

Operational hazards of managing PostgreSQL DBs over 100TB

Why?

Storage Limits

AWS RDS	64TB
Google CloudSQL	64TB
Azure Database Flexible Server	64TB

Storage Limits

AWS Aurora	256TB
Google AlloyDB	128TB

Teresa Lopes

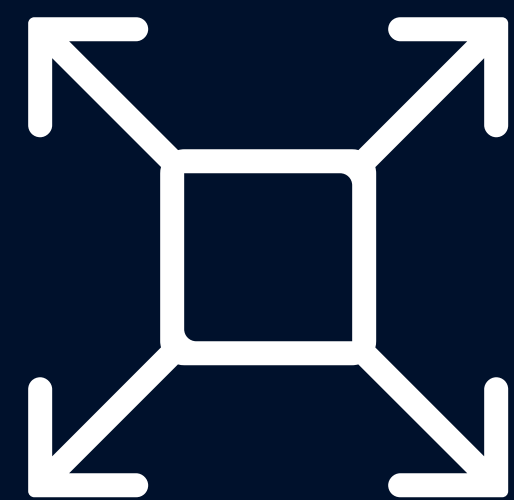
Database Engineer

Adyen



Agenda





Big, but how big?



**Bring balance
to the Database**



**Vacuum, it is
always Vacuum**



**To backup or
not To backup**



**Data, Data
everywhere**



Big, but how big?



Small DB + Low TPS



Large DB + Low TPS



Small DB + High TPS



Small DB + Strict Latency



Source: <https://www.redbull.com/>

Perfect Storm



Perfect Storm

Massive DB

Perfect Storm

Massive DB
+ Very High TPS

Perfect Storm

Massive DB
+ Very High TPS
+ Strict Latency

Perfect Storm

Massive DB

+ Very High TPS

+ Strict Latency

+ Hot Data

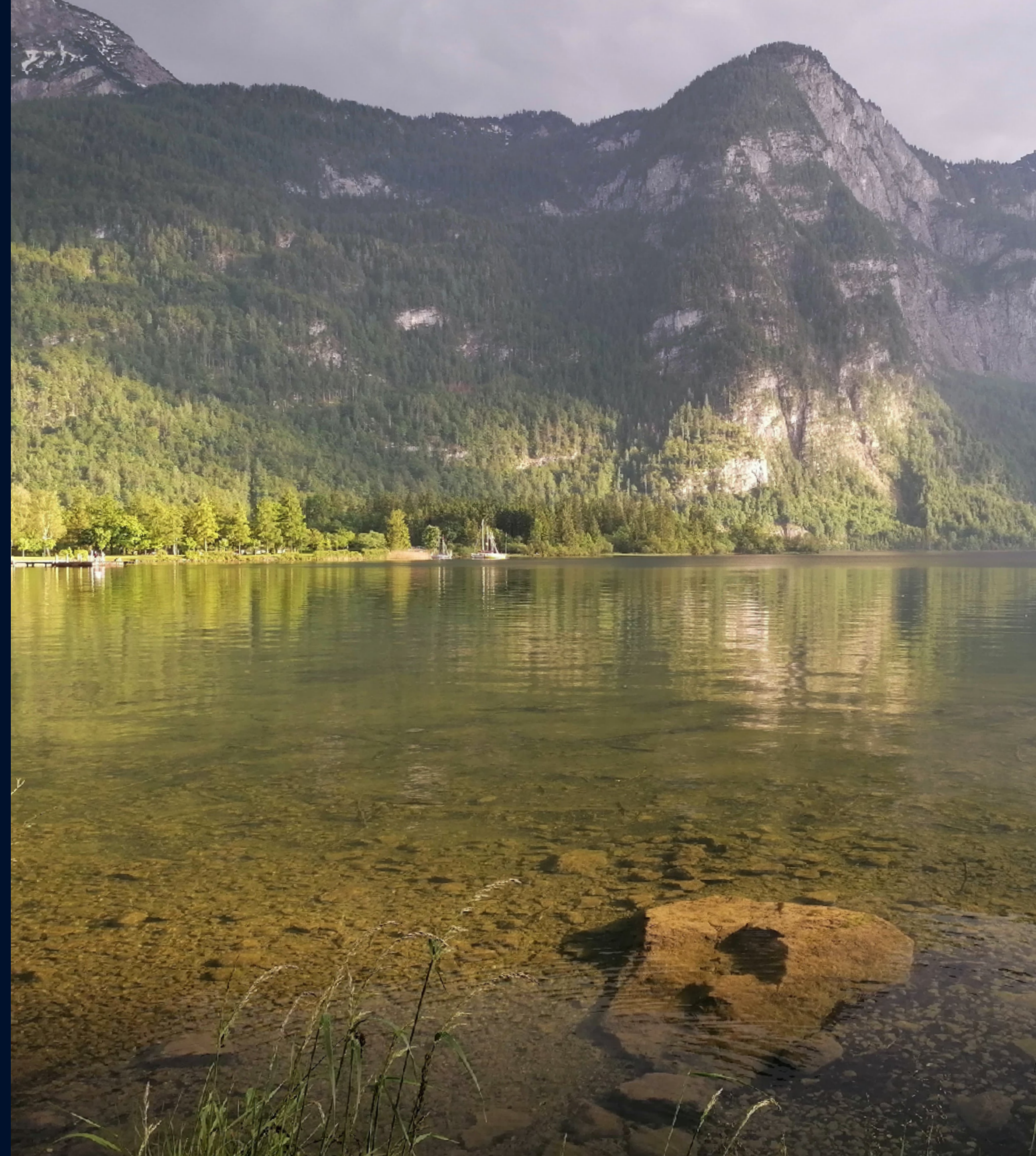
Perfect Storm



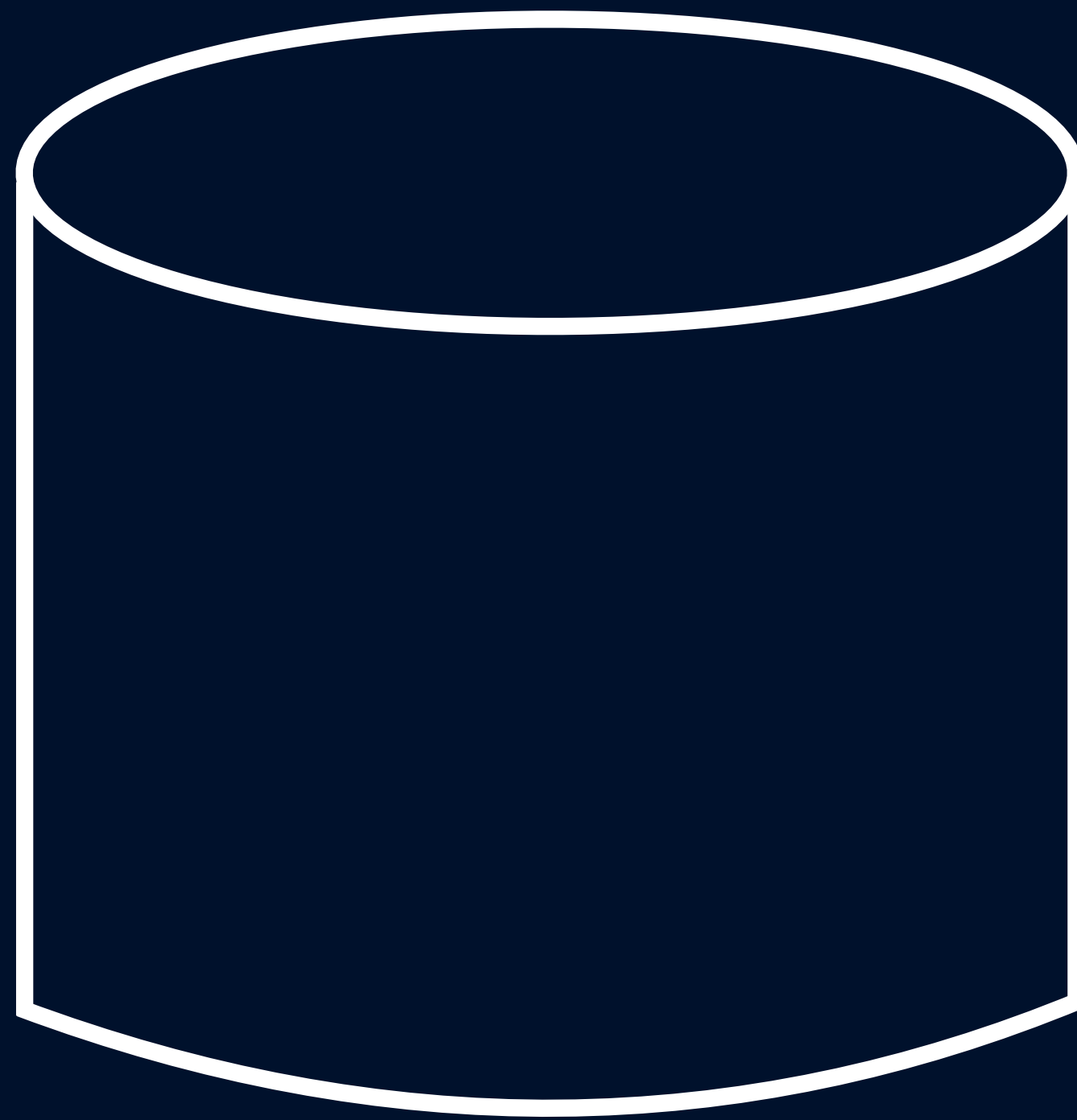
Source: <https://wien.orf.at>

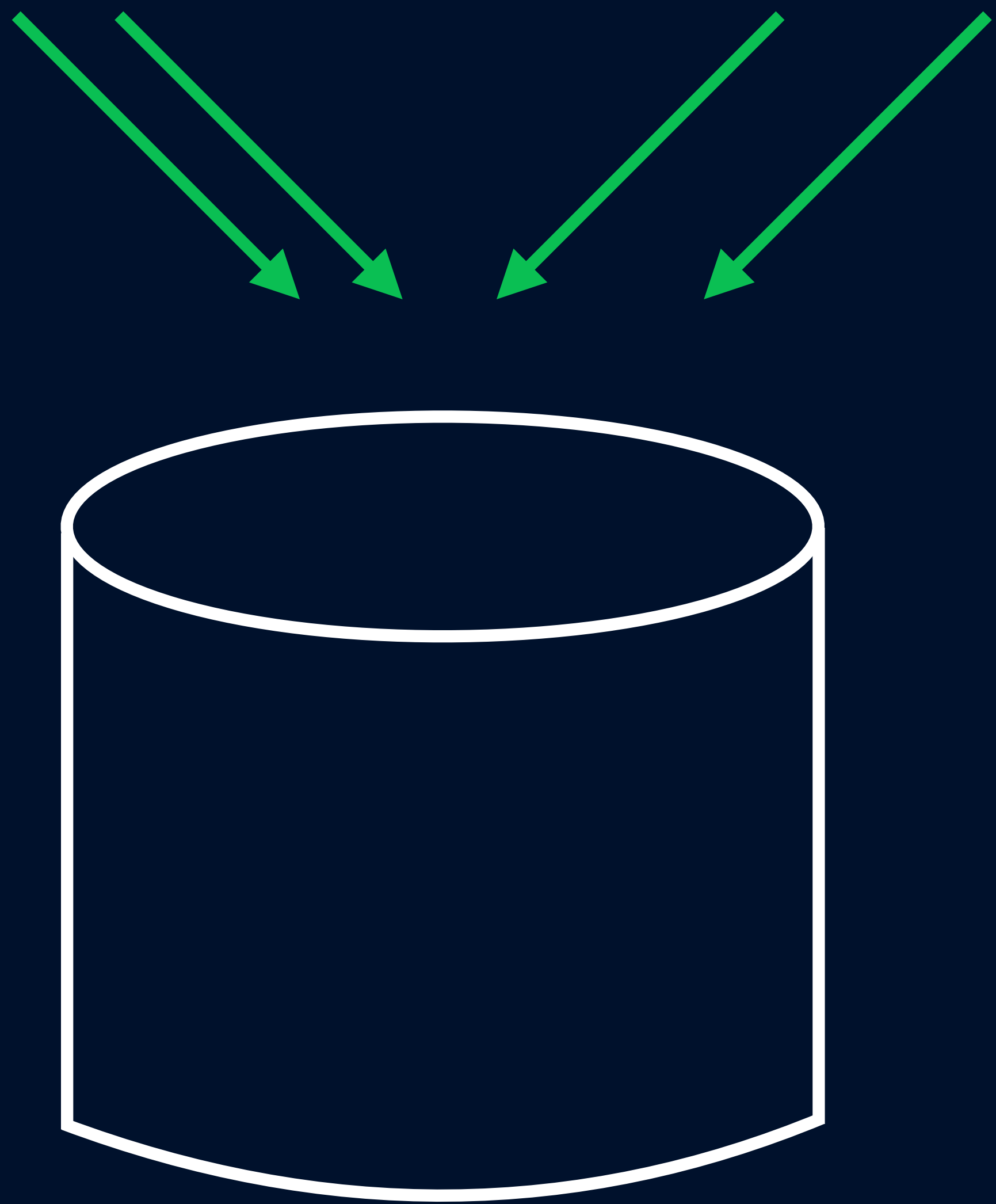


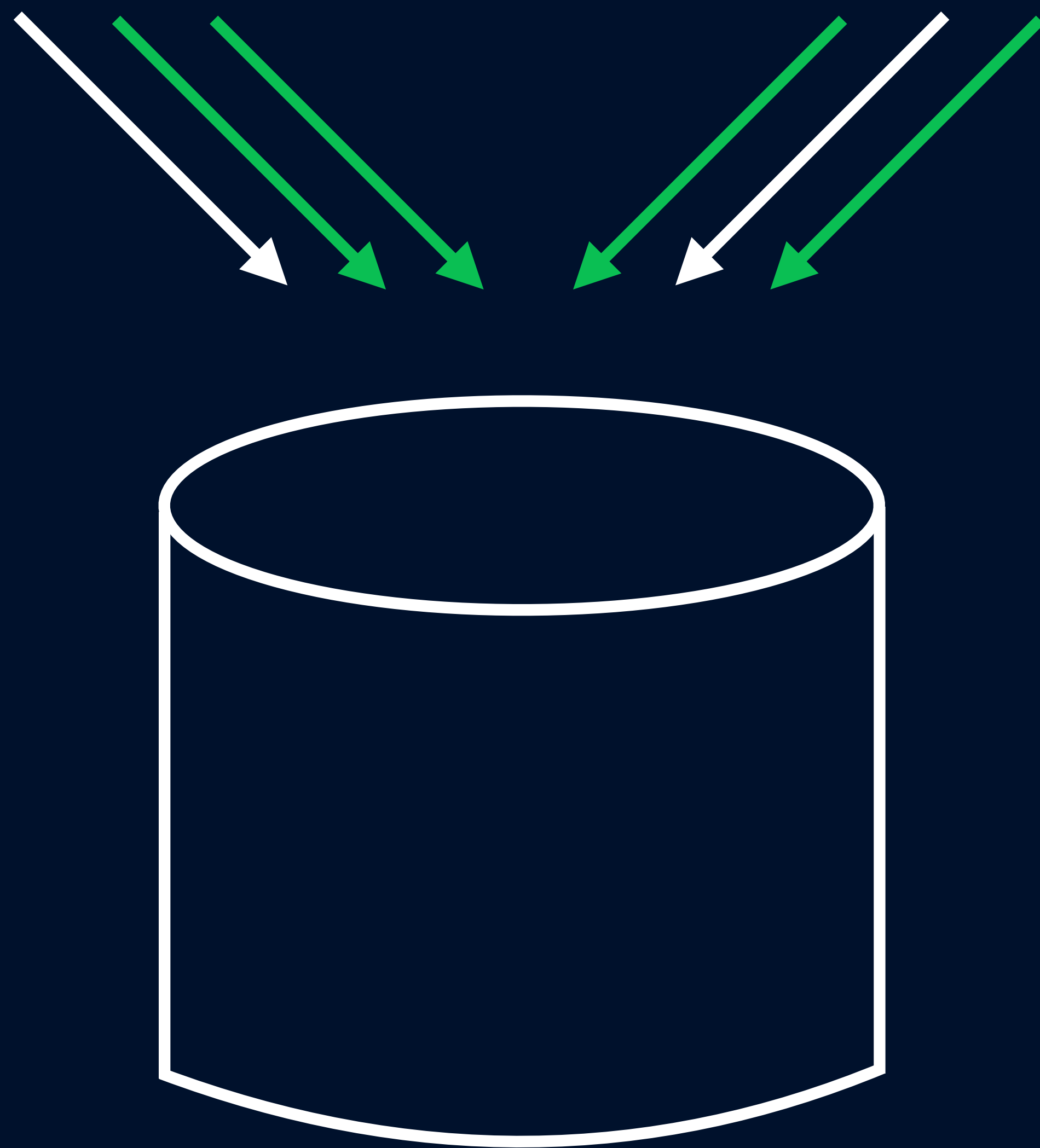
Bring balance to the Database



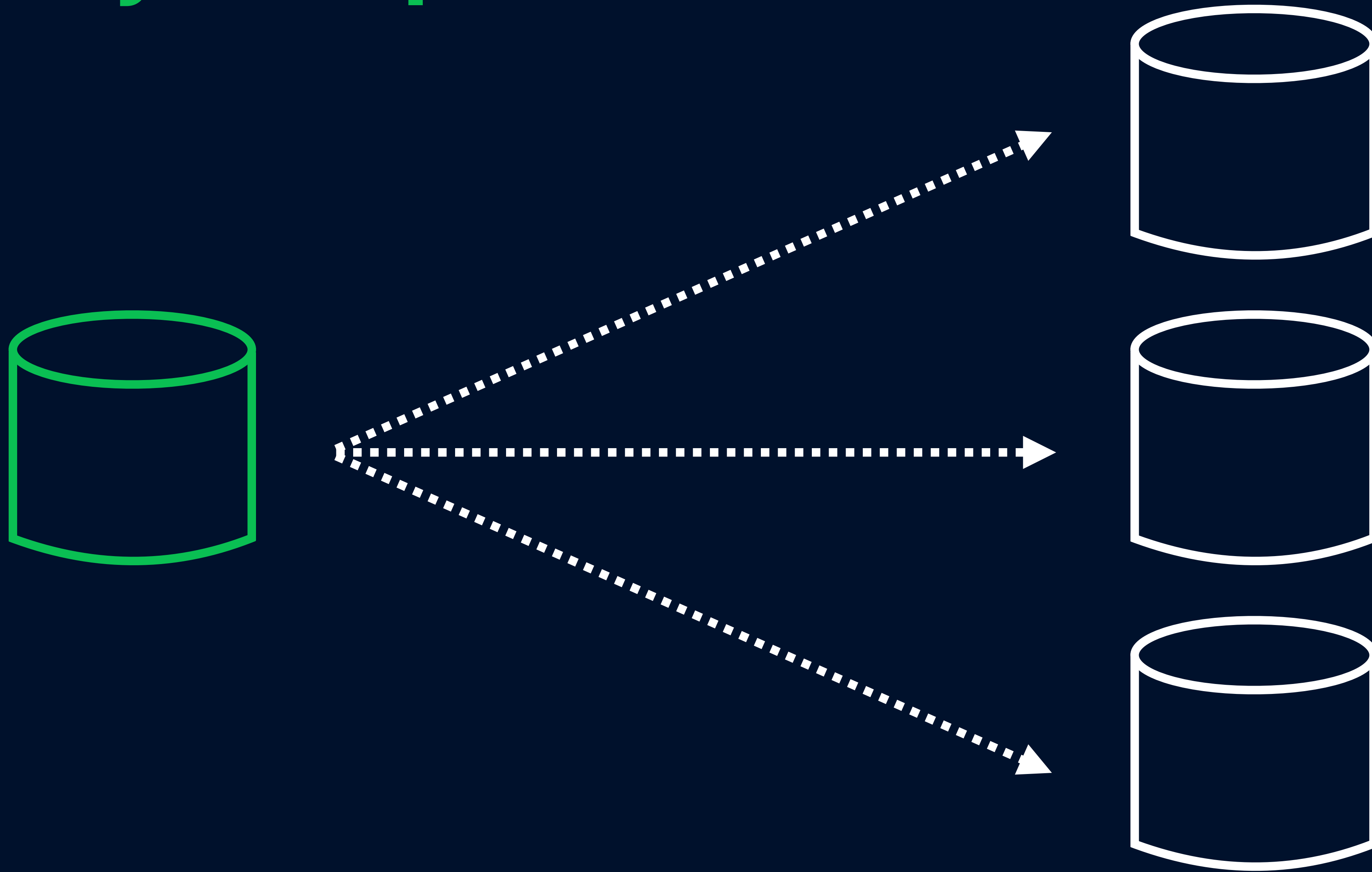
Single Instance



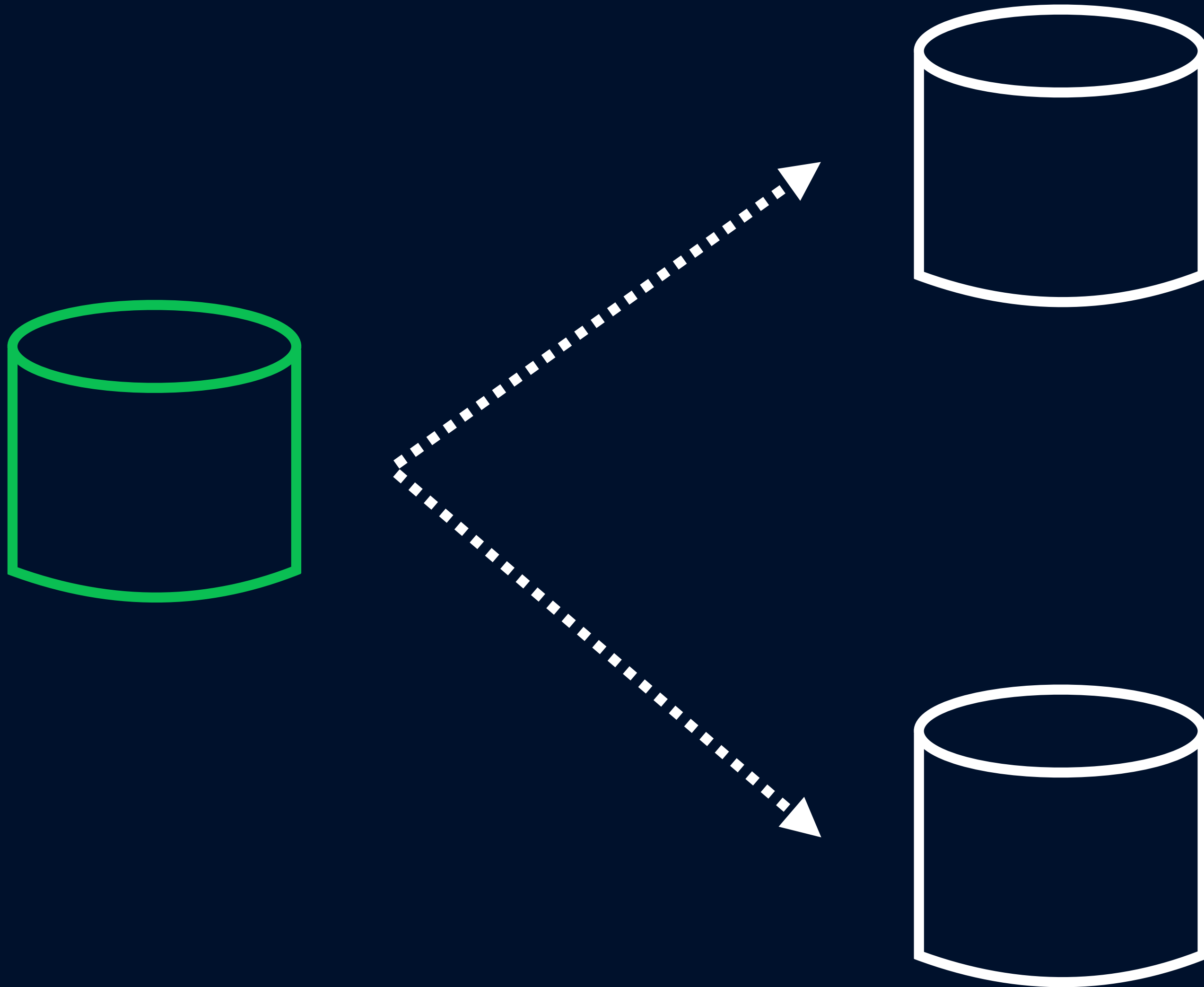




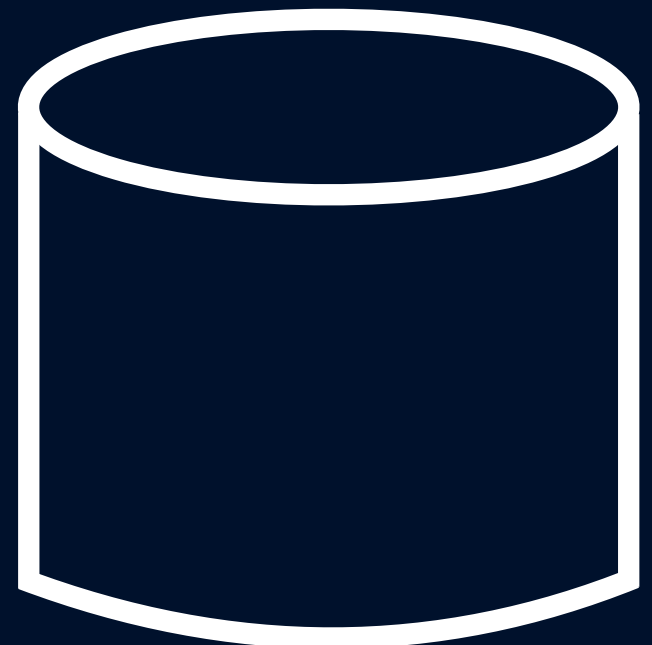
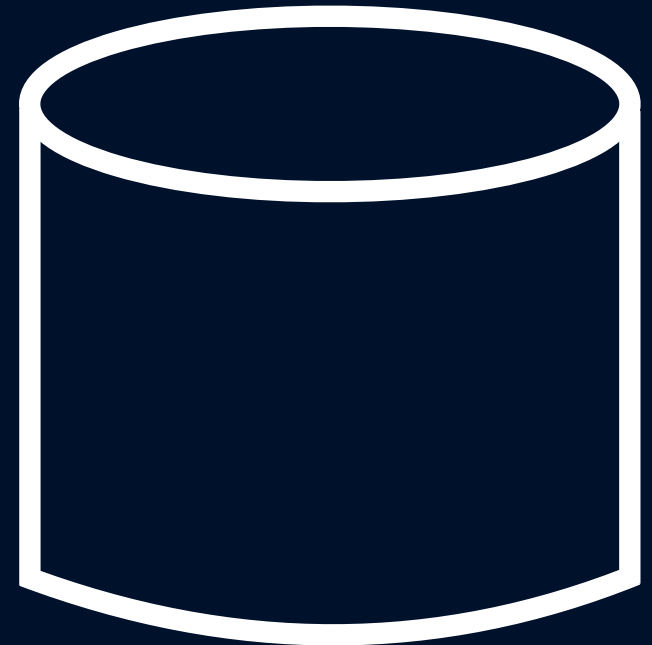
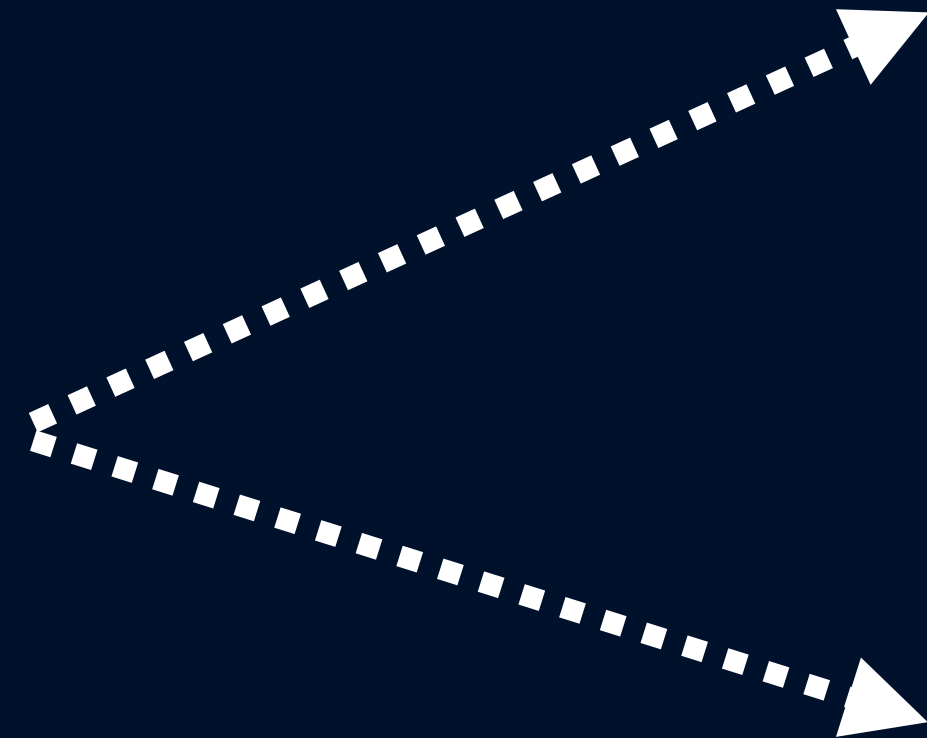
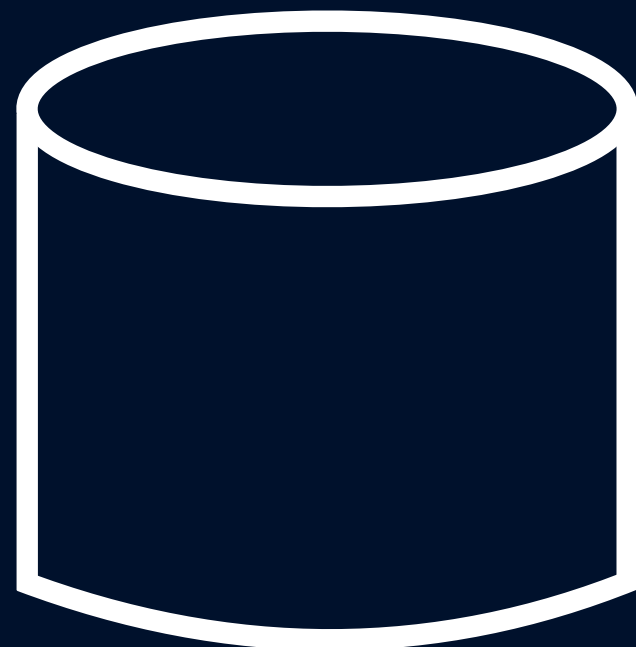
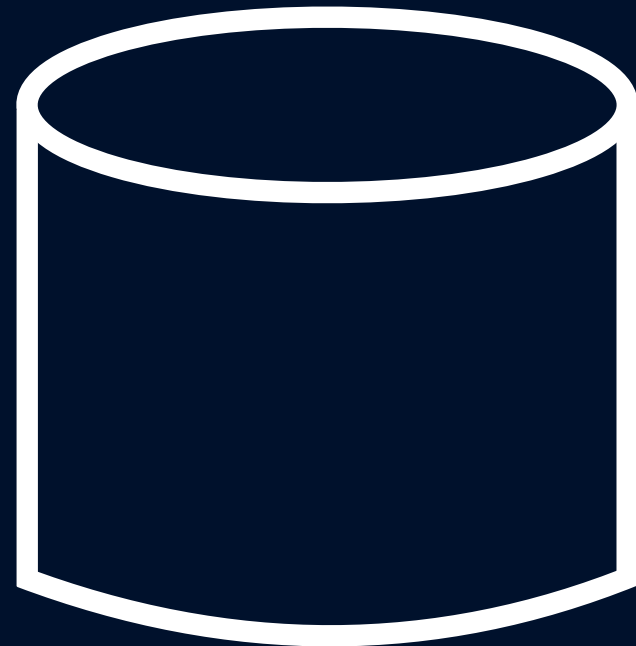
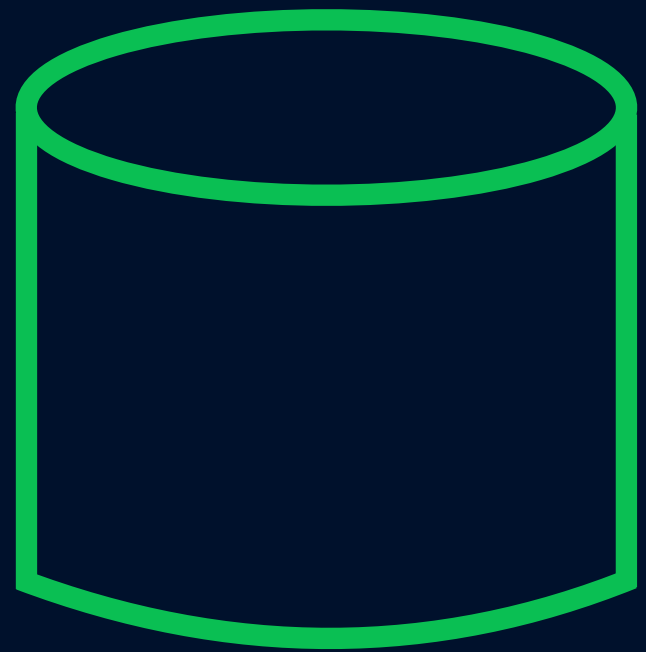
Primary + Replicas



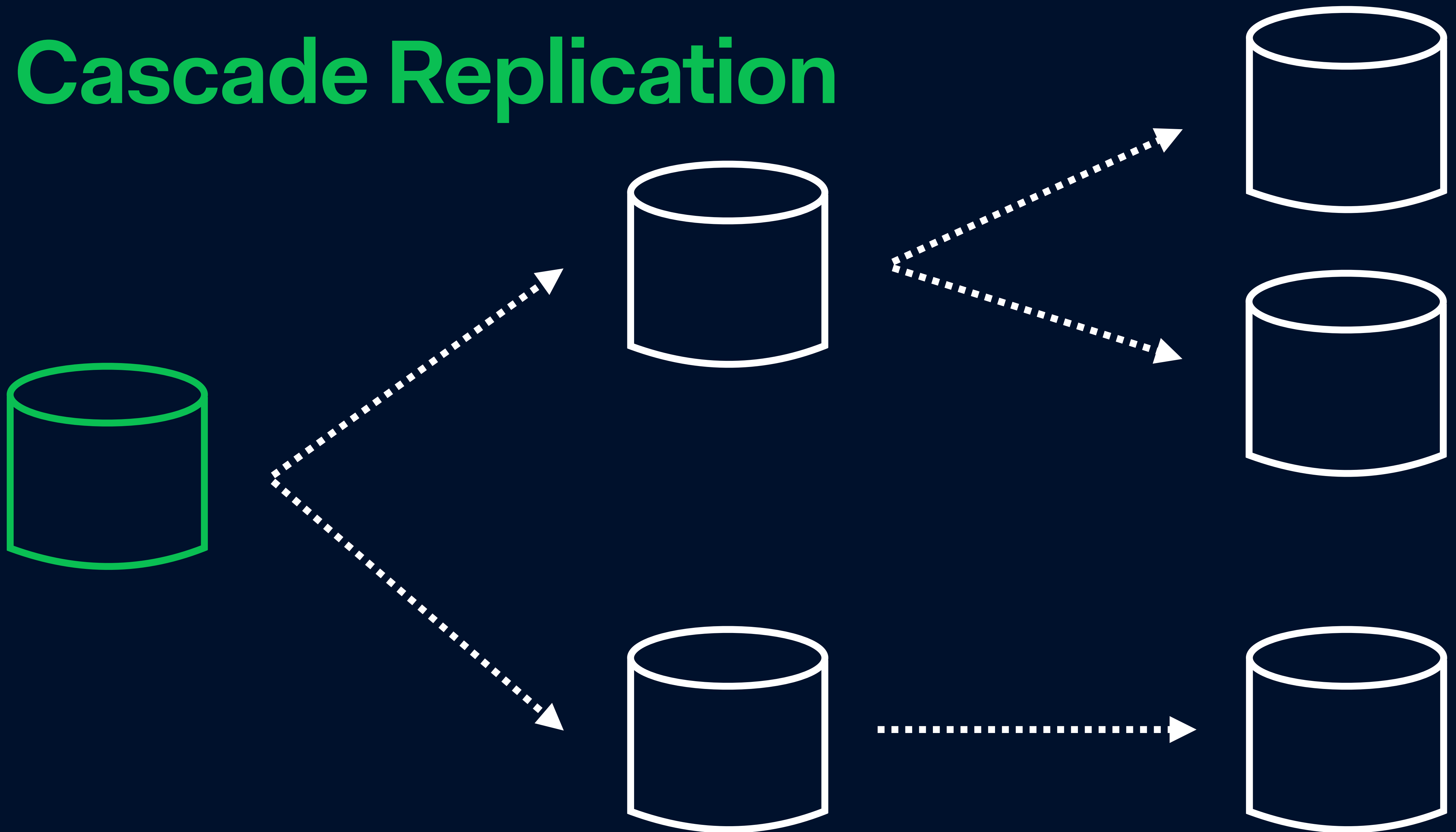
Cascade Replication



Cascade Replication



Cascade Replication



Replication



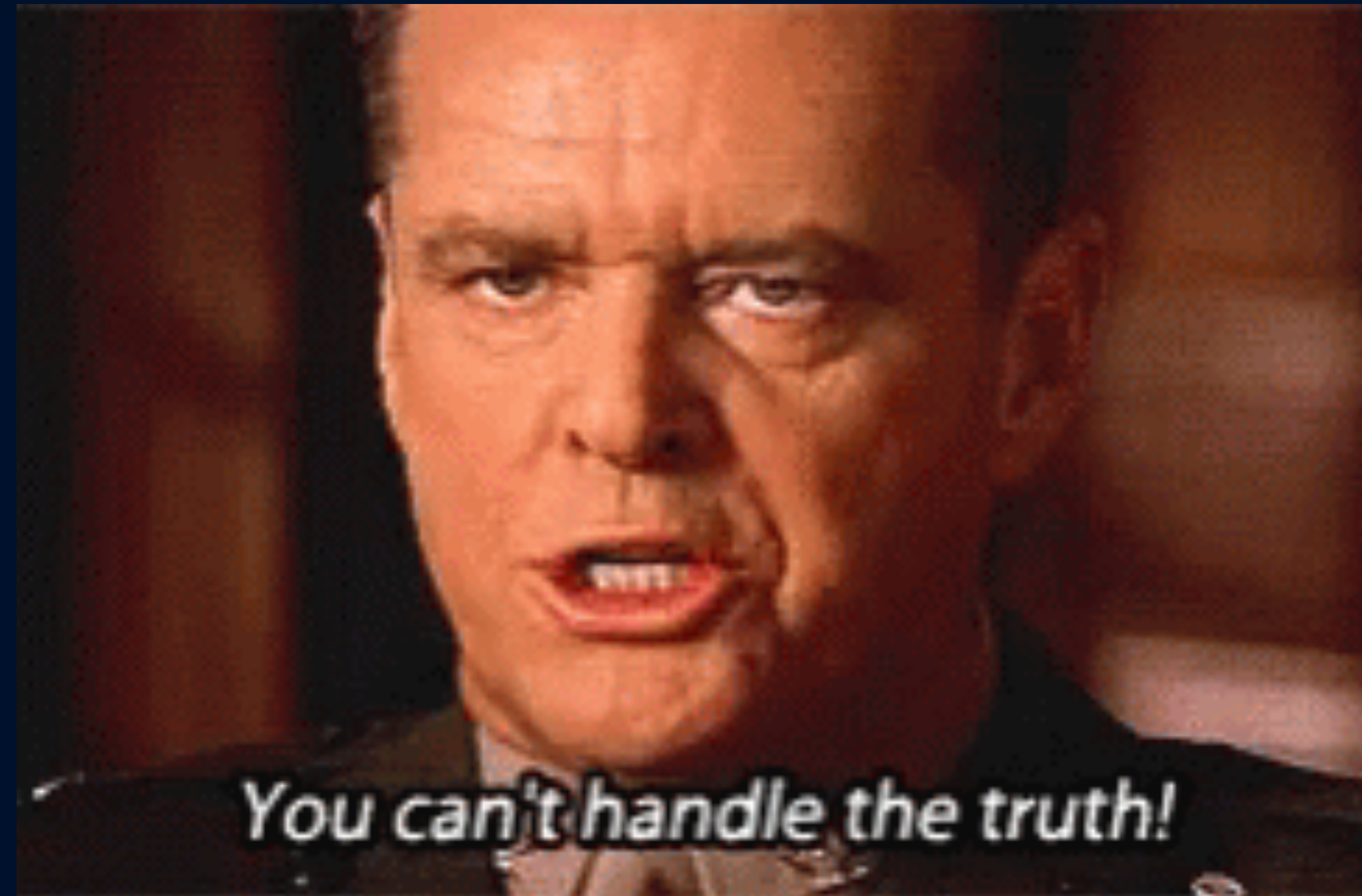


**Vacuum, it is
always Vacuum...**



ONE DOES NOT SIMPLY

AVOID VACUUM



You can't handle the truth!

Why?

**I DON'T
ALWAYS VACUUM,**

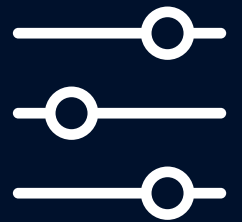
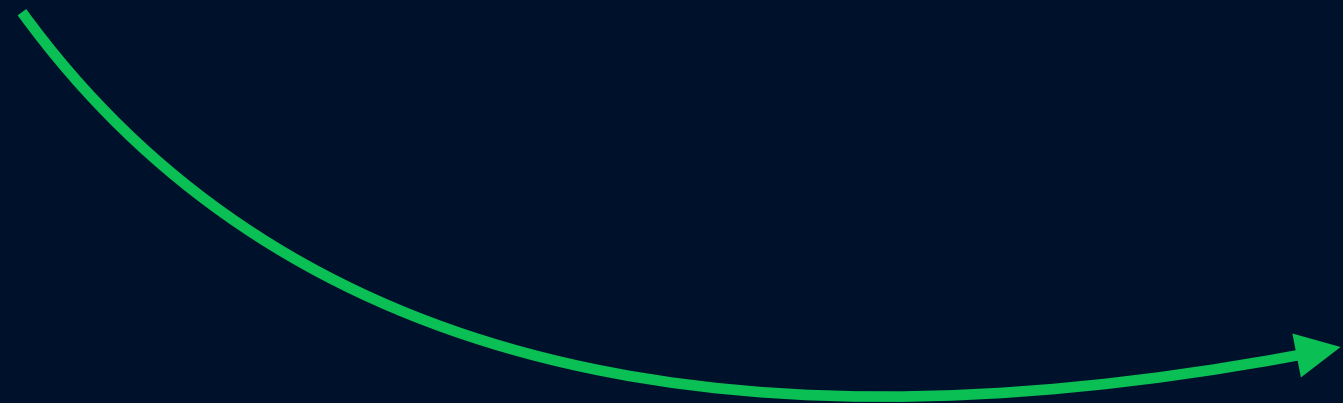


**BUT WHEN I
DO, IT TAKES 3 DAYS**

How?

Autovacuum 101

Autovacuum 101



`autovacuum_vacuum_threshold`

`autovacuum_vacuum_scale_factor`

`autovacuum_work_mem` (or `maintenance_work_mem`)

`autovacuum_analyze_threshold`

`autovacuum_analyze_scale_factor`

`autovacuum_max_workers`

`log_autovacuum_min_duration`

Autovacuum

Difficulty Level: HARD

Autovacuum

Difficulty Level: HARD



cost

Autovacuum

Difficulty Level: HARD



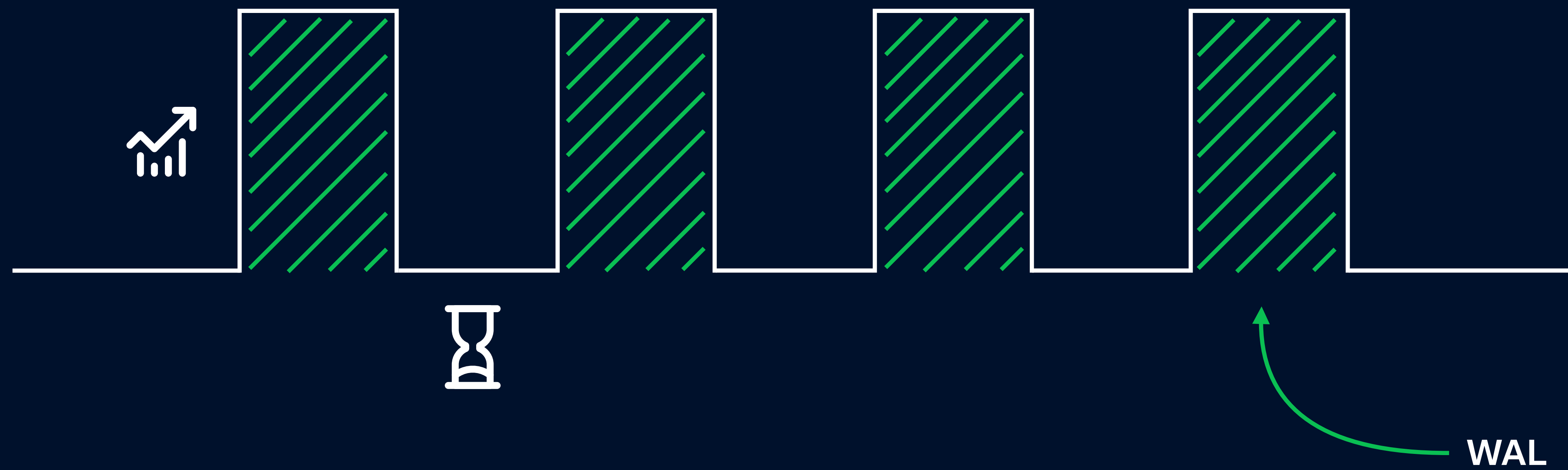
cost

vs



delay

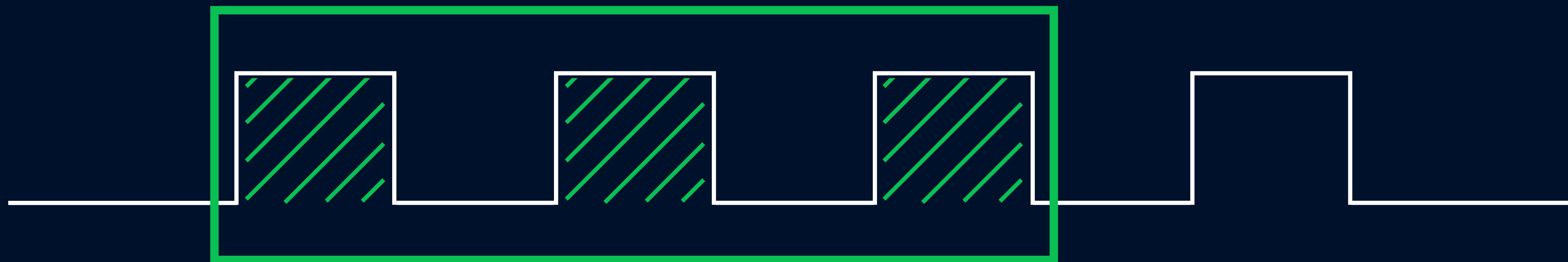


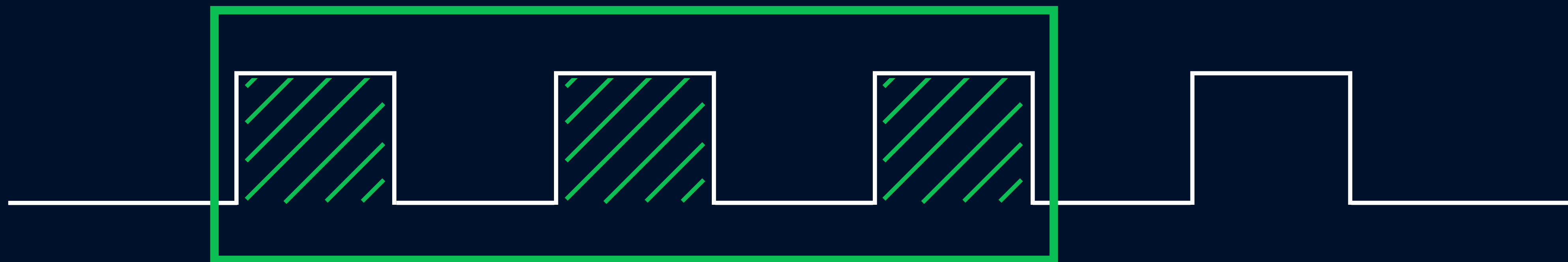


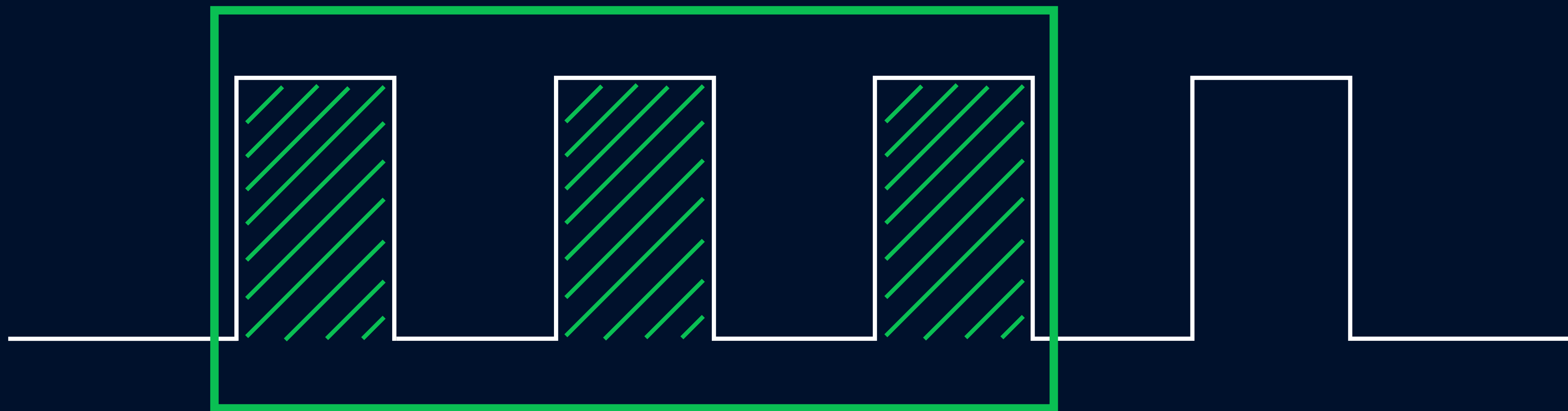
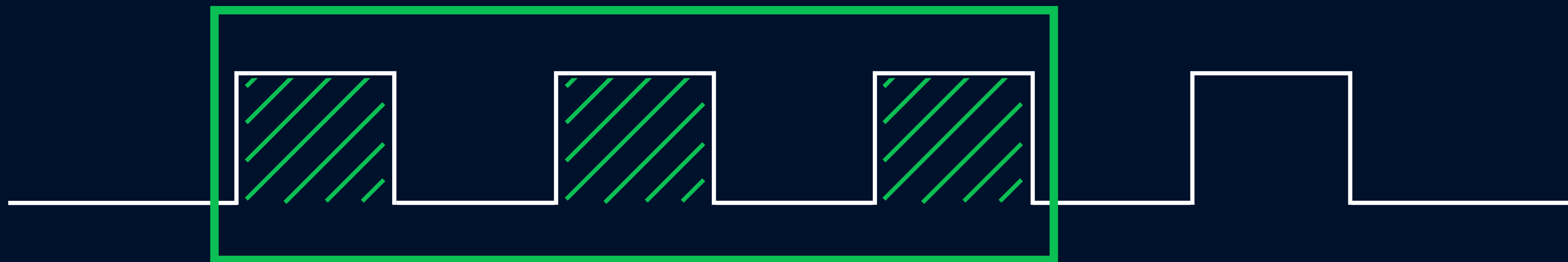








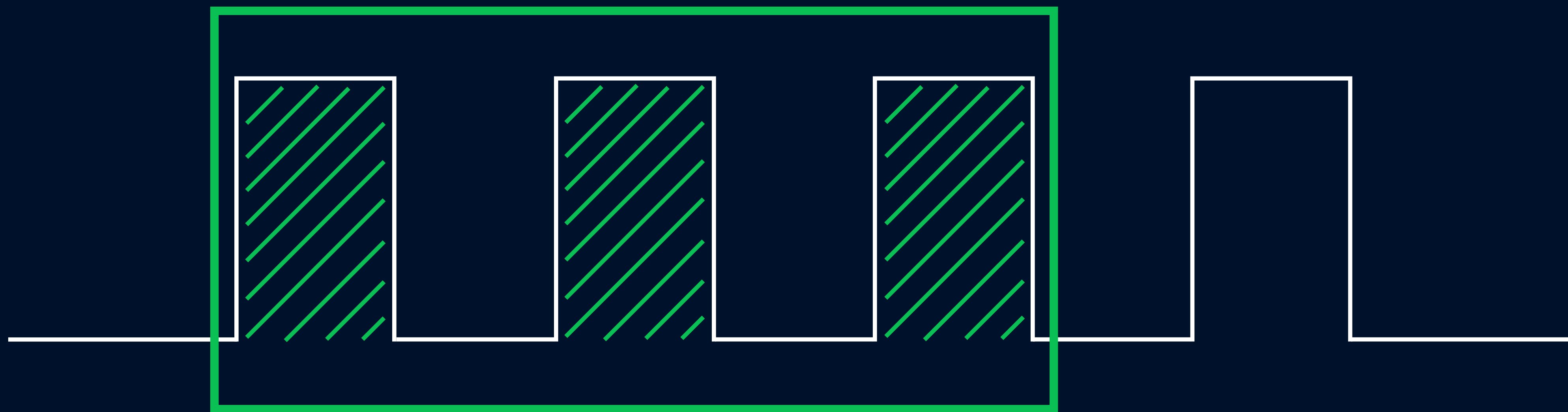


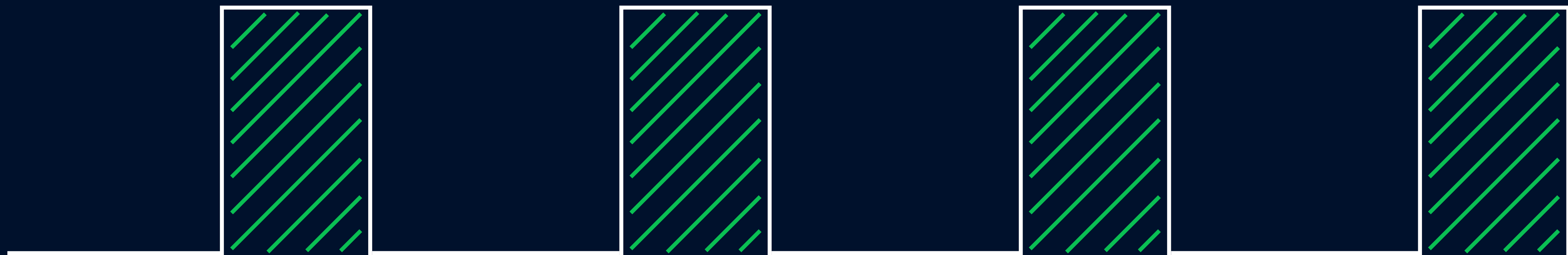
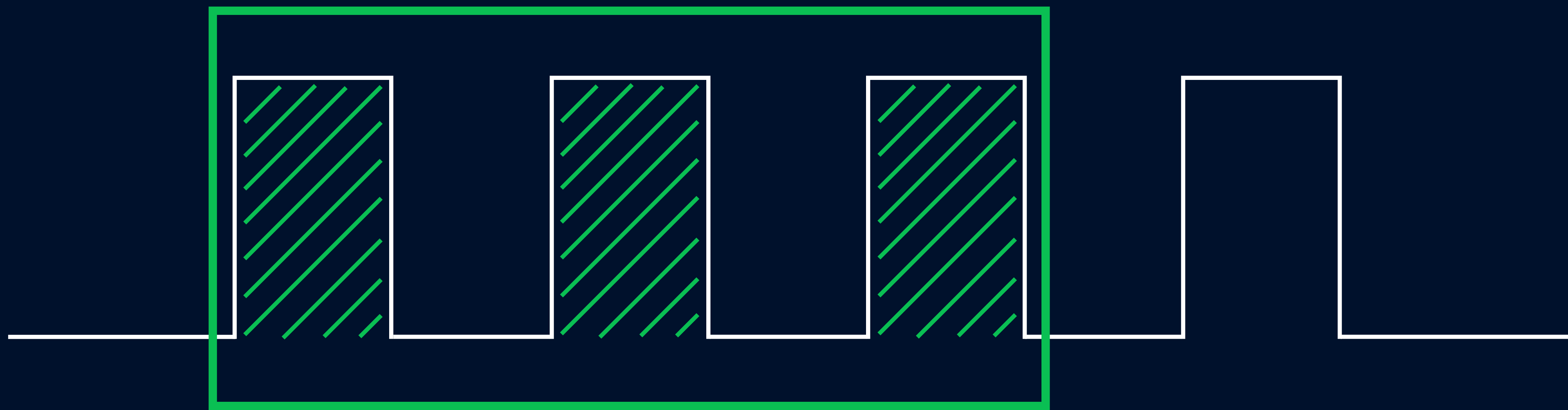


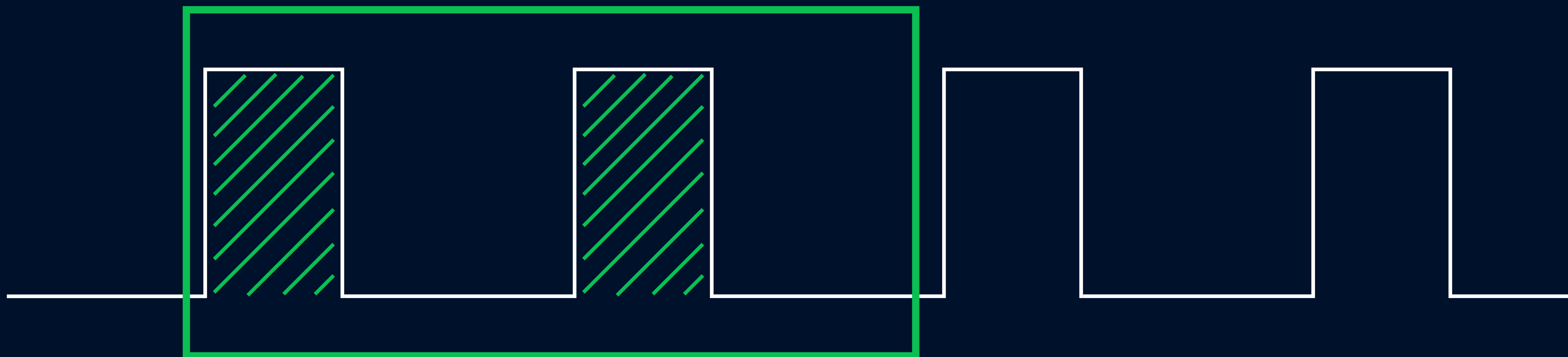
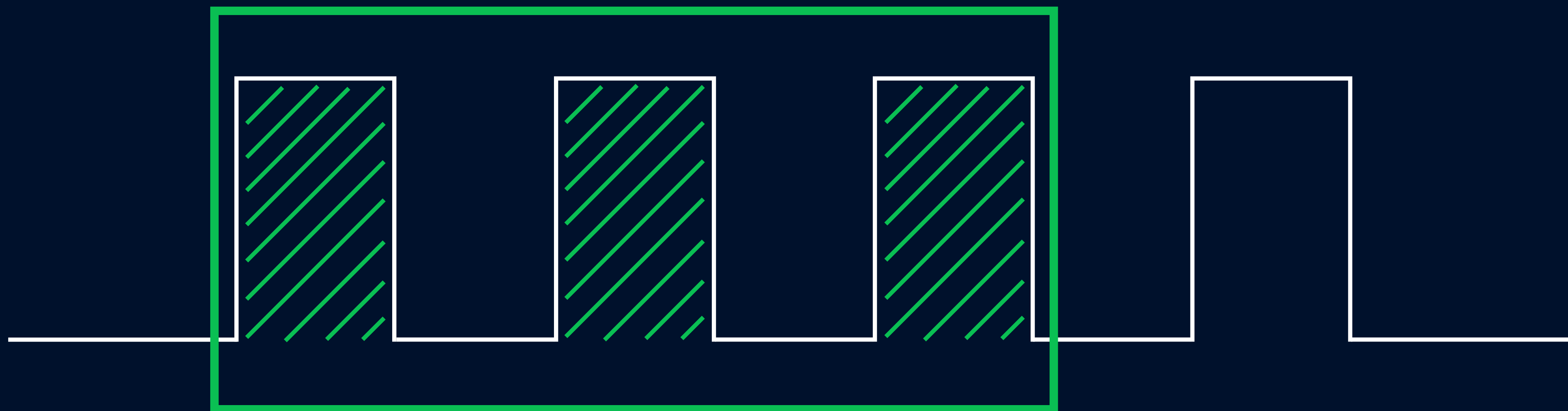












Manual Vacuum

Ruthless!

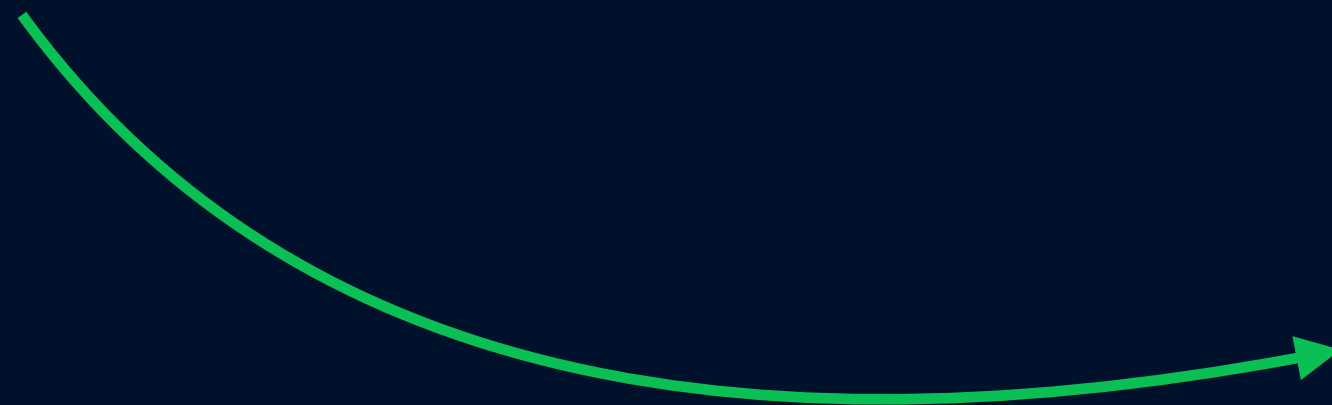
Manual Vacuum



Manual Vacuum



- Top bloated tables



Manual Vacuum



- Top bloated tables
- **Freeze tables**

Manual Vacuum



- Top bloated tables
- Freeze tables
- **Wraparound**

Manual Vacuum



- Top bloated tables
- Freeze tables
- Wraparound
- **Top N big tables**

Manual Vacuum

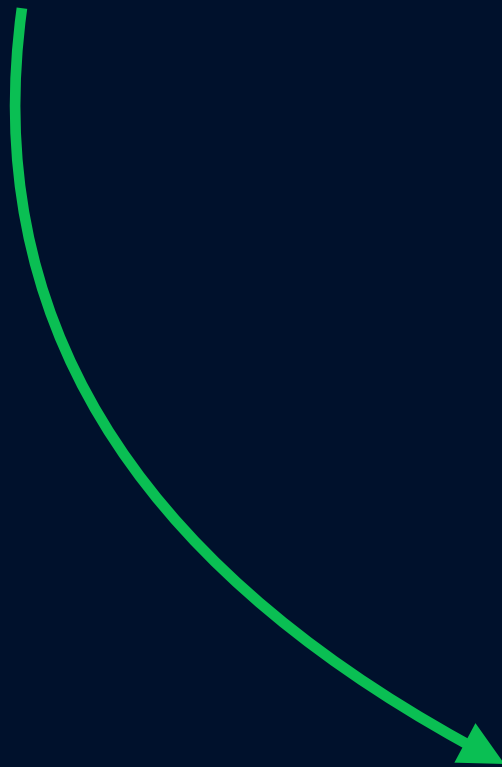


- Top bloated tables
- Freeze tables
- Wraparound
- Top N big tables
- **index_cleanup off** ⚡

“So, unless you are sure **you know** what you are doing (that is, you are in a wraparound-point emergency), please **pretend** this option `[index_cleanup off]` **doesn't exist**. Really.”

Christophe Pettus

PostgreSQL v17



“New memory management system for VACUUM...

- **Allow vacuum to more efficiently remove and freeze tuples**

(Melanie Plageman, Heikki Linnakangas)

- **Allow vacuum to more efficiently store tuple references**

(Masahiko Sawada, John Naylor)”

Manual Vacuum



Manual Vacuum

Do not forget the Analyze!





**To backup or not
To backup**





RUNNING PG_BASEBACKUP ON YOUR +100TB DATABASE

imgflip.com

Some “Math”:

Some “Math”:

Backup

=

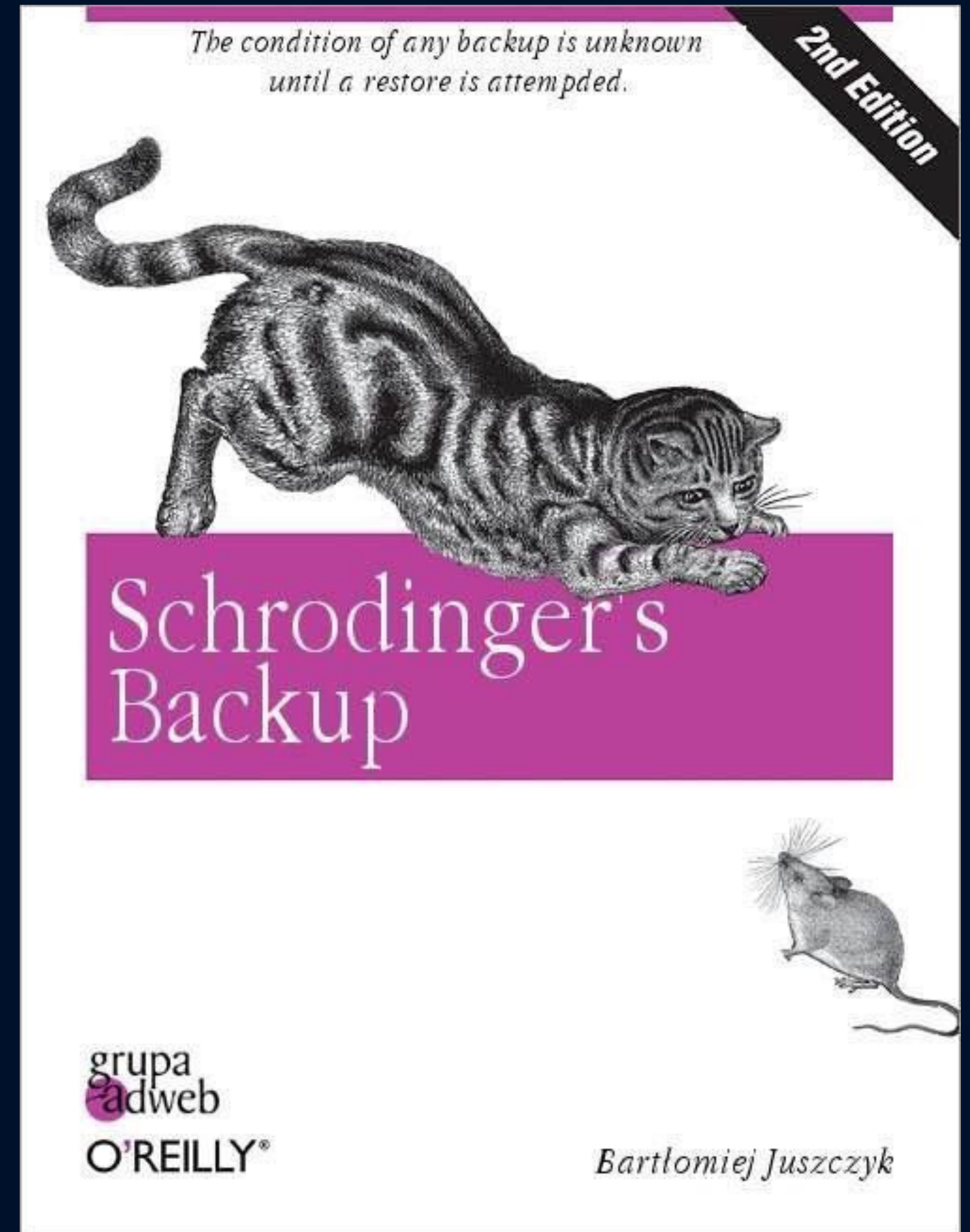
Backup + Restore

Some “Math”:

Backup

=

Backup + Restore



Source: <https://orlybooks.com/>

How we do it then?



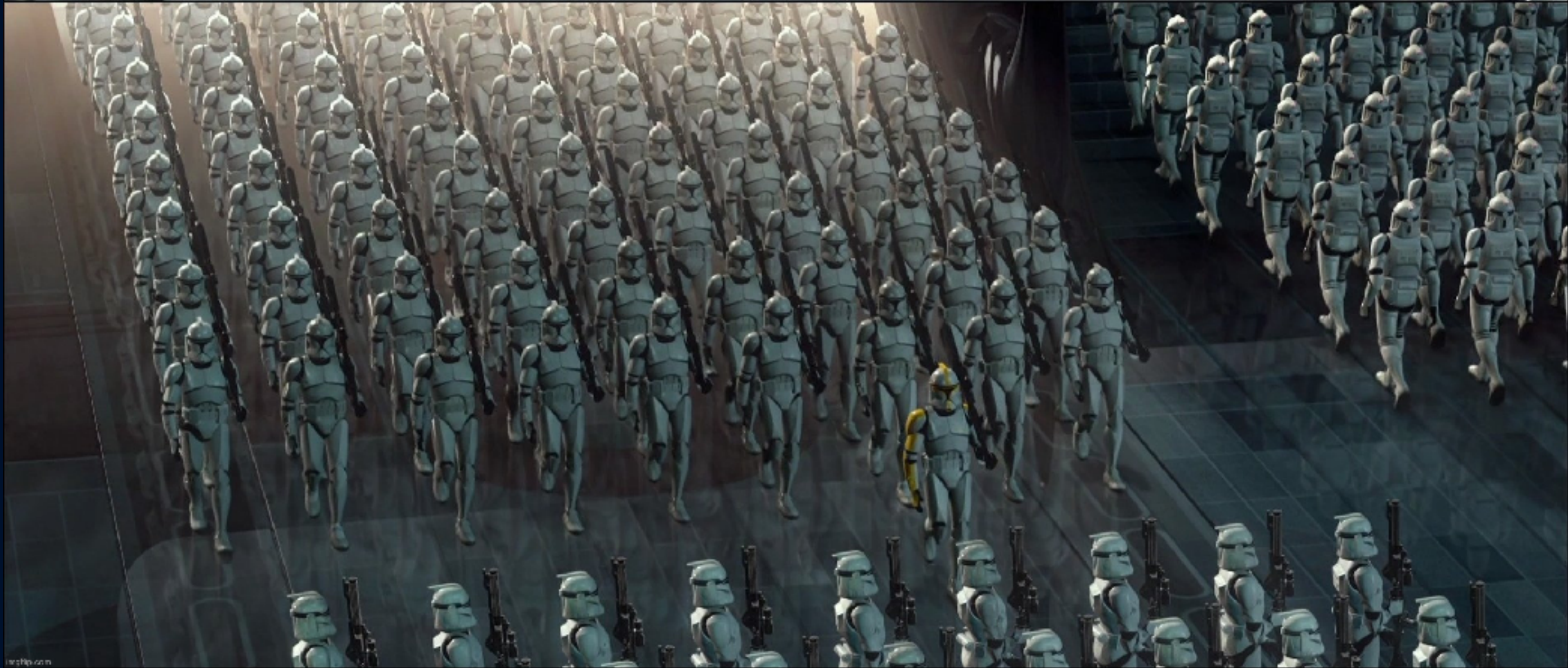
**Storage
Snapshots**

+



**WAL
Backups**

But, and the restores?!





Data, Data
everywhere





Get your columns in order


```
teresal=# \d test_order
```

```
Table "public.test_order"
```

Column	Type	Collation	Nullable	Default
-----+-----+-----+-----+-----				
c1	boolean			
c2	integer			
c3	smallint			
c4	bigint			

```
teresal=# \d test_order
```

```
Table "public.test_order"
```

Column	Type	Collation	Nullable	Default
-----+-----+-----+-----+-----				
c1	boolean			
c2	integer			
c3	smallint			
c4	bigint			

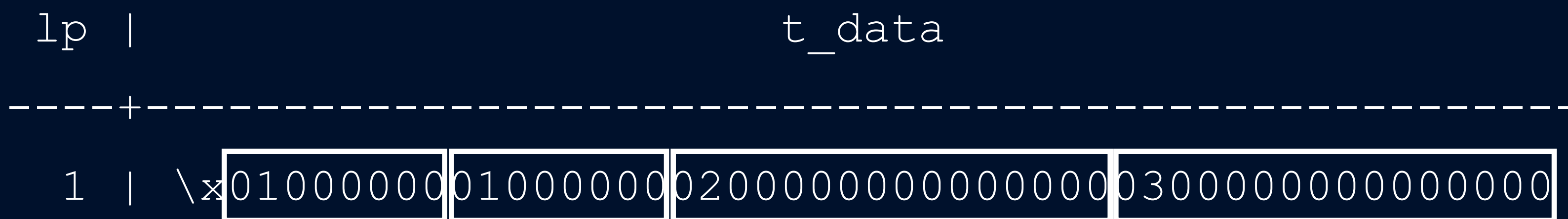
```
teresal=# select * from test_order;
```

c1	c2	c3	c4
----+----+----+----			
t	1	2	3

1p |

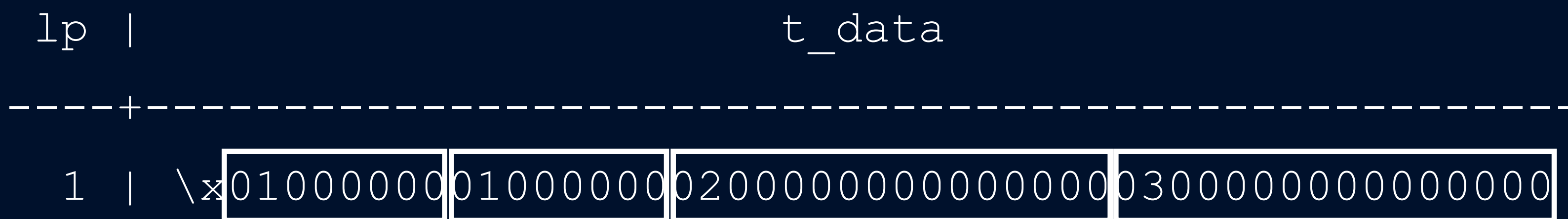
t_data

1 | \x010000000100000002000000000000000003000000000000000



boolean

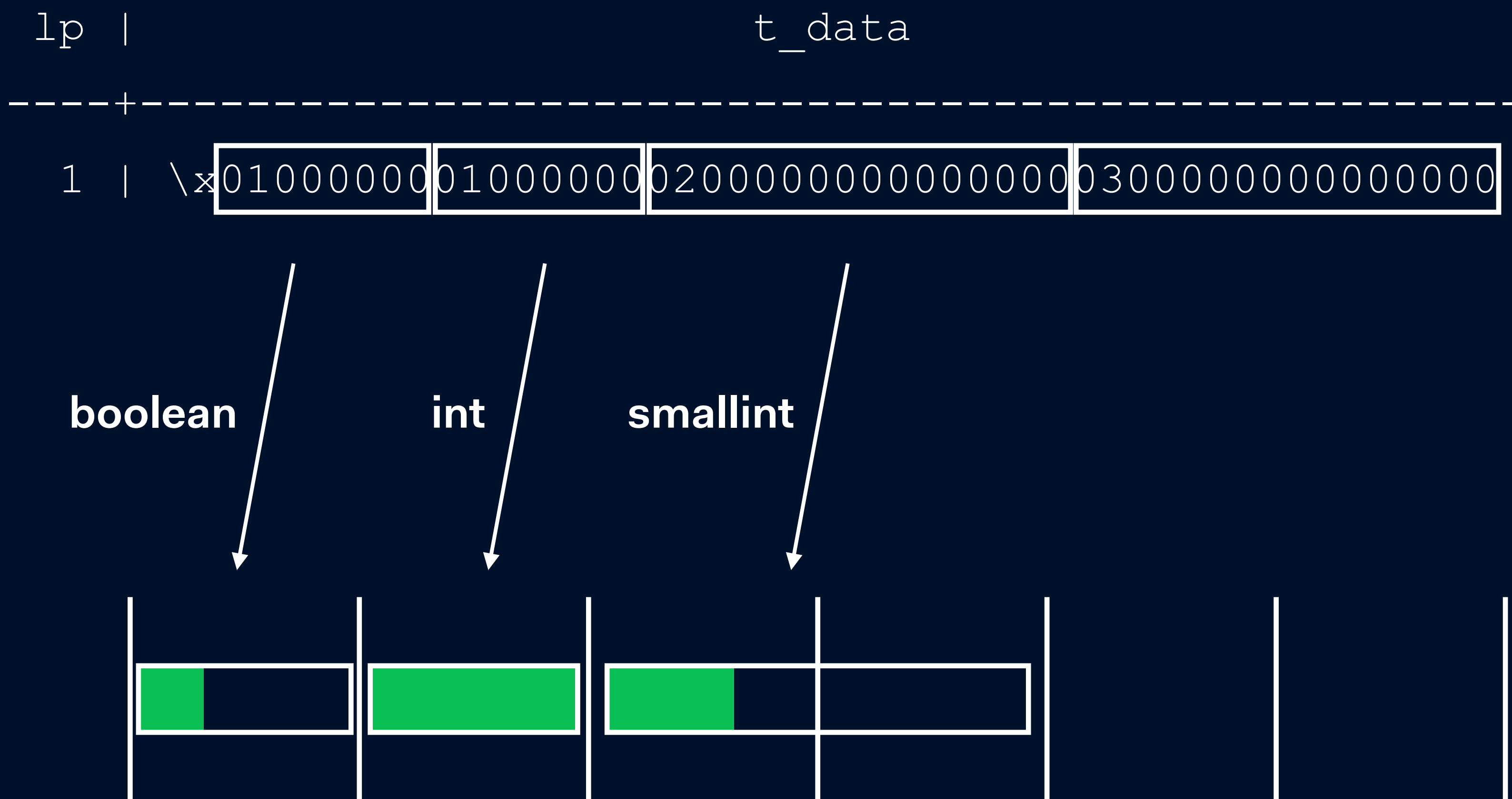


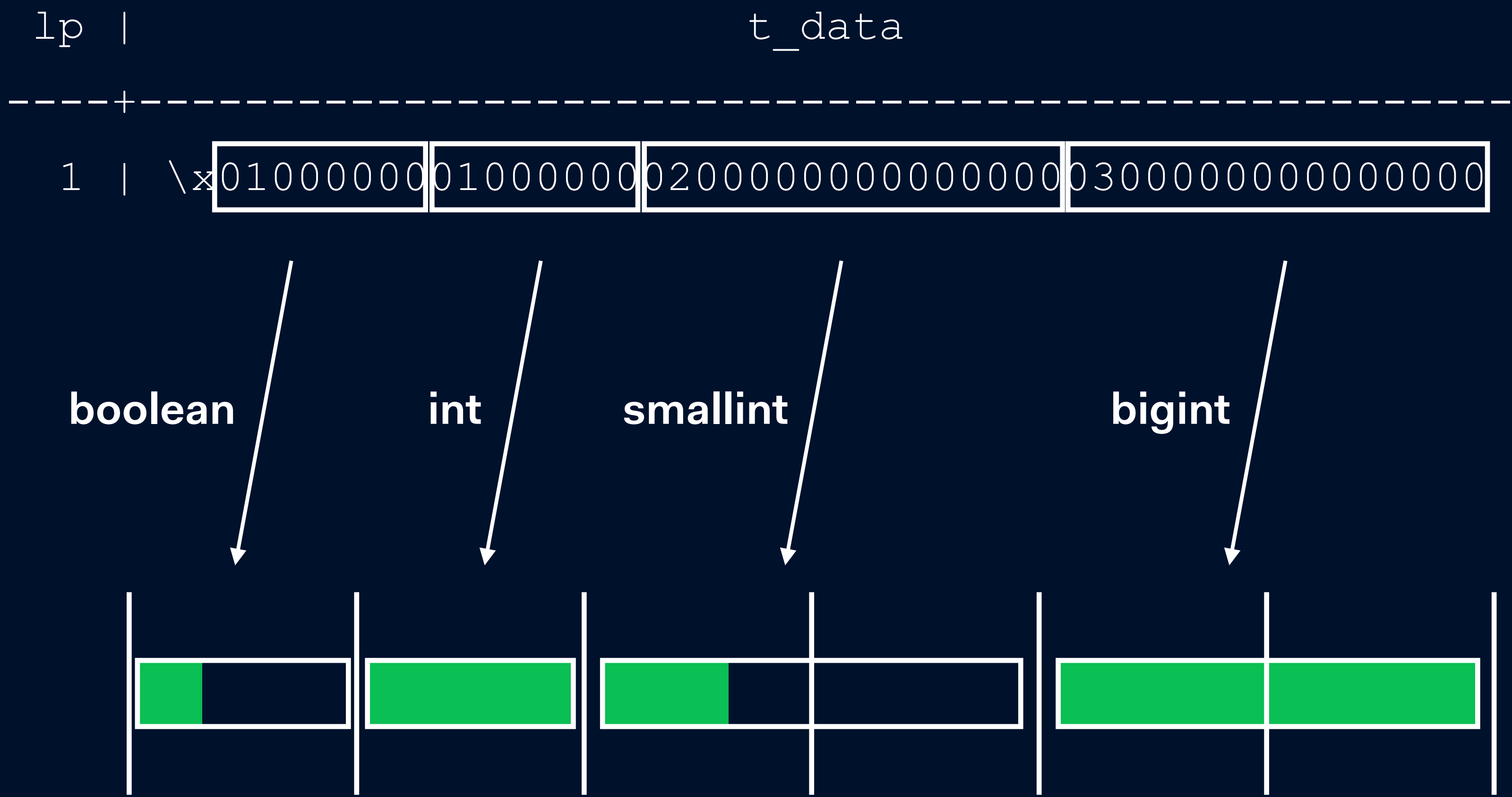


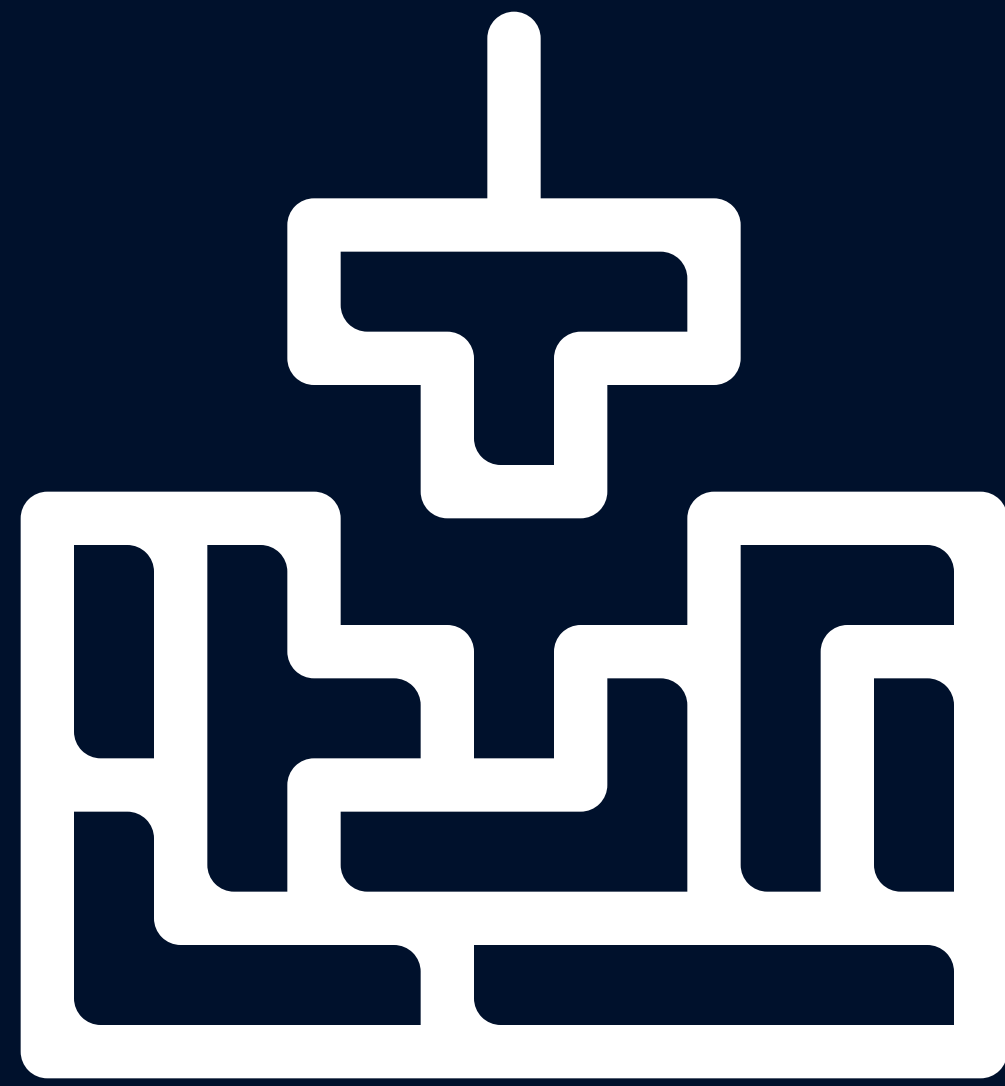
boolean

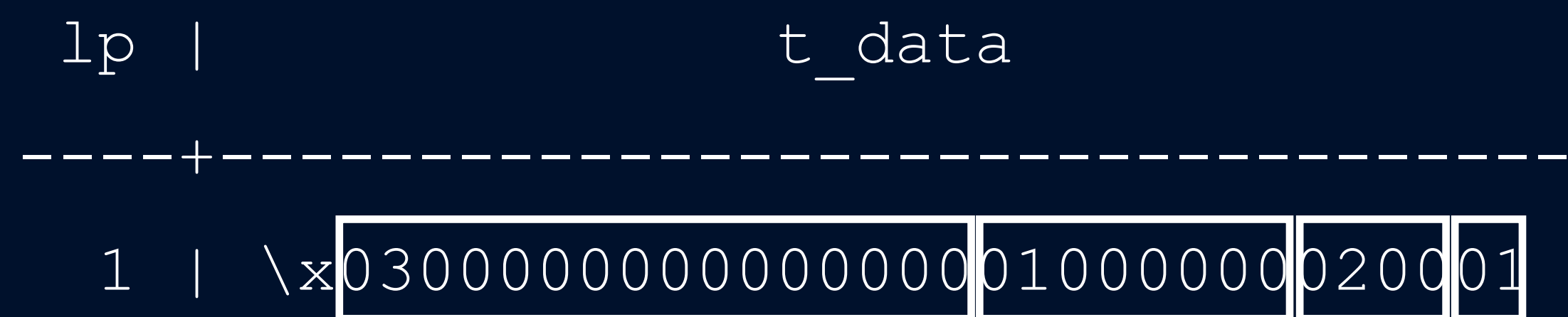
int

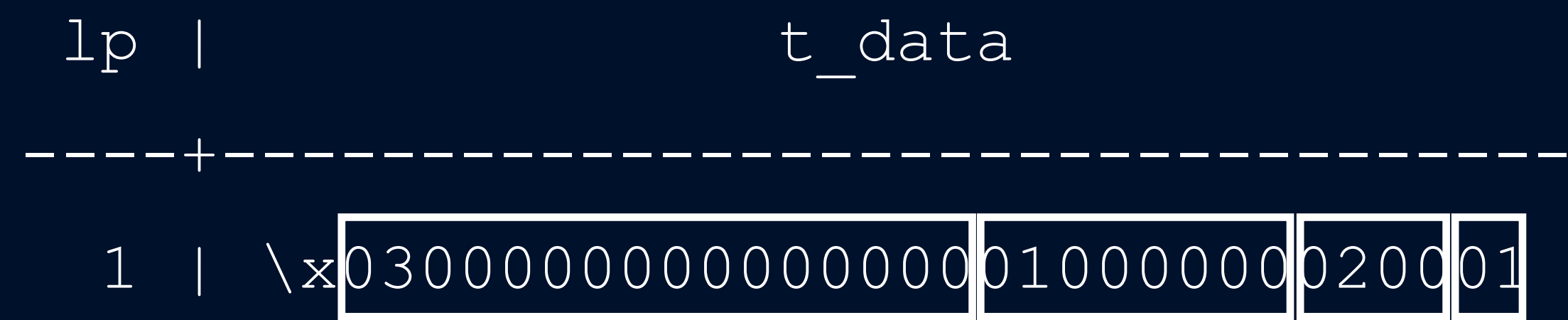






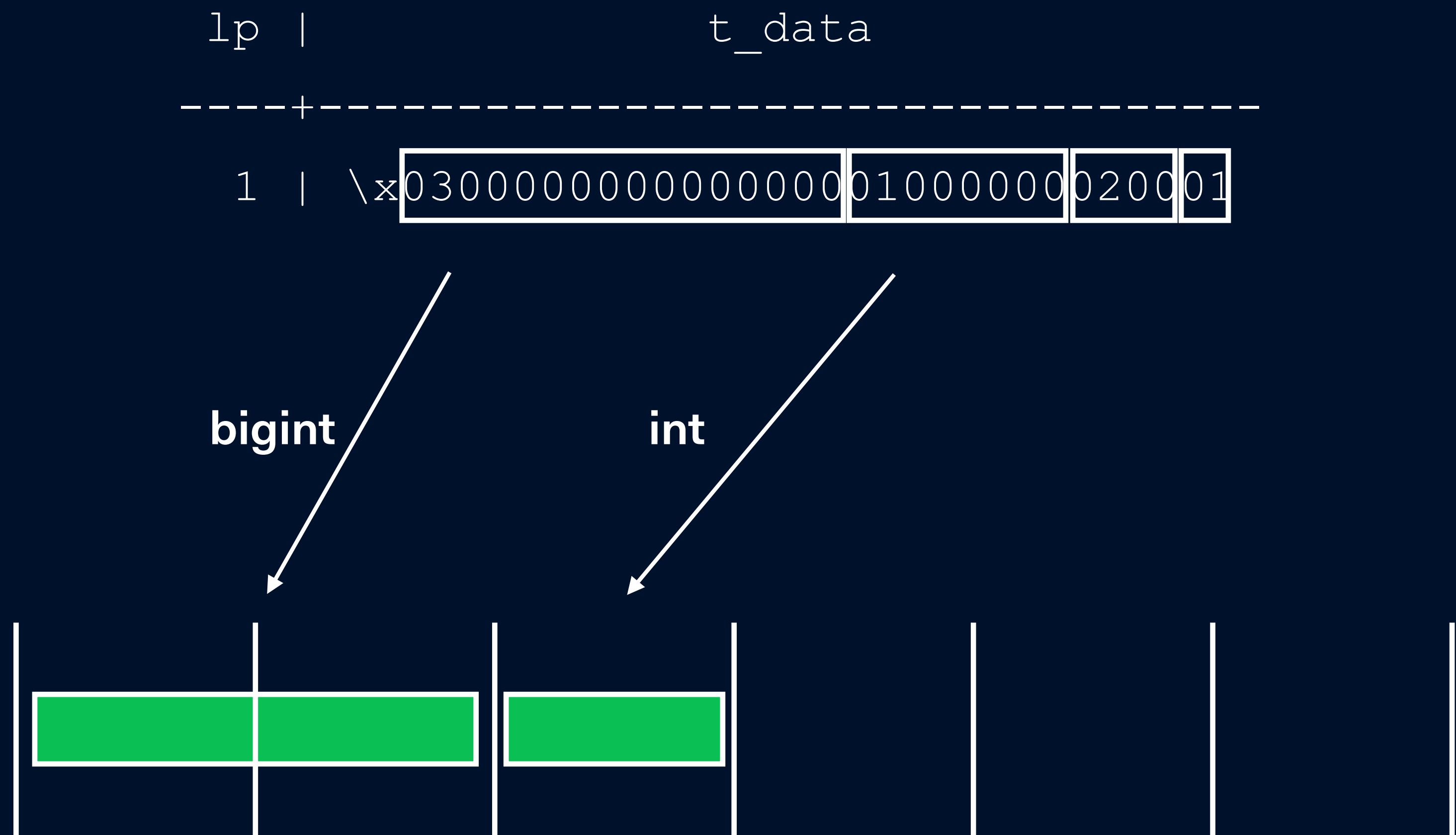


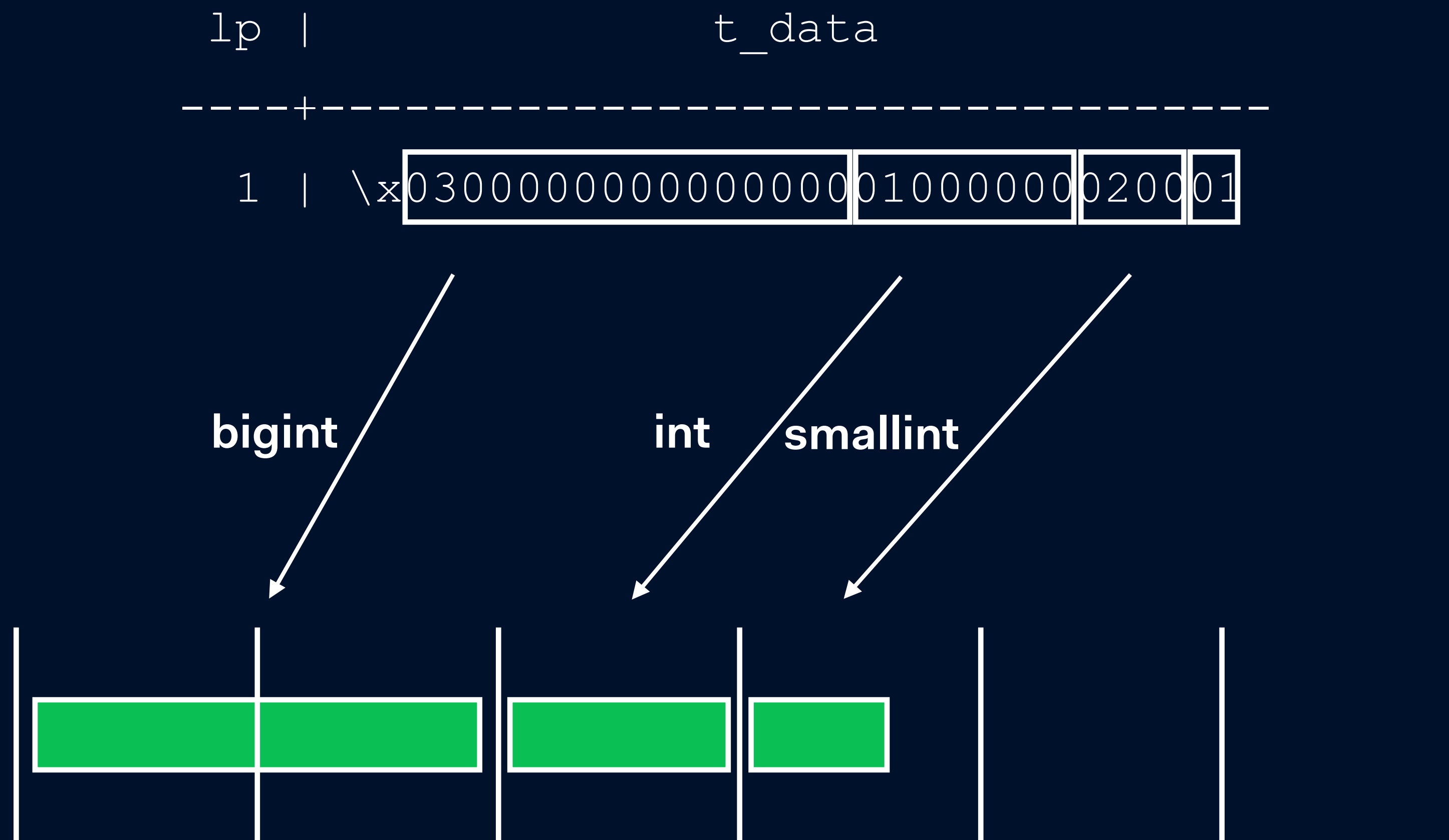


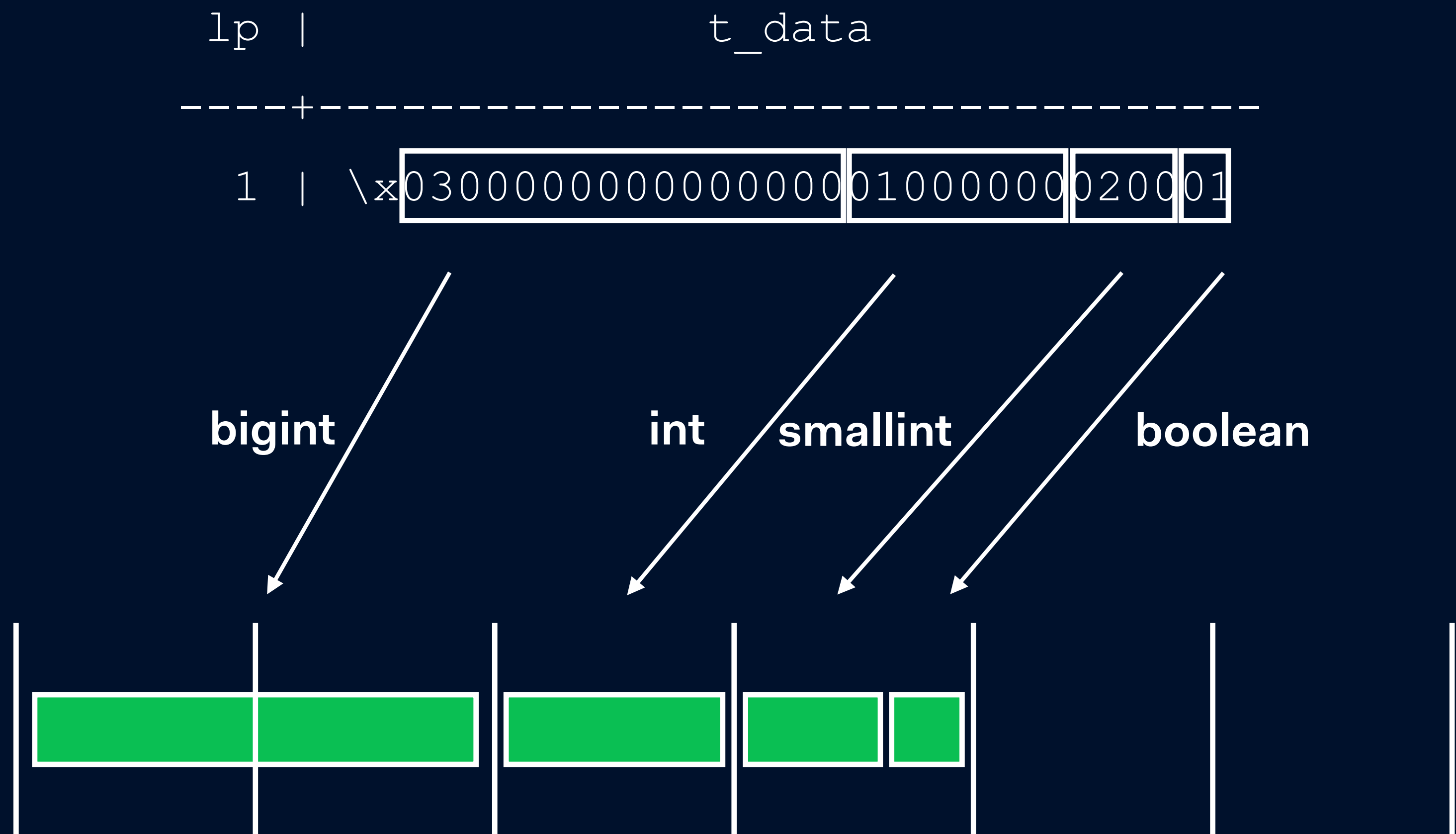


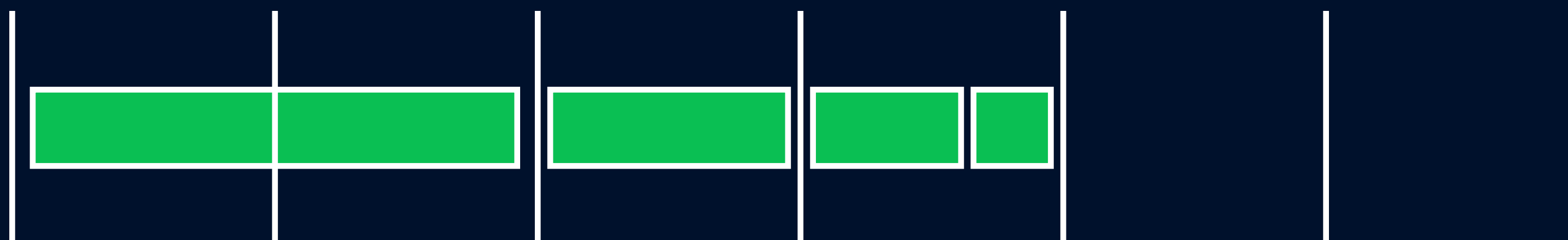
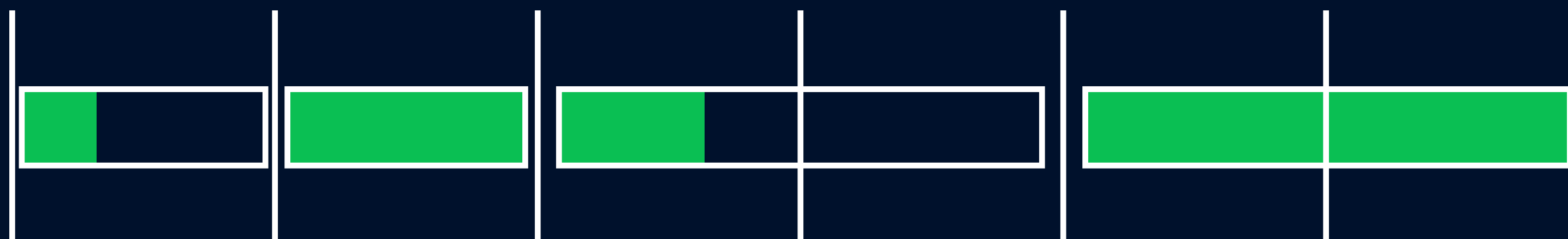
bigint











More “Math”:

10% of +100TB

=

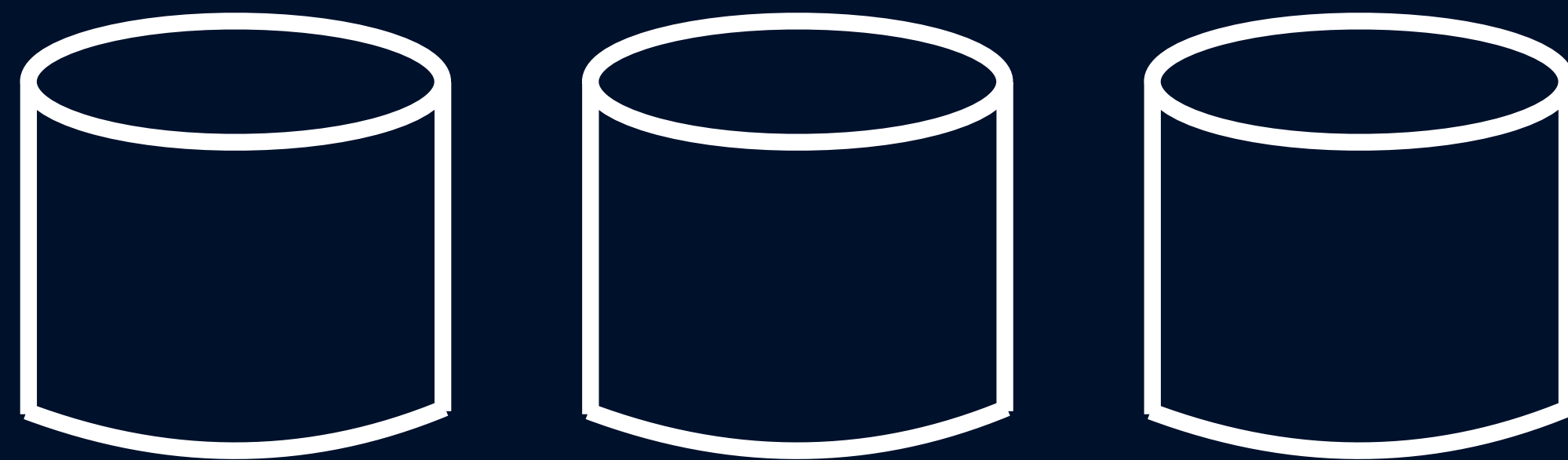
+10TB

More “Math”:

10% of +100TB

=

+10TB

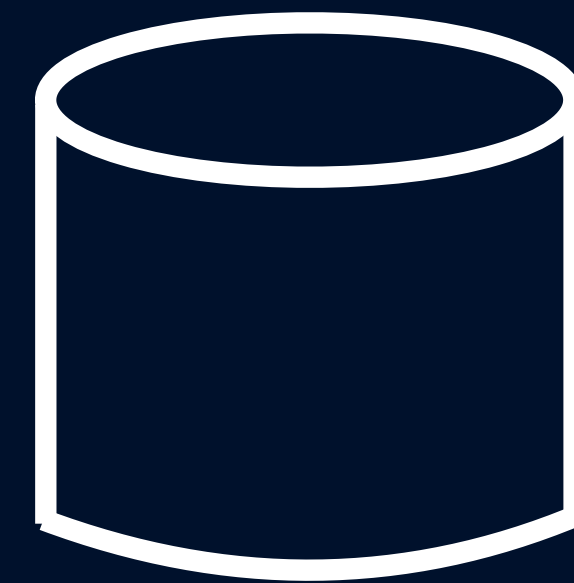
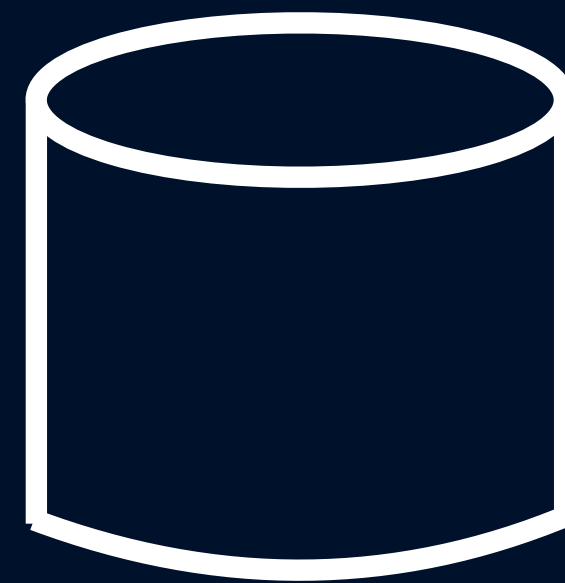


More “Math”:

10% of +100TB

=

+10TB



DECEMBER 6, 2024 • 41 MINUTES

Column Tetris

Nikolay and Michael discuss "Column Tetris" — what it is, why it matters, how to order columns for new tables, and how to re-organise existing ones. Here are some link...



FOSDEM 2024

Reducing Costs and Improving Performance With Data
Modeling in Postgres

Charly
Batista

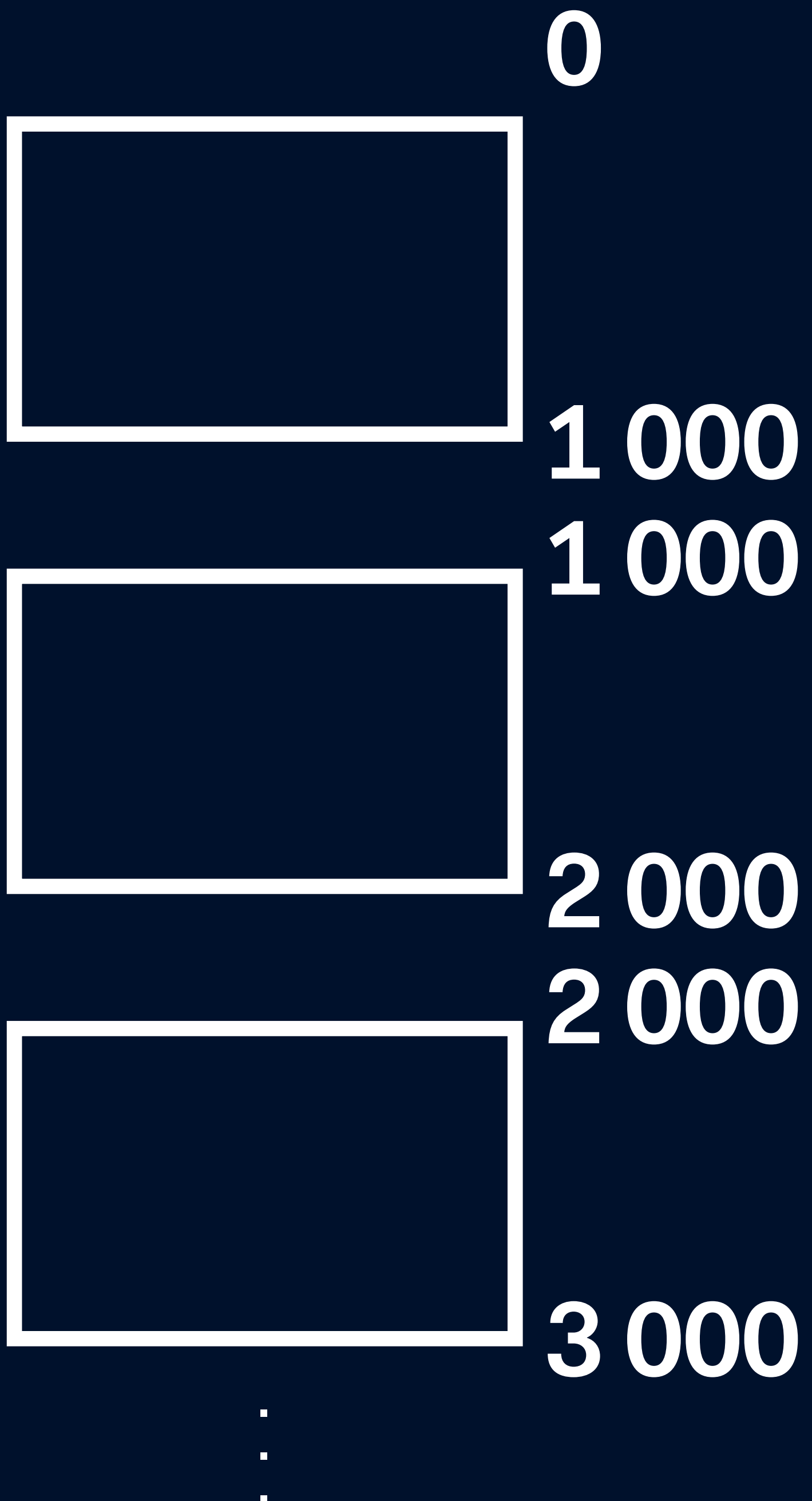
**Tame the chaos,
Partition!**



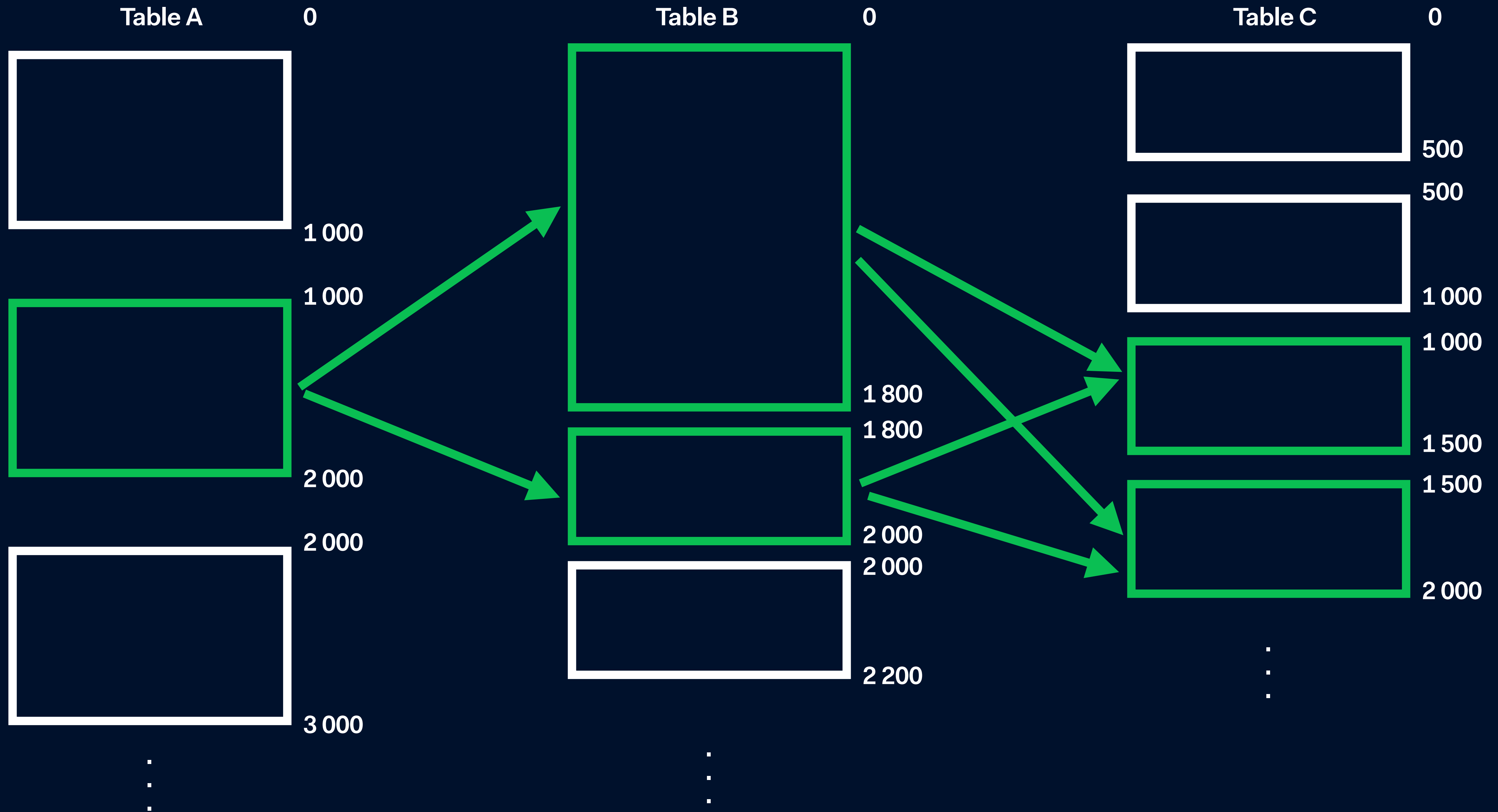
0

4 000





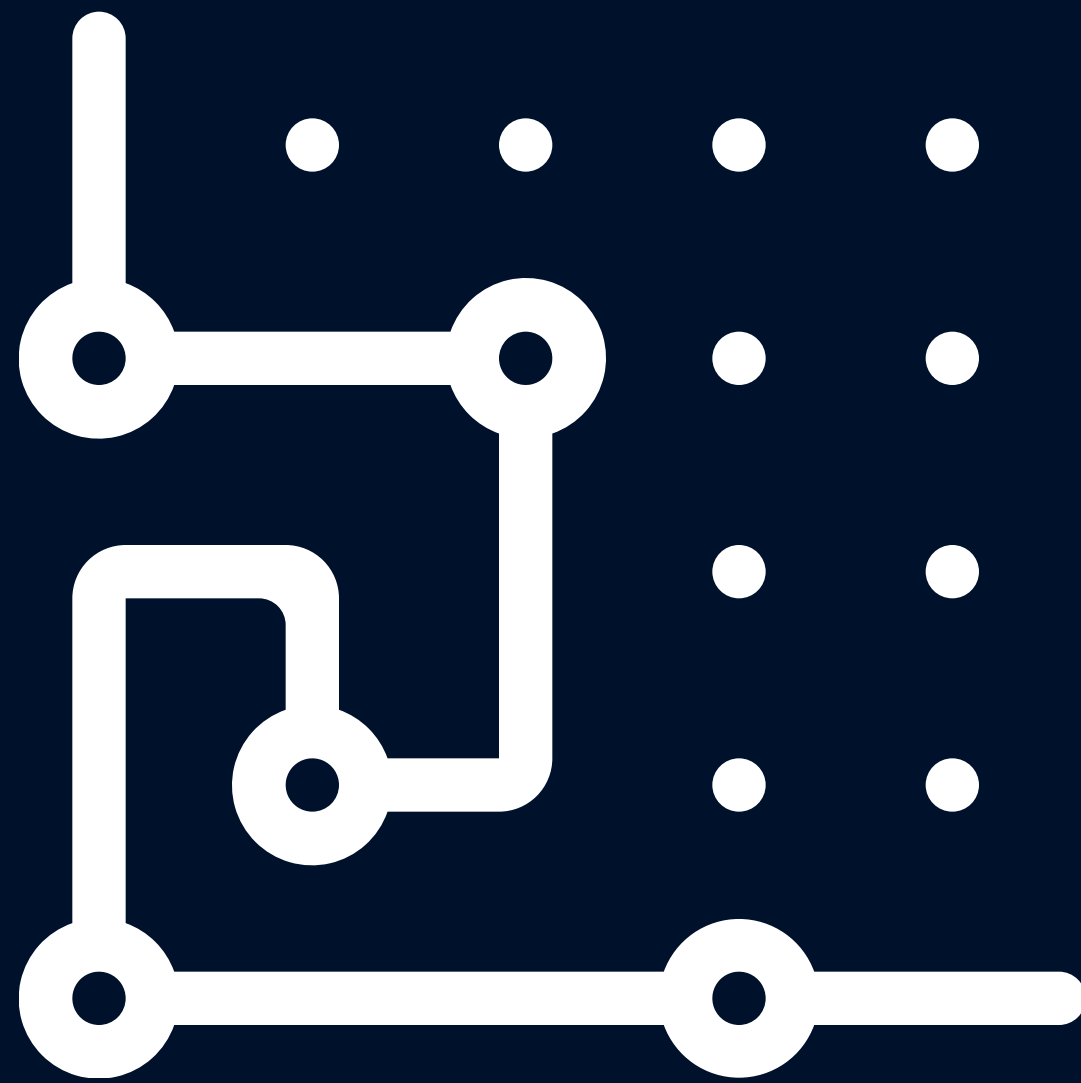
1 table ✓





fast-path locks

LWLock:lock_manager



Planning Time: 12.389 ms

Execution Time: 0.164 ms

Partitioning at **Adyen**

[📖 README](#) [💡 Code of conduct](#) [📄 MIT license](#) [🔒 Security](#) [☰](#)

adyen-postgres-partitioning

These functions are designed to create and maintain partitions in PostgreSQL with a minimal impact on the applicaton. The priority is to not impact the application. When multiple options are available the weakest lock possible is being used, when a heavy lock is required we use a timeout to prevent long lasting locks.

Every function in this project starts with a detailed comment on what the function does and how to use it.



Partitioning at **Adyen**

Tech stories

A Deep Dive into Table Partitioning part 1 🐰: Introduction to Table Partitioning

By Cosmin Octavian Pene (Java Engineer) & Derk van Veen (Database Engineer),
Adyen

Partitioning at Adyen

Tech stories

A Deep Dive into Table Partitioning part 1 🐰: Introduction to Table Partitioning

By Cosmin Octavian Pene(Java Engineer) & Derk van Veen (Database Engineer),

Adyen Tech stories

A Deep Dive into Table partitioning 🐰 Part 2: Partitioning at Adyen

By Derk van Veen (Database Engineer) & Cosmin Octavian Pene(Java Engineer),
Adyen

Partitioning at **Adyen**

Tech stories

A Deep Dive into Table Partitioning part 1 🐰: Introduction to Table Partitioning

By Cosmin Octavian Pene(Java Engineer) & Derk van Veen (Database Engineer),
Adyen Tech stories

A Deep Dive into Table partitioning 🐰 Part 2: Partitioning at Adyen

By Derk van Veen (Database Engineer) & Cosmin Octavian Pene(Java Engineer),
Adyen

Tech stories

A Deep Dive into Table partitioning 🐰 Part 3: Maintenance Under Pressure

By Derk van Veen (Database Engineer) & Cosmin Octavian Pene(Java Engineer),
Adyen.

Partitioning at **Adyen**

Tech stories

A Deep Dive into Table Partitioning part 1 🐰: Introduction to Table Partitioning

By Cosmin Octavian Pene(Java Engineer) & Derk van Veen (Database Engineer),
Adyen Tech stories

Tech stories

A Deep Dive into Table partitioning 🐰 Part 3: Maintenance Under Pressure

By Derk van Veen (Database Engineer) & Cosmin Octavian Pene(Java Engineer),
Adyen. Tech stories

A Deep Dive into Table partitioning 🐰 Part 2: Partitioning at Adyen

By Derk van Veen (Database Engineer) & Cosmin Octavian Pene(Java Engineer),
Adyen

A Deep Dive into Table partitioning 🐰 Part 4: How the default partition saved the day

By Derk van Veen, Database Engineer

Data Temperature







Query Performance

Prepare your Statements

Prepare your Statements (Carefully)



Plans

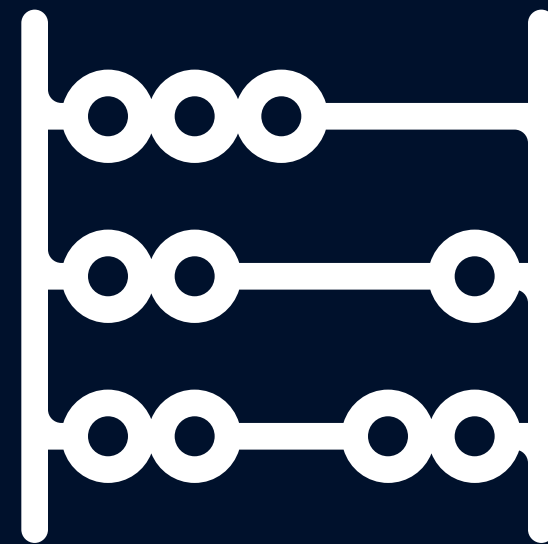
Generic

```
plan_cache_mode=force_generic_plan
```

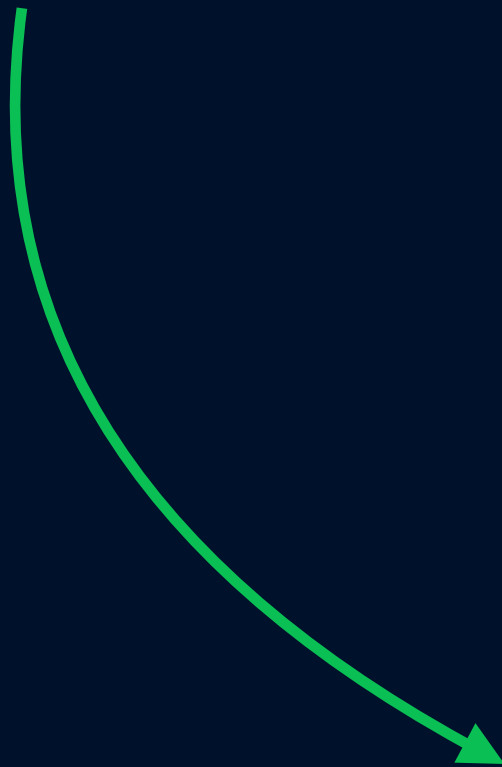


Custom

```
plan_cache_mode=force_custom_plan
```



PostgreSQL v16



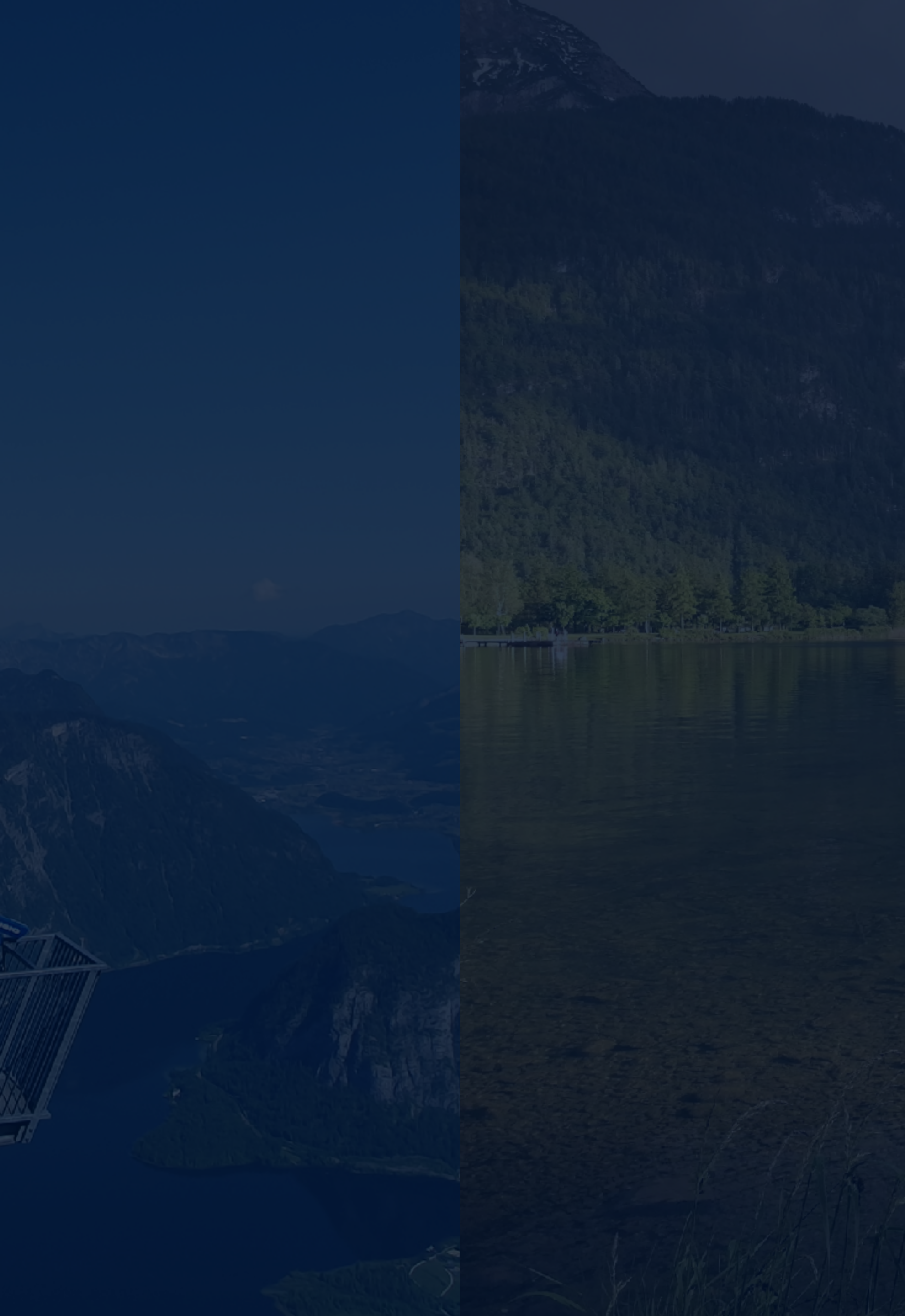
Add EXPLAIN option `GENERIC_PLAN` to display the generic plan for a parameterized query

(Laurenz Albe)

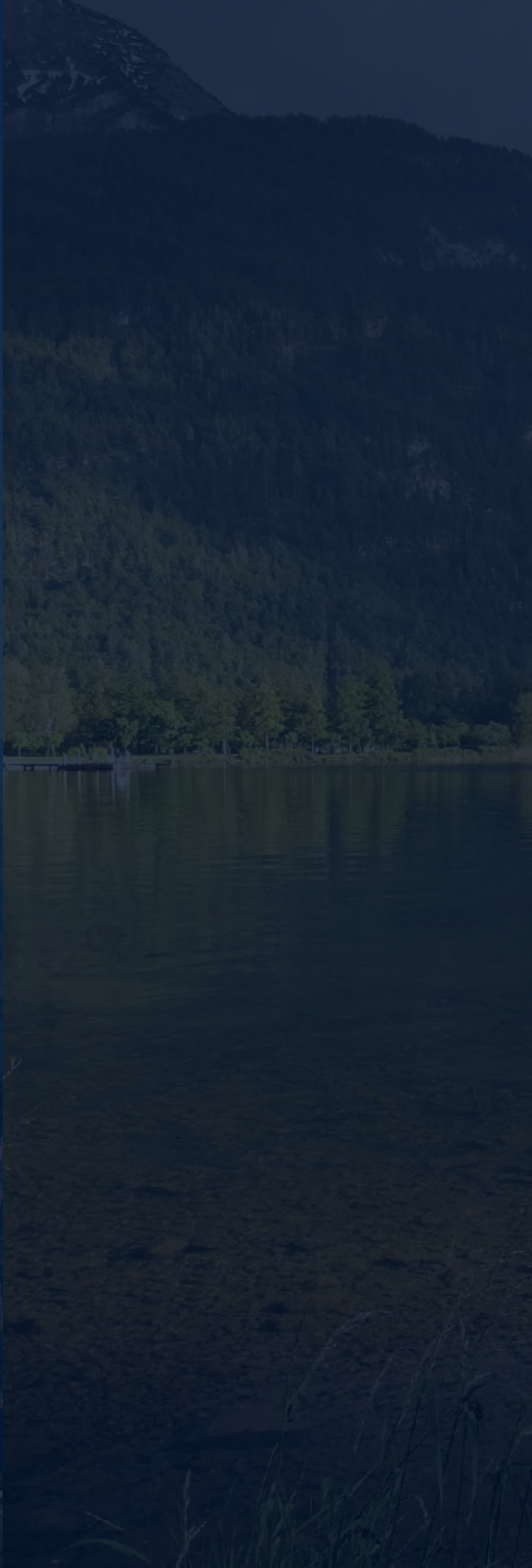
Wrap Up











**What works at 100GB will
fail catastrophically at
+100TB.**

Operational hazards of managing PostgreSQL DBs over 100TB