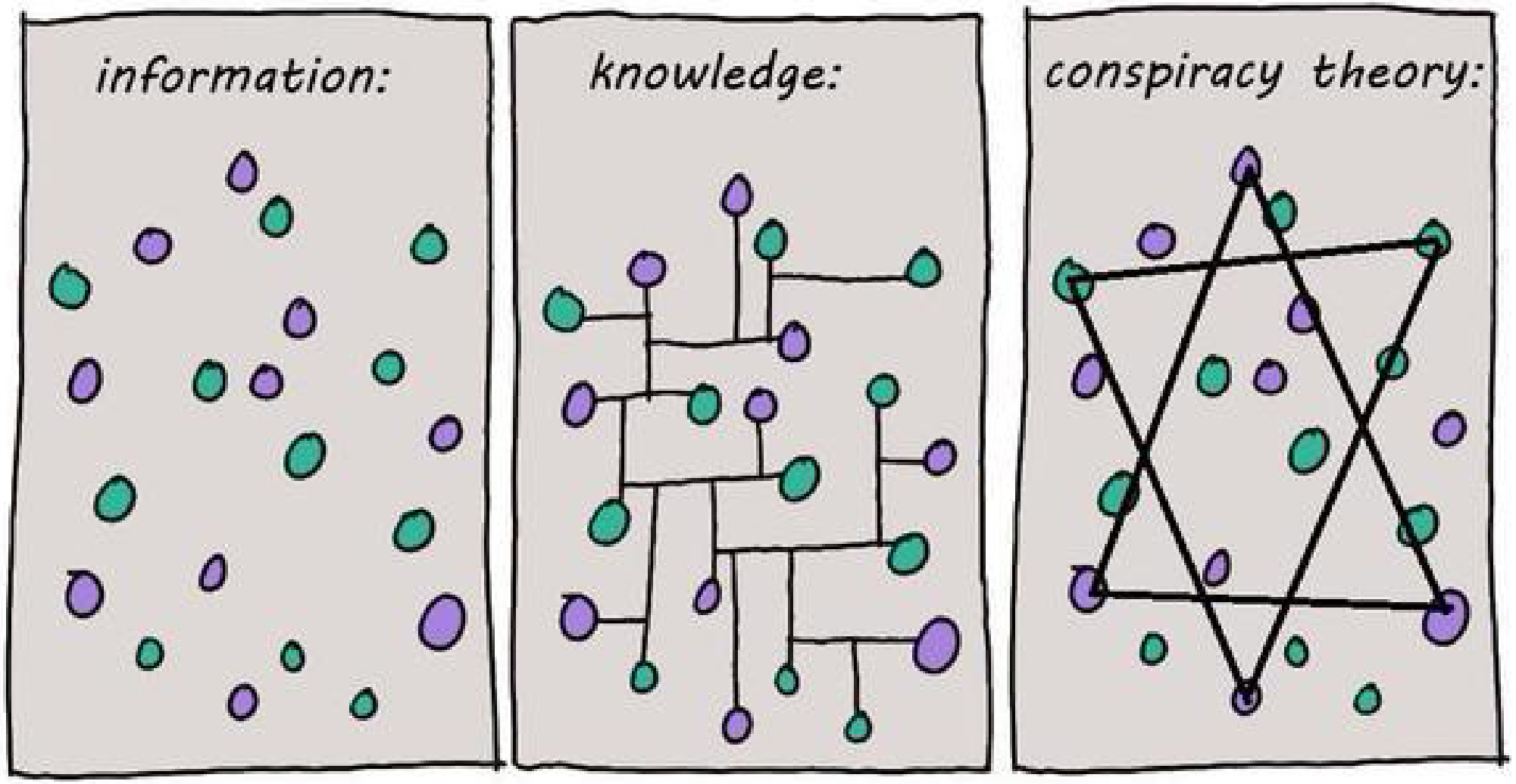
Elixir and Money

Tomasz Kowal

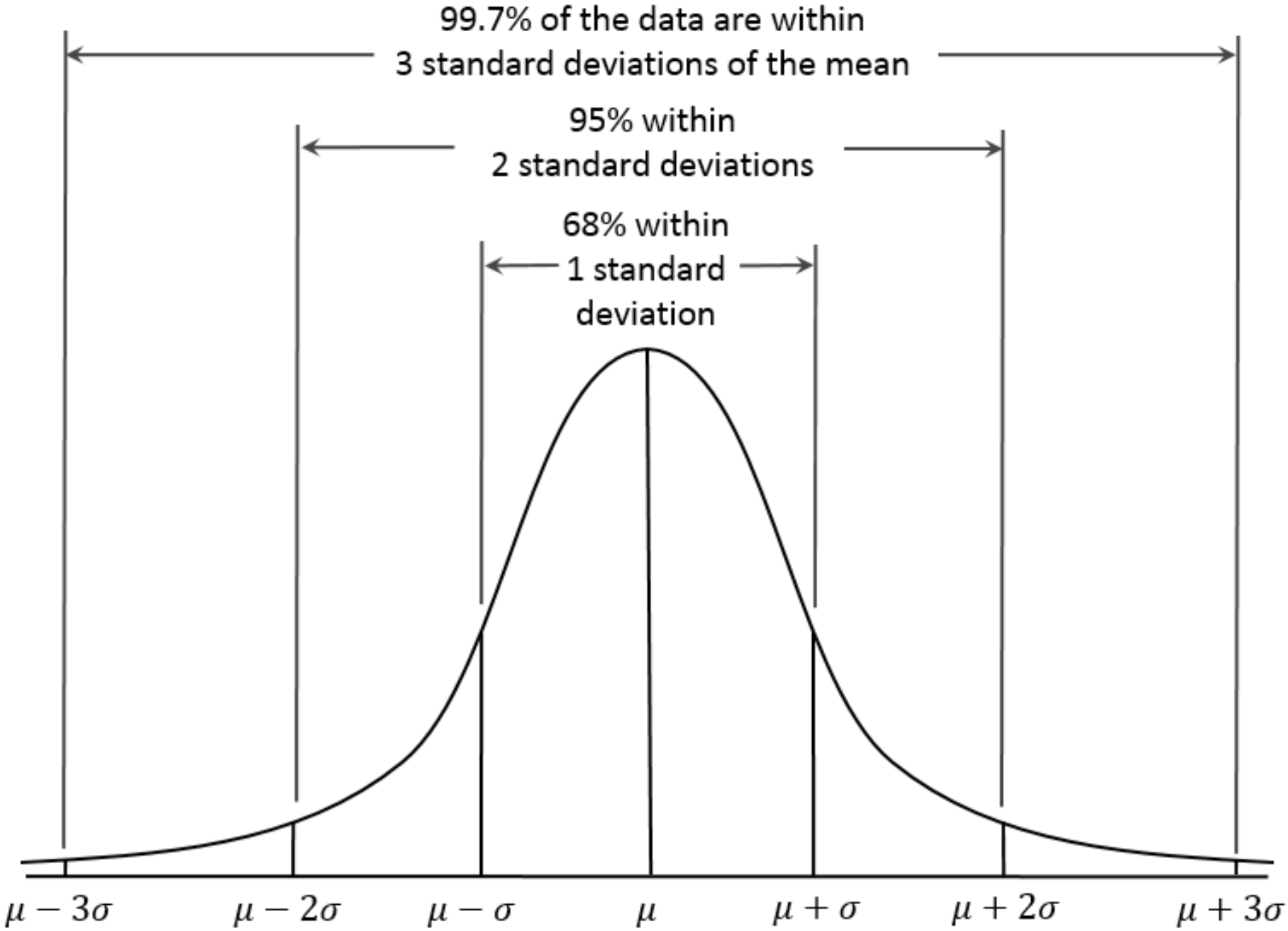
ClubCollect


information:


information:
knowledge:



Source: unknown



Source: Wikipedia

ACID vs BASE

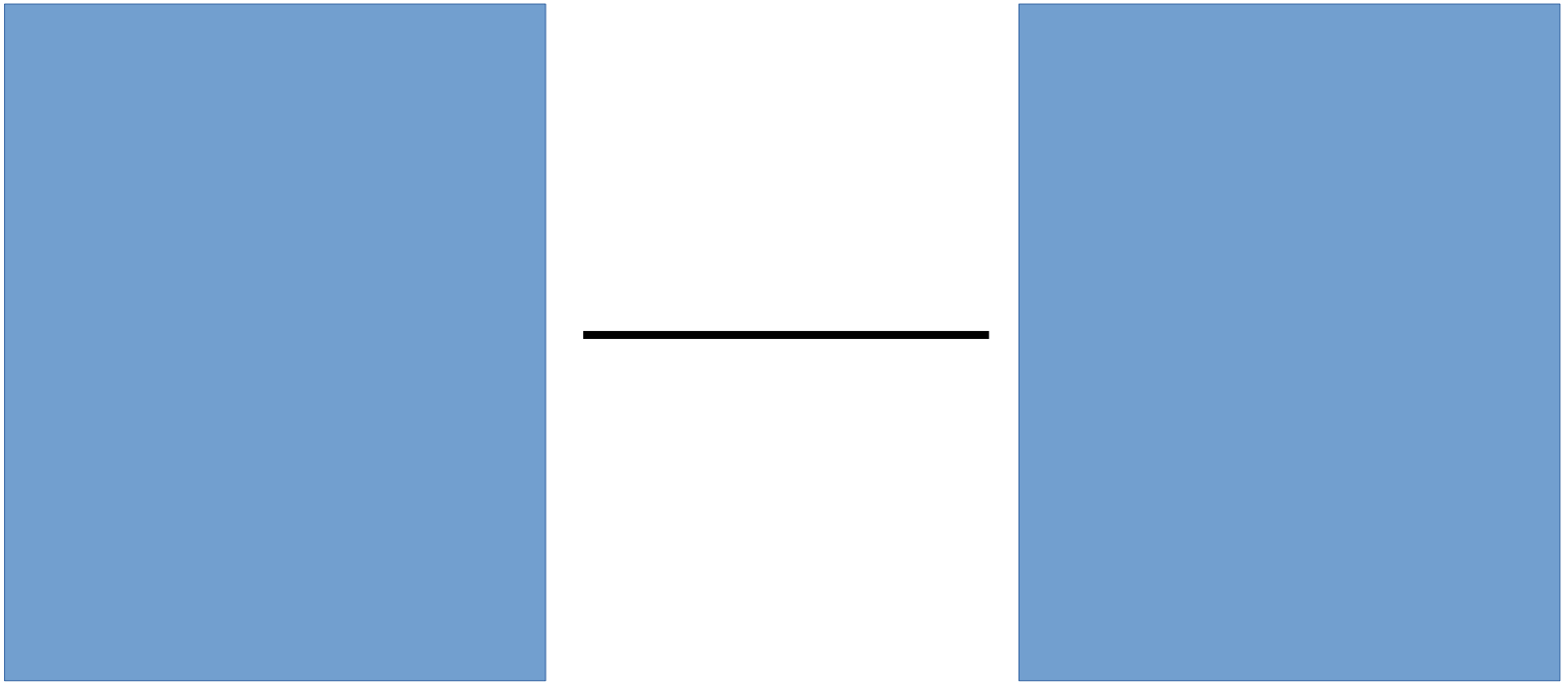
Atomicity
Consistency
Isolation
Durability

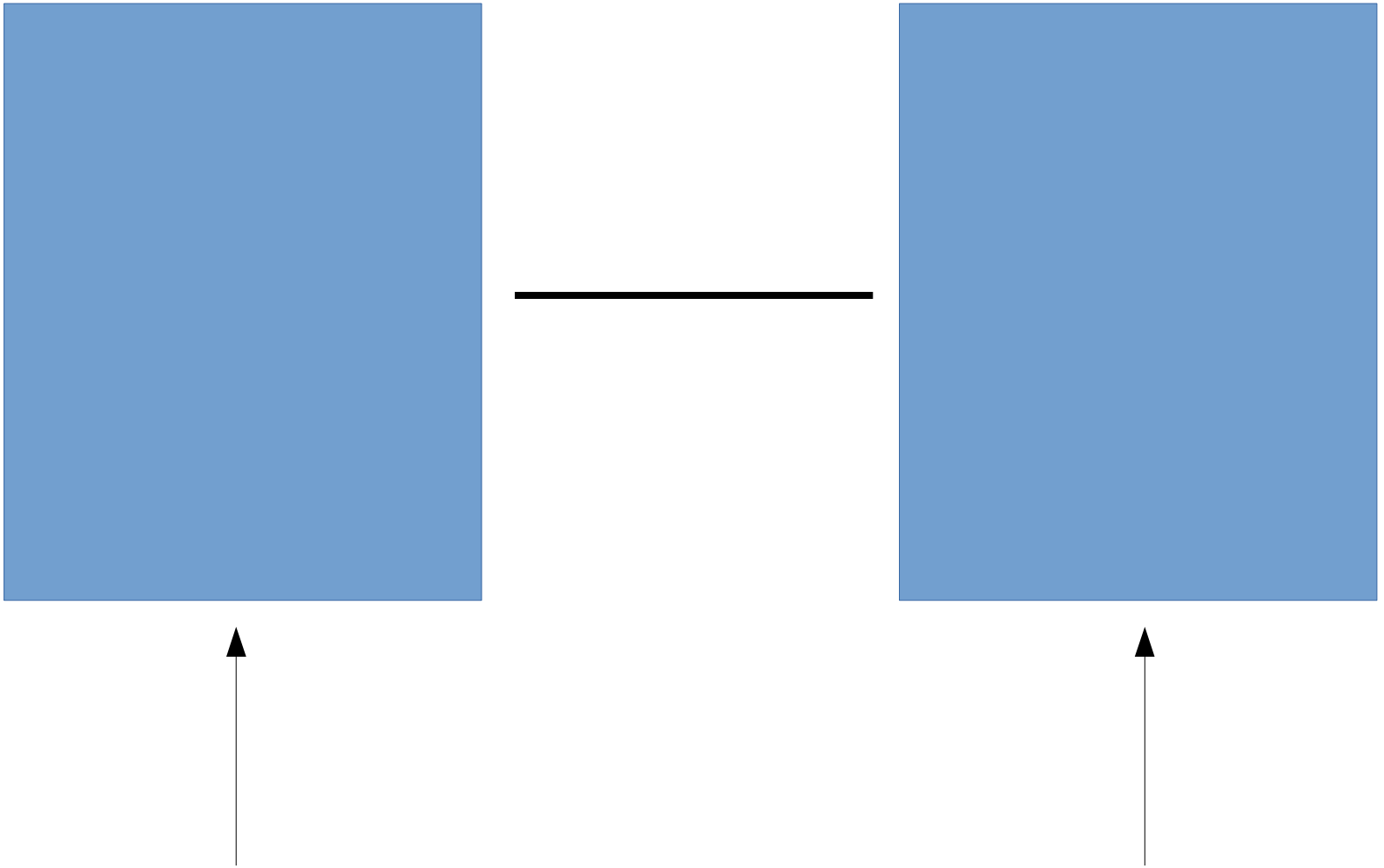
Consistency in ACID

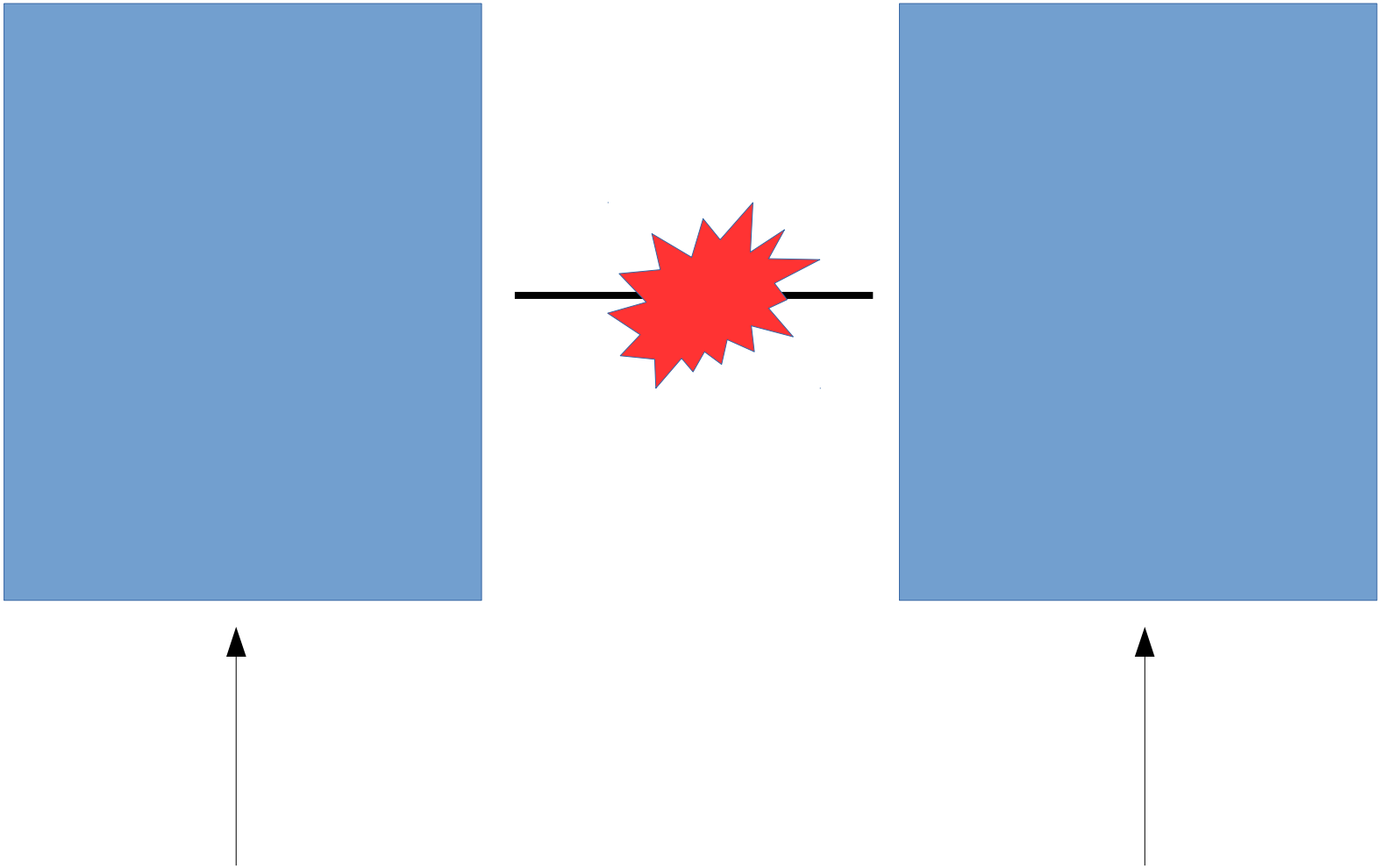
any transaction will bring the database from one valid state to another

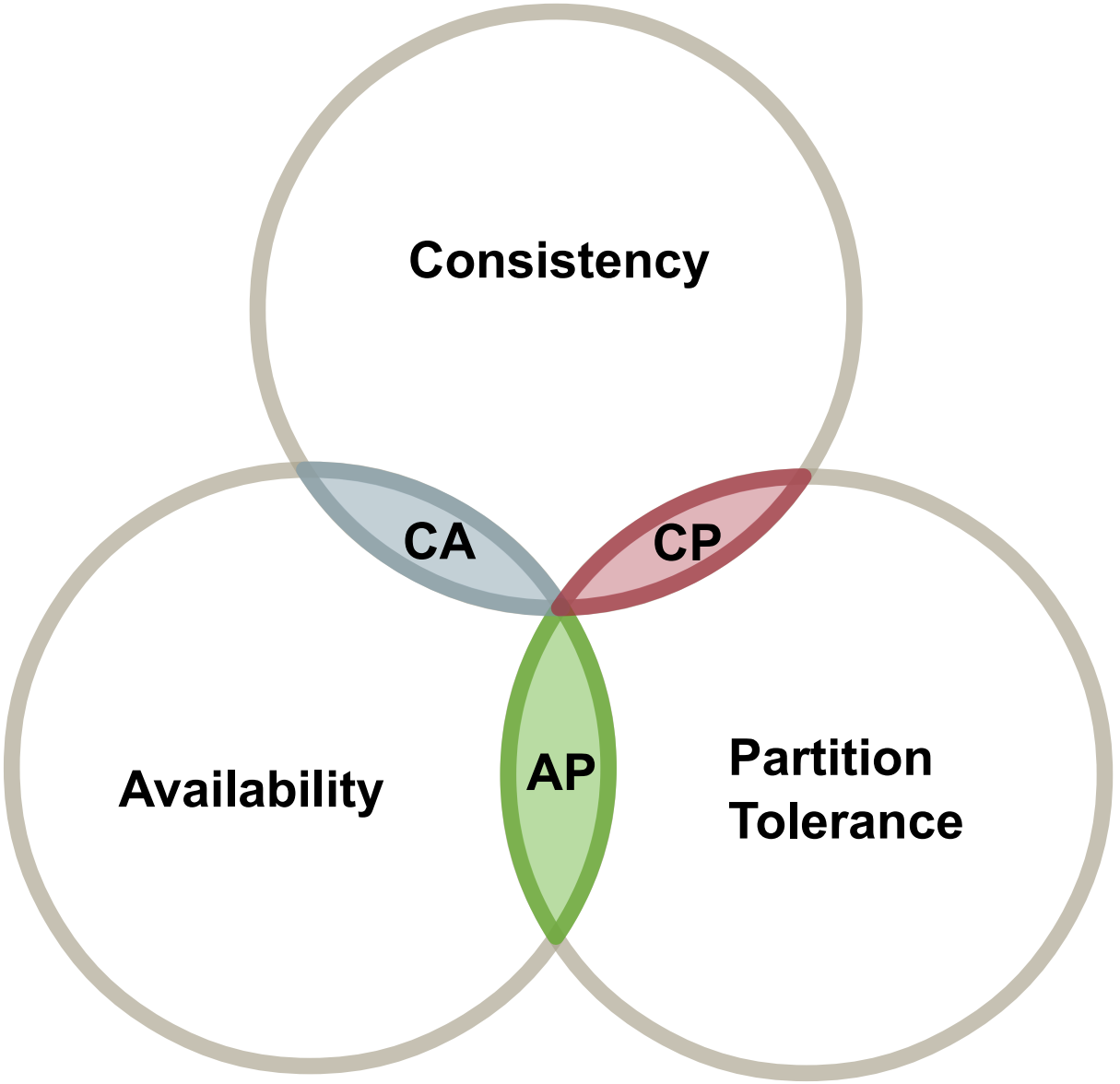
Isolation in ACID

- concurrent execution of transactions results in a system state that would be obtained if transactions were executed sequentially
- transaction will behave as if it is the only operation being performed upon the database




Write A
Write B


Write A
Write B

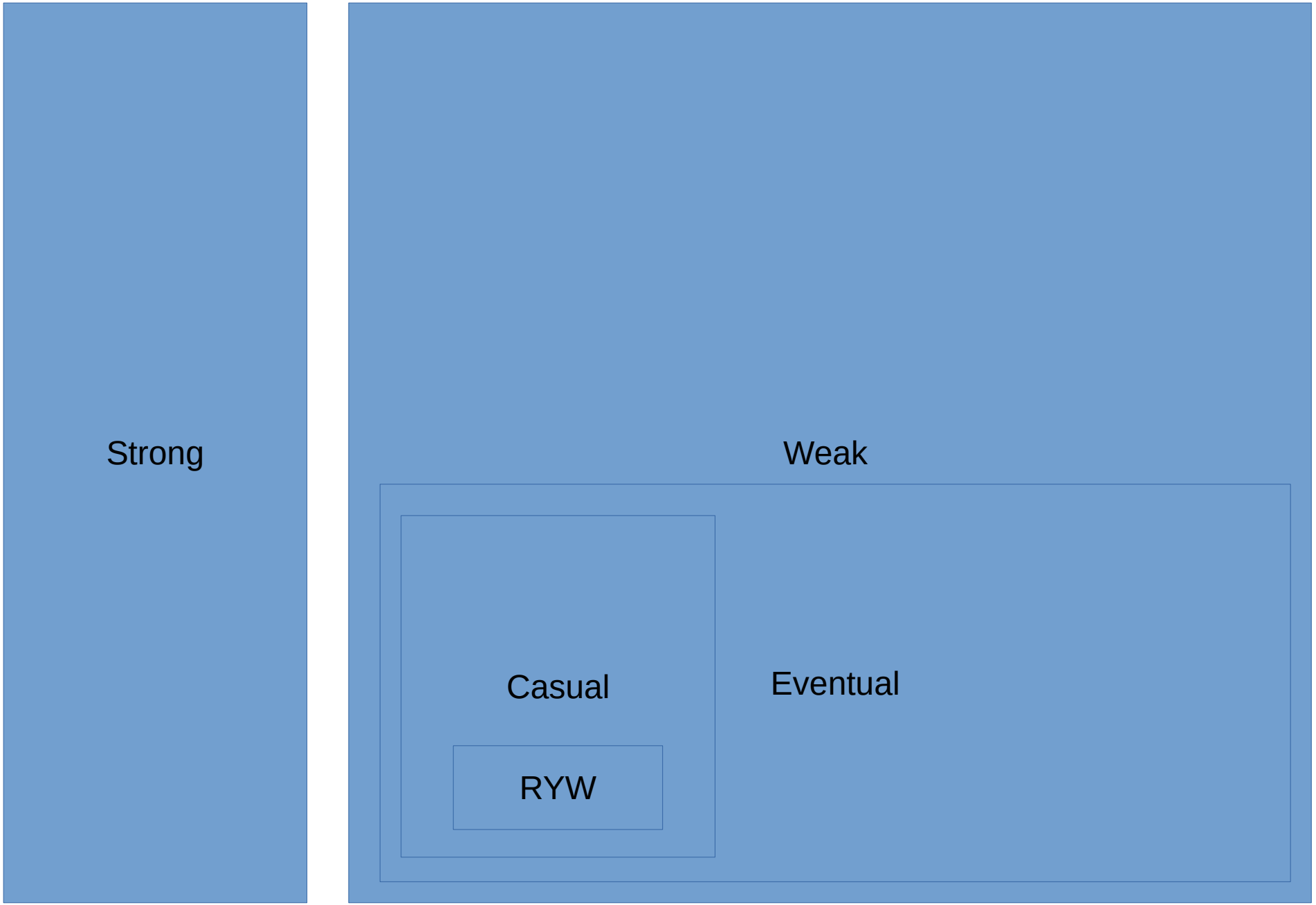

Consistency
CA
CP
Availability
AP
Partition
Tolerance

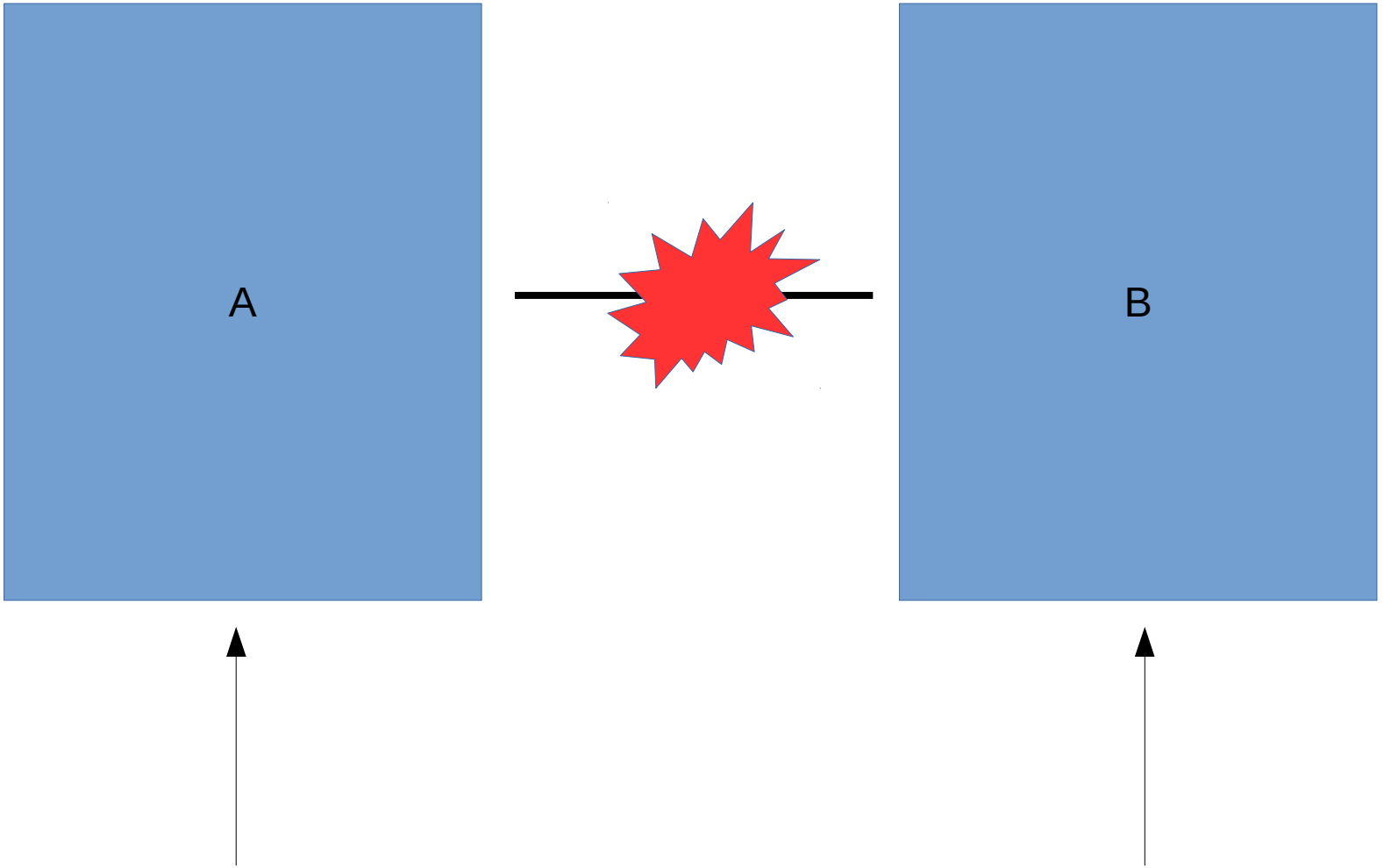
Consistency in CAP

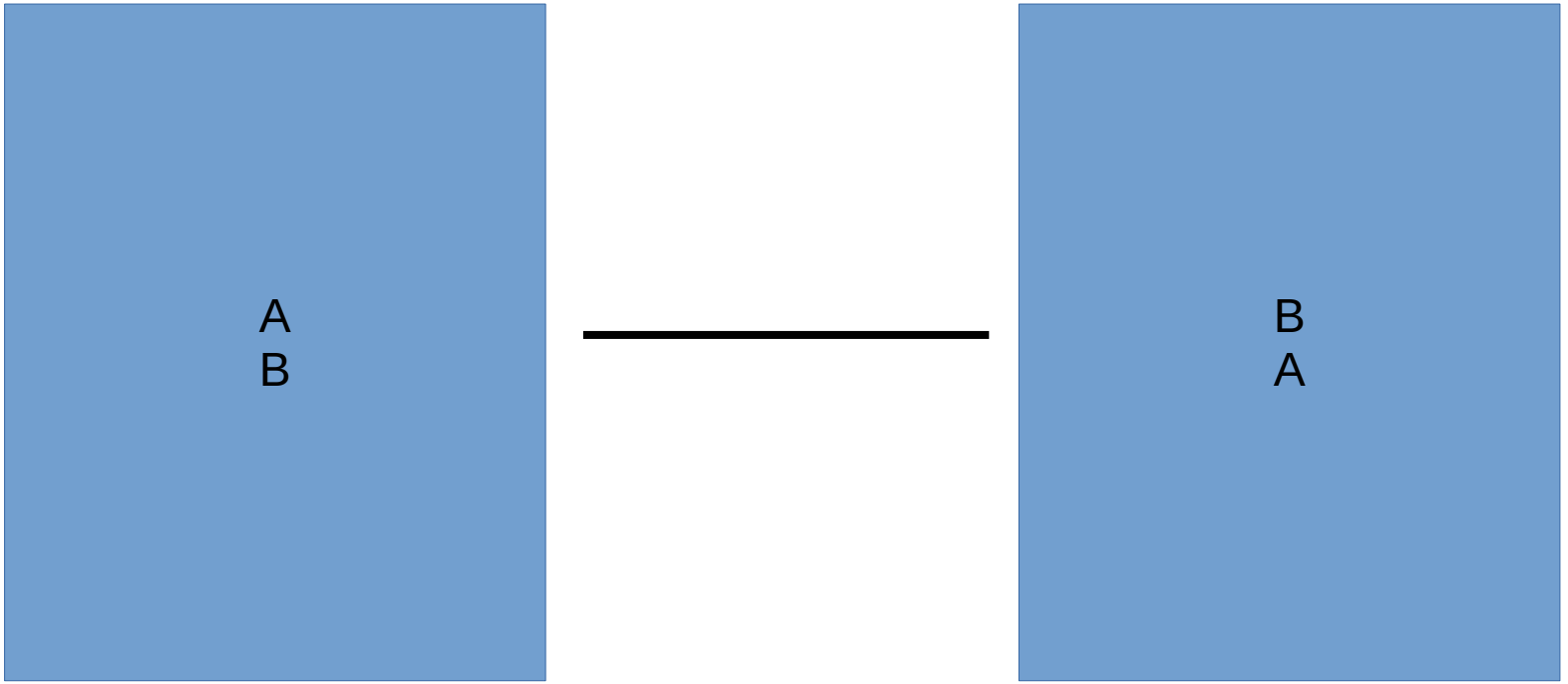
Every read receives the most recent write or an error

Consistency in consistency models

Set of rules that must be followed by transactions


Strong
Weak
Casual
Eventual
RYW


A
B
Write A
Write B


A
B
B
A

BASE

- Basically Available,
- Soft State,
- Eventually Consistent

```
if (Alice.balance > amount) do
  new_Alice = Alice.balance - amount
  new_Bob = Bob.balance + amount

  Alice.balance = new_Alice
  Bob.balance = new_Bob
end
```

```
if (Alice.balance > amount) do
  new_Alice = Alice.balance - amount
  new_Bob = Bob.balance + amount
-----------------------------------
  Alice.balance = new_Alice
  Bob.balance = new_Bob
end
```

amount: 10PLN, from: Alice, to: Bob

elixir

ELM

- Elixir is as a language that makes trading consistency for availability easy

- you can totally ignore all those fancy, but complex distributed stuff and there are many other benefits

PostgreSQL

Let it crash

```
:ok = File.write(„file.txt”, „contents”)
```

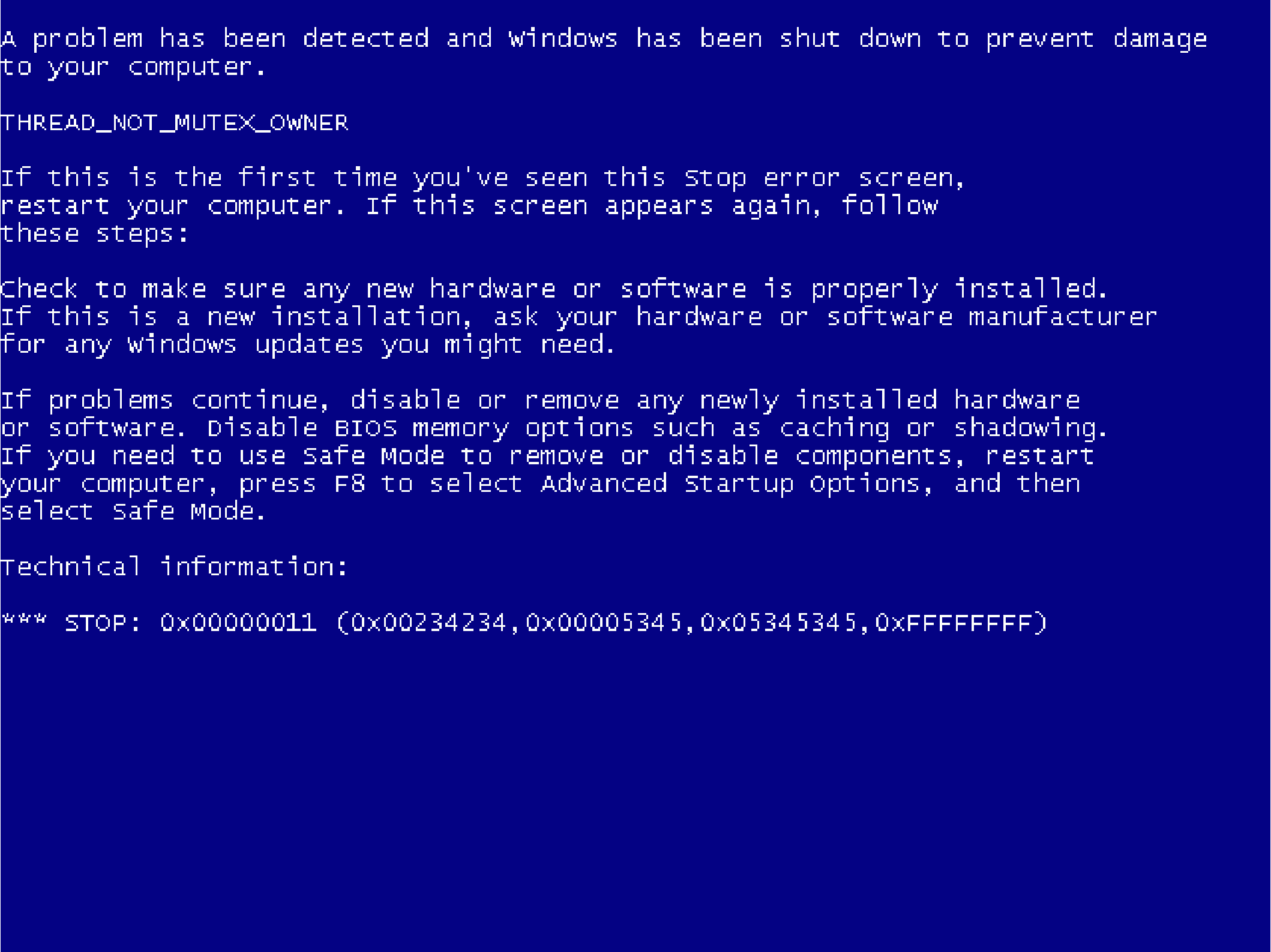
Let it crash

```
case File.write(„file.txt”, „contents) do
  :ok -> print_success_message
  {:error, :enoent} -> display_dialog
end
```

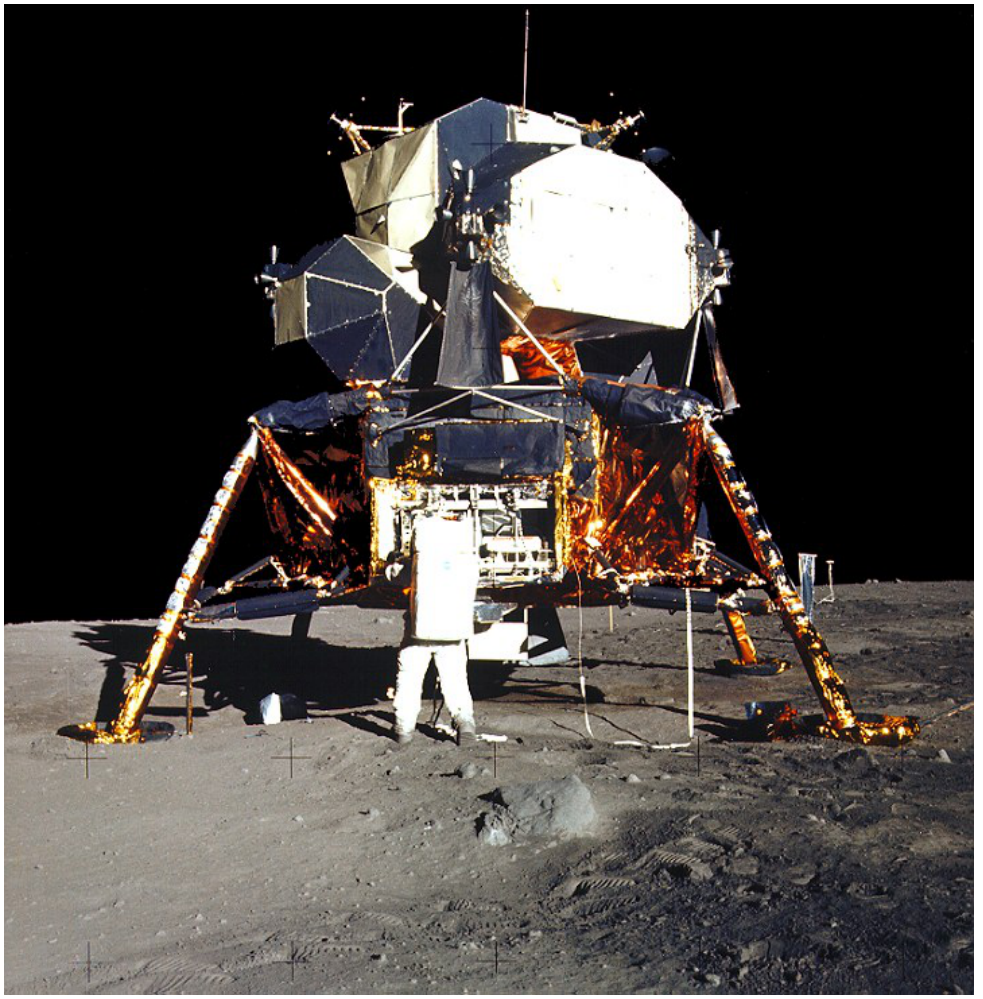
Dynos are also restarted (cycled) at least once per day to help maintain the health of applications running on Heroku.

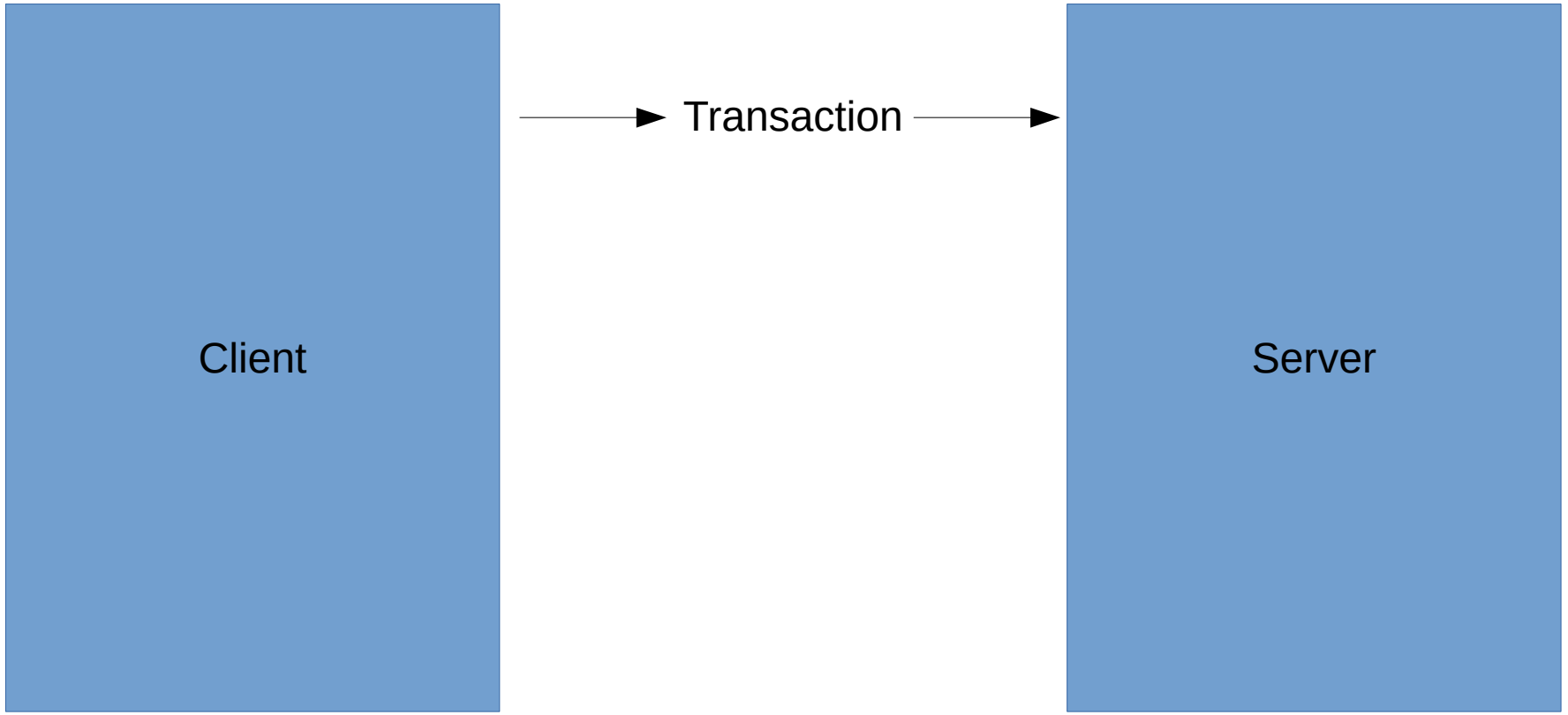
Any changes to the local filesystem will be deleted.

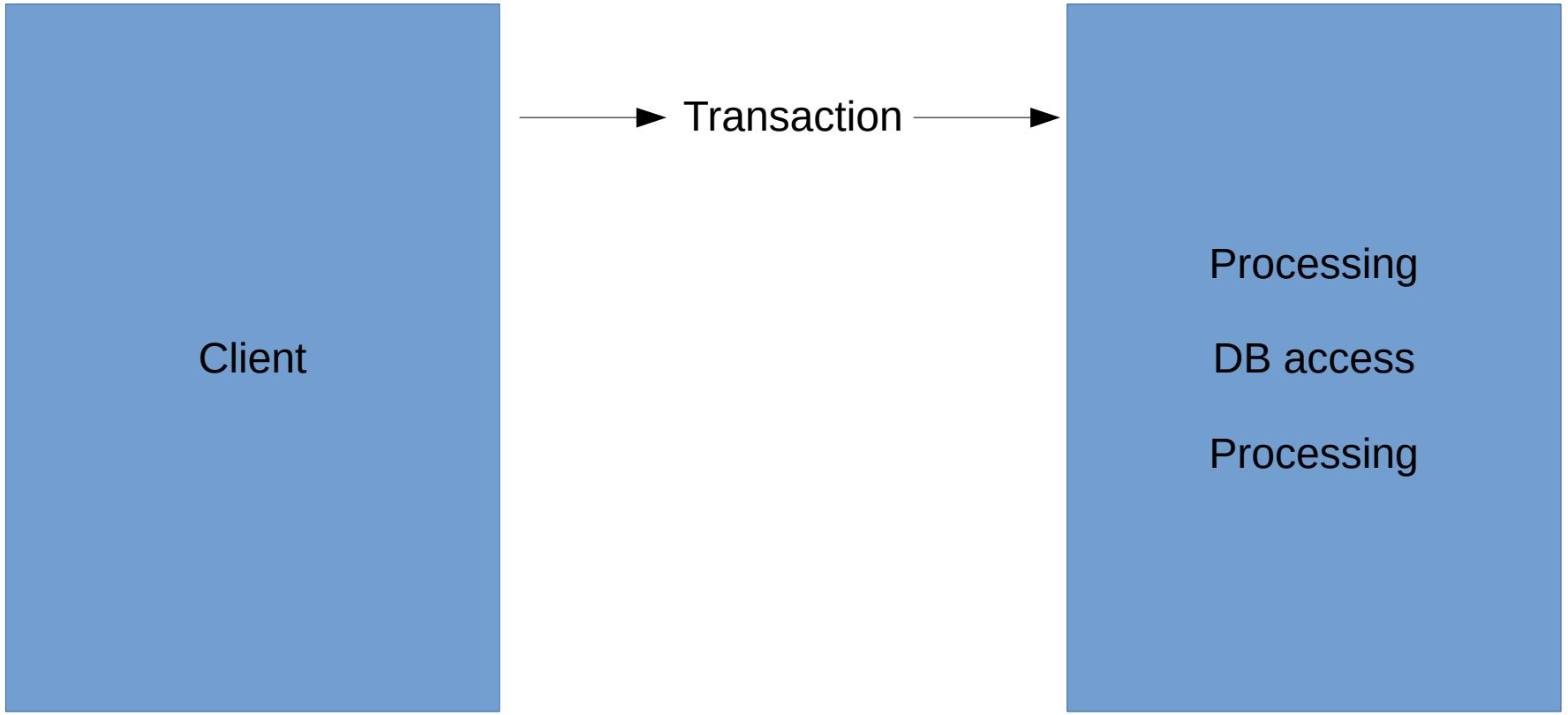
The cycling happens once every 24 hours (plus up to 216 random minutes, to prevent every dyno for an application from restarting at the same time).

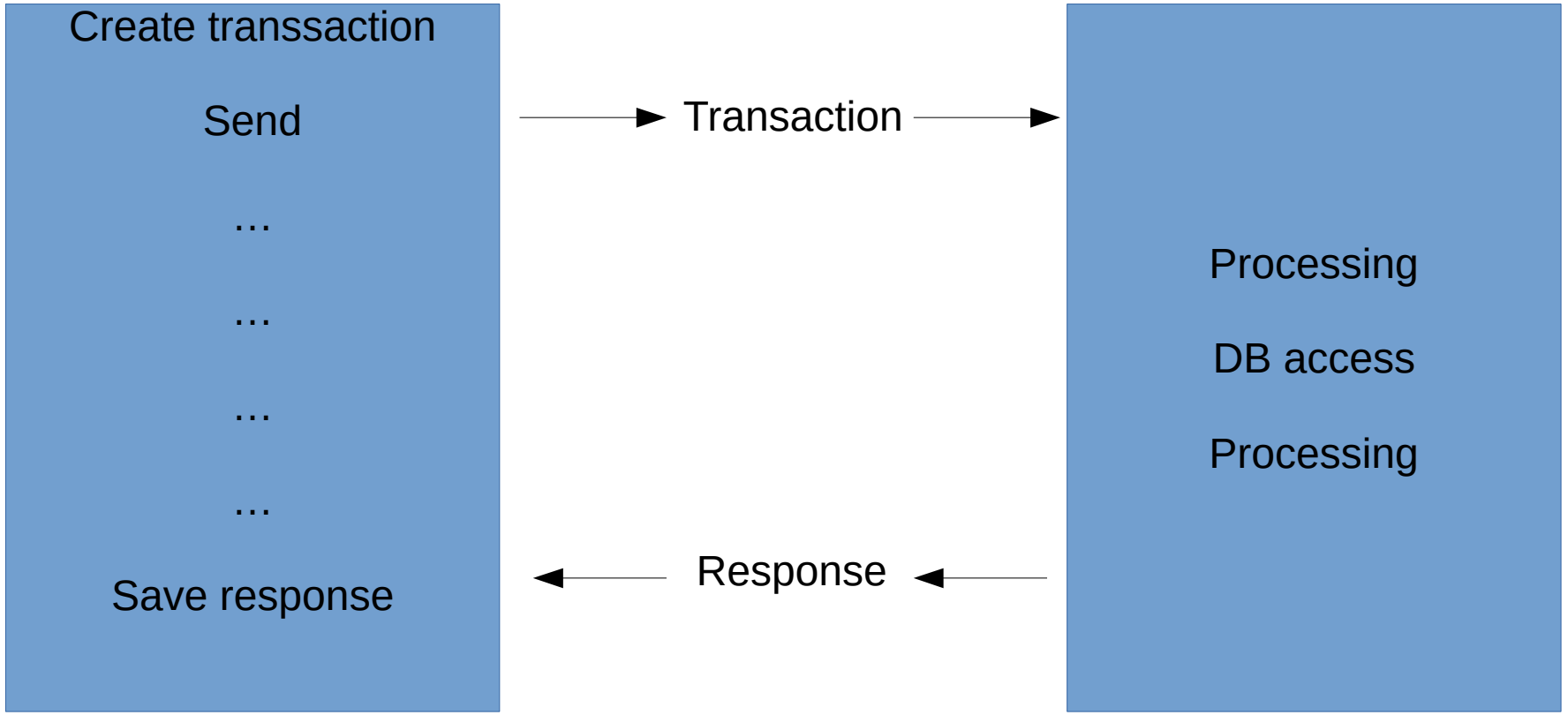

A problem has been detected and windows has been shut down to prevent damage
to your computer.
THREAD_NOT_MUTEX_OWNER
If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:
Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any windows updates you might need.
If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.
Technical information:
*** STOP: 0x00000011 (0x00234234,0x00005345,0x05345345,0xFFFFFFFF)

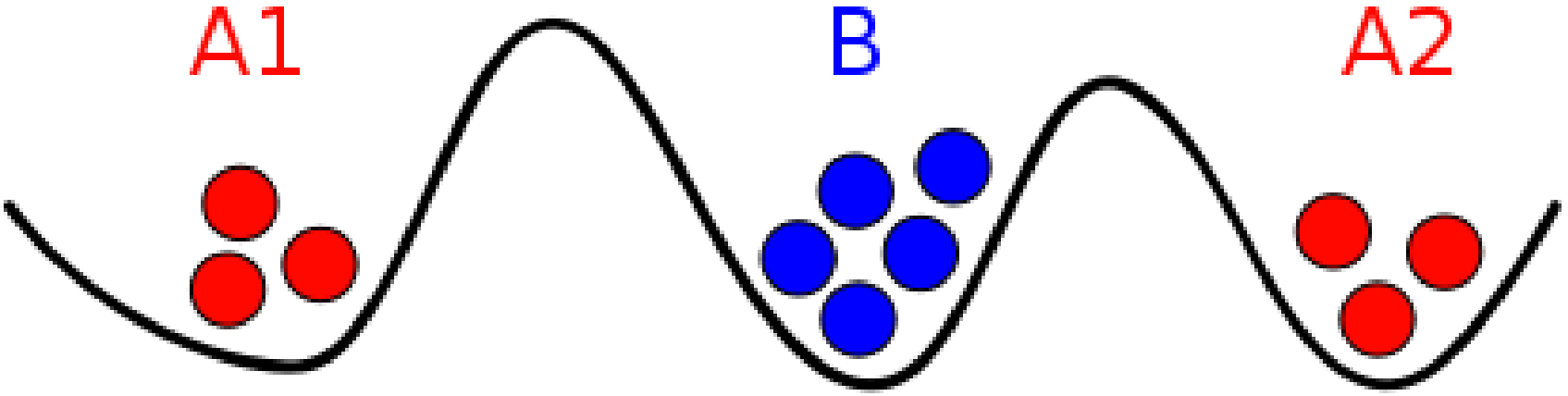
Primary Guidance, Navigation and Control System




Client
Transaction
Server


Client
Transaction
Processing
DB access
Processing


Create transsaction
Send
…
…
…
…
Save response
Transaction
Response
Processing
DB access
Processing


A1
B
A2

How to retry?

- Idempotent calls

How to retry?

- Idempotent calls
- Client side generated UUIDs

How to retry?

- Idempotent calls
- Client side generated UUIDs
- Using PUT

How people see Elixir?

- Great for distributed systems
- Great for complex stateful communication
- Perfect with BASE databases

How do we use Elixir?

- Single node service
- Client – Server without state
- ACID db (Postgres)

Side effects in check

Testing in imperative programming

- Setup state
- Run tested code
- Check new state

Testing in functional programming

- Setup a data structure
- Run tested code
- Check new data structure

Pipe operator

```
gather_data
|> make
|> series
|> of
|> transformations
|> return_data
```

Testing plug

```
test „renders empty index” do
  conn = build_conn()
  conn =
    get(conn, some_path(conn, :index)
  assert html_response(conn, 200)
end
```

Dealing with side effects

```
def save_if_something(data) ->
  if data.some_condition do
    Repo.insert(data.entity)
  end
  data
end
```

Dealing with side effects

```
def save_if_something({data, multi}) ->
  new_multi = if data.some_condition do
    Multi.insert(multi, :tag, data.entity)
  else
    multi
  end
  {data, new_multi}
end
```

Testing Multi

```
test „performs insert when condition is met” do
  data = %{some_condition: true, entity: …}




end
```

Testing Multi

```
test „performs insert when condition is met” do
  data = %{some_condition: true, entity: …}
  {data, multi} =
    save_if_something({data, Multi.new})



end
```

Testing Multi

```
test „performs insert when condition is met” do
  data = %{some_condition: true, entity: …}
  {data, multi} =
    save_if_something({data, Multi.new})
  assert data == data
  assert [{:tag, :insert, …}] ==
    Multi.to_list(multi)
end
```

Dealing with side effects

```
def read_from_db(data) ->
  case Repo.get(…) do
    {:ok, entity} →
      %{data | entity: entity}
    _ -> data
  end
end
```

Dealing with side effects

```
def read_from_db(data, repo \\ Repo) ->
  case repo.get(…) do
    {:ok, entity} →
      %{data | entity: entity}
    _ -> data
  end
end
```

Testing explicit contract

```
defmodule FakeRepo do
  def get(_), do: {:ok, …}
end

test „performs insert when condition is met” do
  data = read_from_db(data, FakeRepo)
  assert data.entity == ...
end
```

Other benefits

- No callbacks
- Great performance
- Umbrella applications

Tradeoffs

- Rapid evolution can be a disadvantage
- Compilation time
- Type system is optional
- Libraries (?)

Summary

- success story
- don’t assume financial systems are ACID
- Elixir solves problems Ruby can’t solve, but it solves problems that Ruby can solve

Thank you

Tomasz Kowal
@snajper47