Lessons for Postgres at Scale

How to Tame a Mastodon

David Christensen

PGConf NYC 2022

~~How to Tame Mastodon~~

About me

Full Stack Dev Background

Postgres Consulting

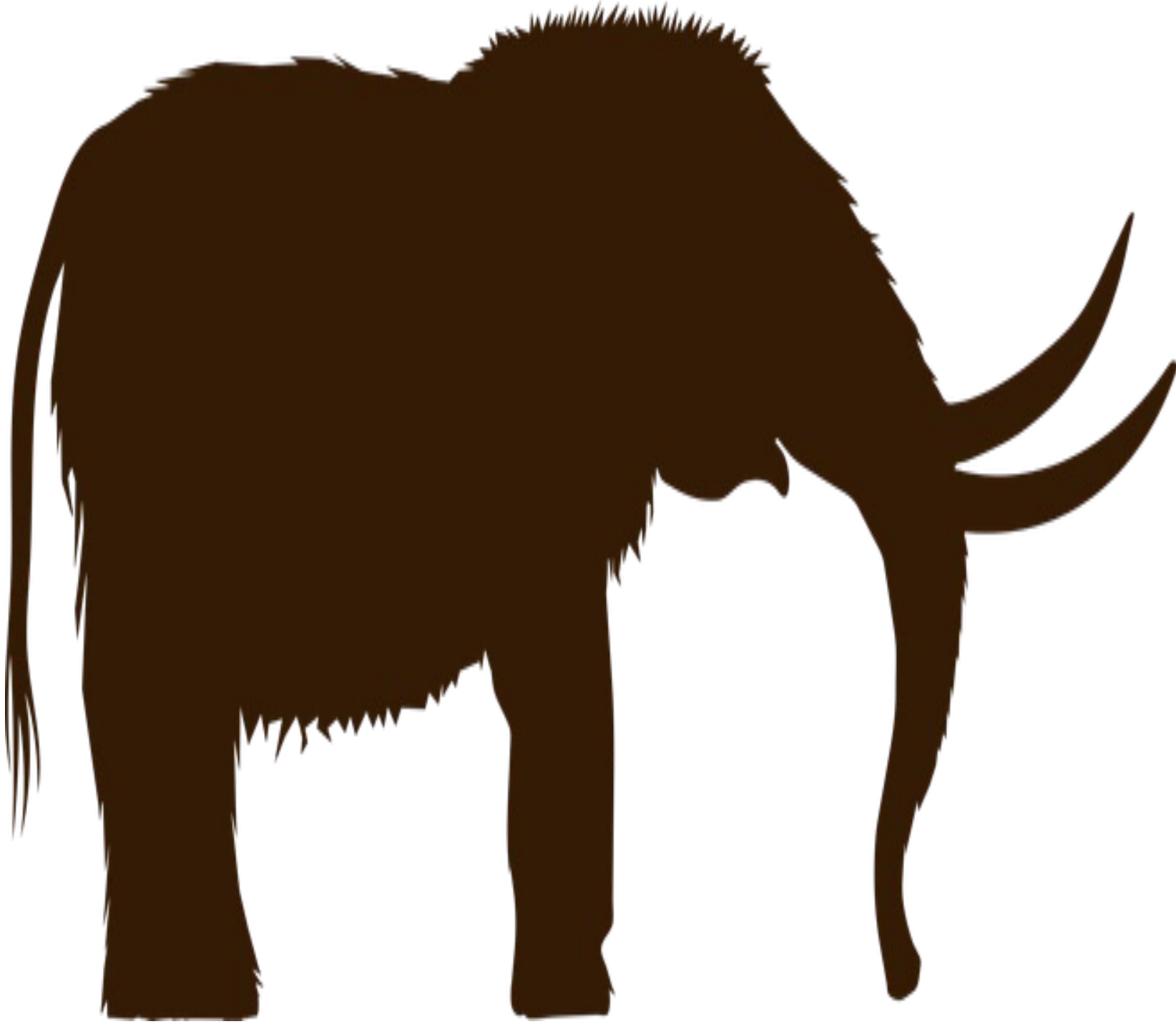
Principal Database Engineer, Solutions Engineering

crunchydata

@pg_dwc

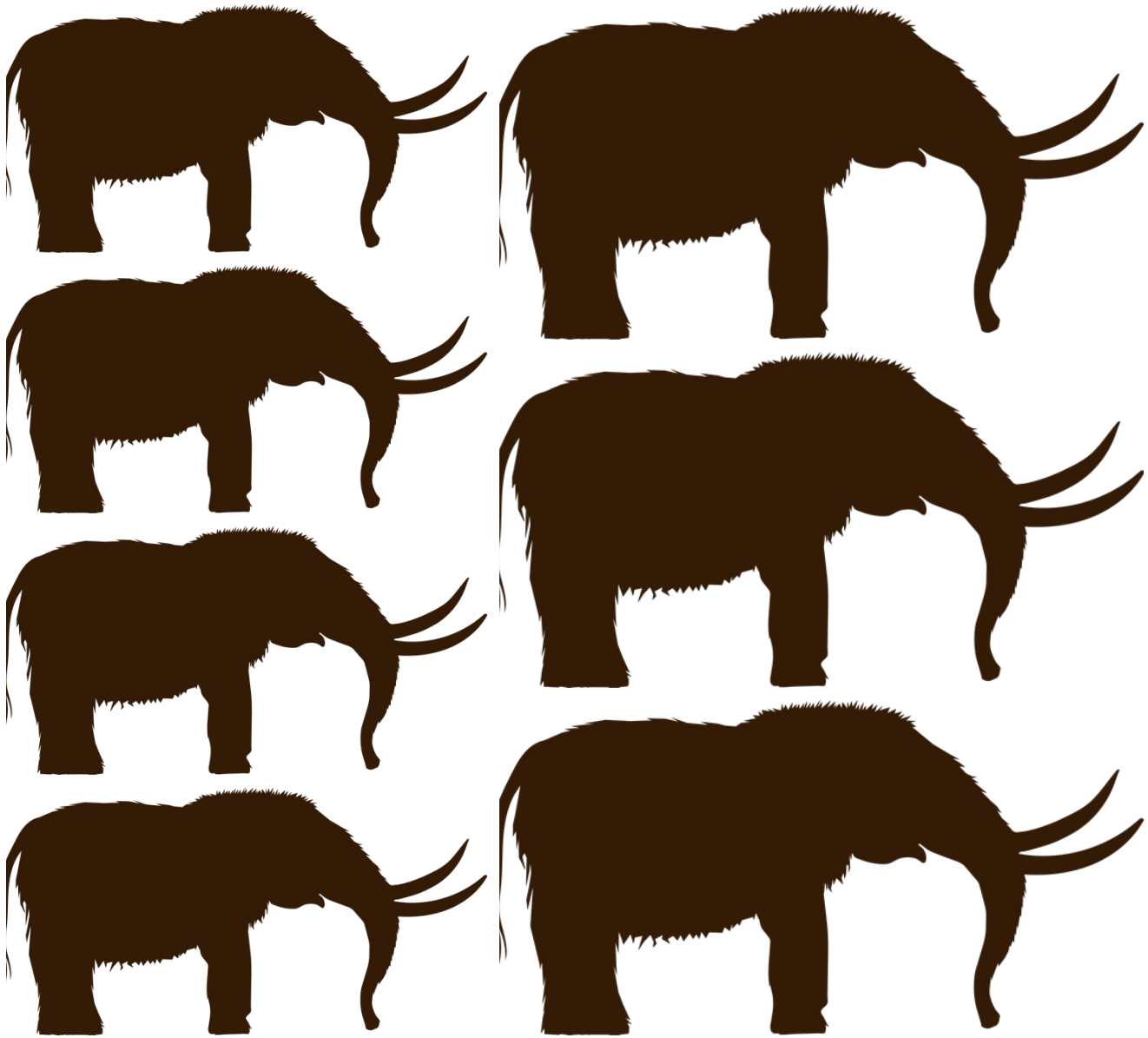
What's the Deal with Big Databases?

Large Data



Transactions

Replicas



Real Life Example Environment

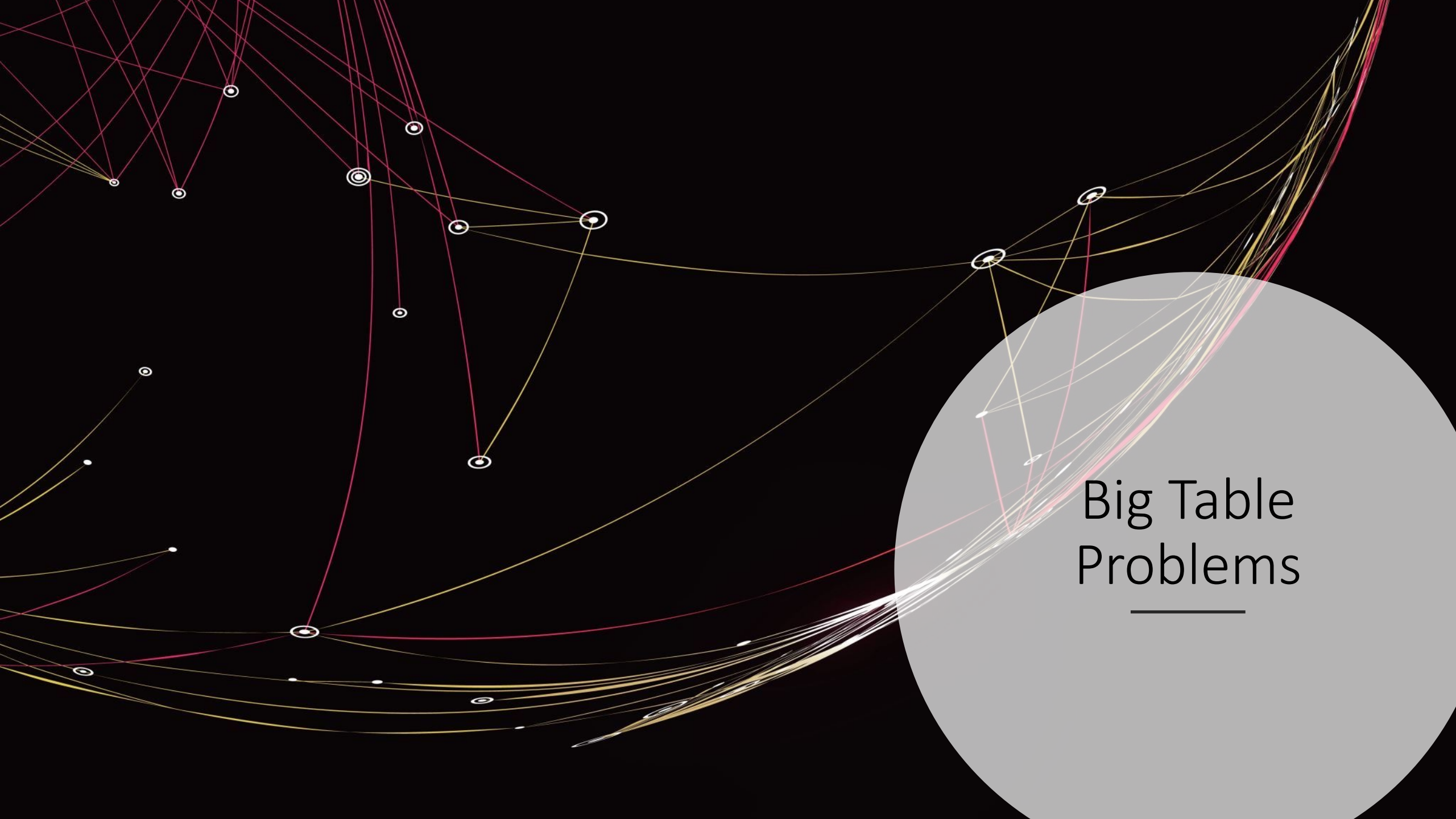
- OLTP Database
- 100 dev, no DBA
- Heroku to Crunchy Bridge migration
- 25TB database with significant growth
- 15 replicas
- WAL volume of 150GB/hr
- Transaction volume of 230M+/hr on primary alone


THIS IS FINE.

Challenges & Solutions

Maintenance needs to be done, but maintenance has risks

- Table size
- Transactions
- Replicas



Big Table Problems

Big Table Problem: Adding Columns

```
ALTER TABLE restaurant ADD COLUMN feedback TEXT DEFAULT compliments_to_the_chef()
```

Locks during rewrite

Solution: Multi-step column changes

ADD COLUMN

- **ALTER TABLE** restaurant ADD **COLUMN** feedback

ADD DEFAULT

- **ALTER TABLE** restaurant ALTER **COLUMN** feedback DEFAULT compliments_to_the_chef()

UPDATE

- **UPDATE TABLE** restaurant SET feedback **TO DEFAULT**

Big Table Problem: Adding Constraints

```
ALTER TABLE
favorite_bands ADD
CONSTRAINT name_check
CHECK (name = 'Led Zeppelin')
```

Locks during creation

Solution: Postpone Validation

ALTER

- **ALTER TABLE** favorite_bands ADD **CONSTRAINT** name_check CHECK (name = 'Led Zeppelin') **NOT VALID**

VALIDATE

- ALTER TABLE favorite_bands VALIDATE **CONSTRAINT** name_check

Big Table Problem: Index Creation

The naïve approach is time consuming and locks

CREATE INDEX ON customers (last_name, first_name)

Locking **bad**

Solution: Create Index Concurrently

CREATE INDEX CONCURRENTLY

- Quick lock
- Runtime tradeoff
- You break it you clean it up

Big Table Problem: Unused Indexes

- Queries change
- High index overhead
- Some redundant indexes

Solution: Index analysis/cleanup

Combine indexes

- Combine **btrees where they make sense**

Look for unused indexes

- Pg_stat_user_indexes
- Pg_statio_user_indexes

Gather data on all nodes

- Unused on primary != unused in cluster
- Reset for active stats

Big Table Problem: Skewed Data

- Data skews
- Falling back to seq scan
- Still maintain whole index

Solution: Partial indexes

CREATE INDEX foo … WHERE bar = 1

- Handy for data skew

Only has data for qual

- Faster updates
- Only relevant data

Big Table Problem: Vacuum

Vacuum required for performance

High xacts = wraparound autovacs

Long vacuums & less frequent vacuums

Solution: Vacuum-related configuration

Tune autovacuum

- Autovacuum_workers = 6
- Maintenance_work_mem = 30GB

Per-table tuning

- autovacuum_vacuum_insert_scale_factor=0, autovacuum_vacuum_insert_threshold=<constant>

Target daily vacuums

Big Table Problem: Large single table

- Many Rows
- Recent vs Historical
- Vacuum, Indexes
- Periodic Data Removal

Huge table solution: Partitioning

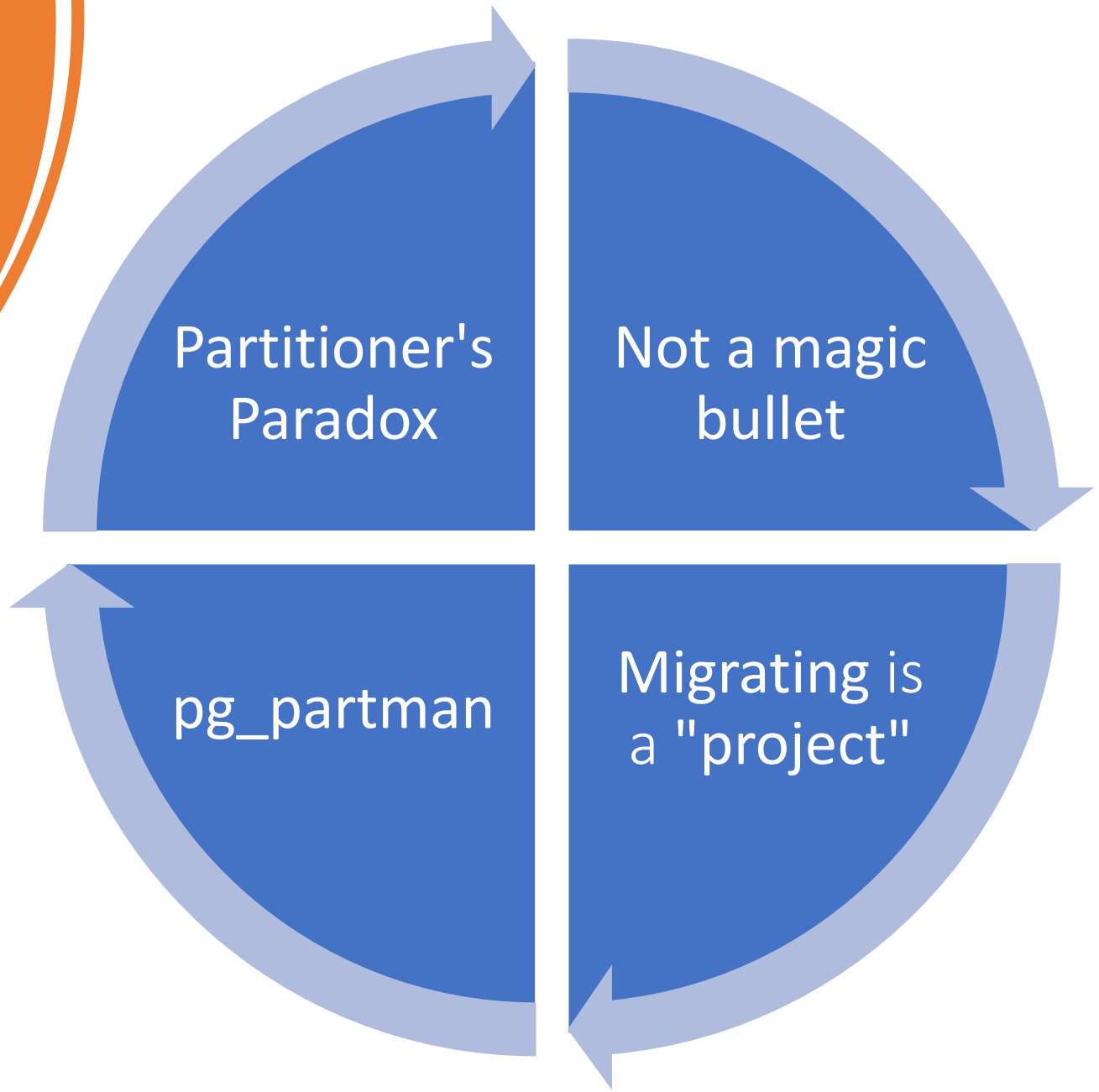
Break into smaller tables

- Queries don't need to know
- Can tune/index partitions individually

Helps with data lifecycle management

- ATTACH PARTITION
- DETACH PARTITION
- Sometimes performance

Partitioning Caveats



Big Table Problem: Wasted Table Space

Column Order affects Padding

Wasted space

Many column issue

Big table solution: Optimize Table Size

Order by size

- Fixed-size, **largest to smallest**
- Variable length or NULLable last

Non-trivial

- (bool, bigint, bool, bigint, bool, bigint) = 72 bytes
- (bigint, bigint, bigint, bool, bool, bool) = 52 bytes
- 30% savings

Caveats

High Transaction Challenges

High Transaction Problems: WAL Generation

Archives single files

Restore can't catch up with archive

No breathing room for replay

WAL Size Workarounds

Pgbackrest

- Async archive/restore
- Daemon mode

Force restore failure

- Switches to streaming
- Replica catches up

High Transaction Problems: High xid Misc

- Anti-wraparound vacuums
- SAVEPOINT/Subtrans overflow
- SERIAL limits

Solutions

Tune VACUUM

- We covered

Remove SAVEPOINTs

- Will cover

Use bigints for ids

- 8 bytes is big enough for anyone

High Transaction Problems: pgBouncer Limits

100% usage

=Bottleneck

PgBouncer solution: Multi-Bouncer

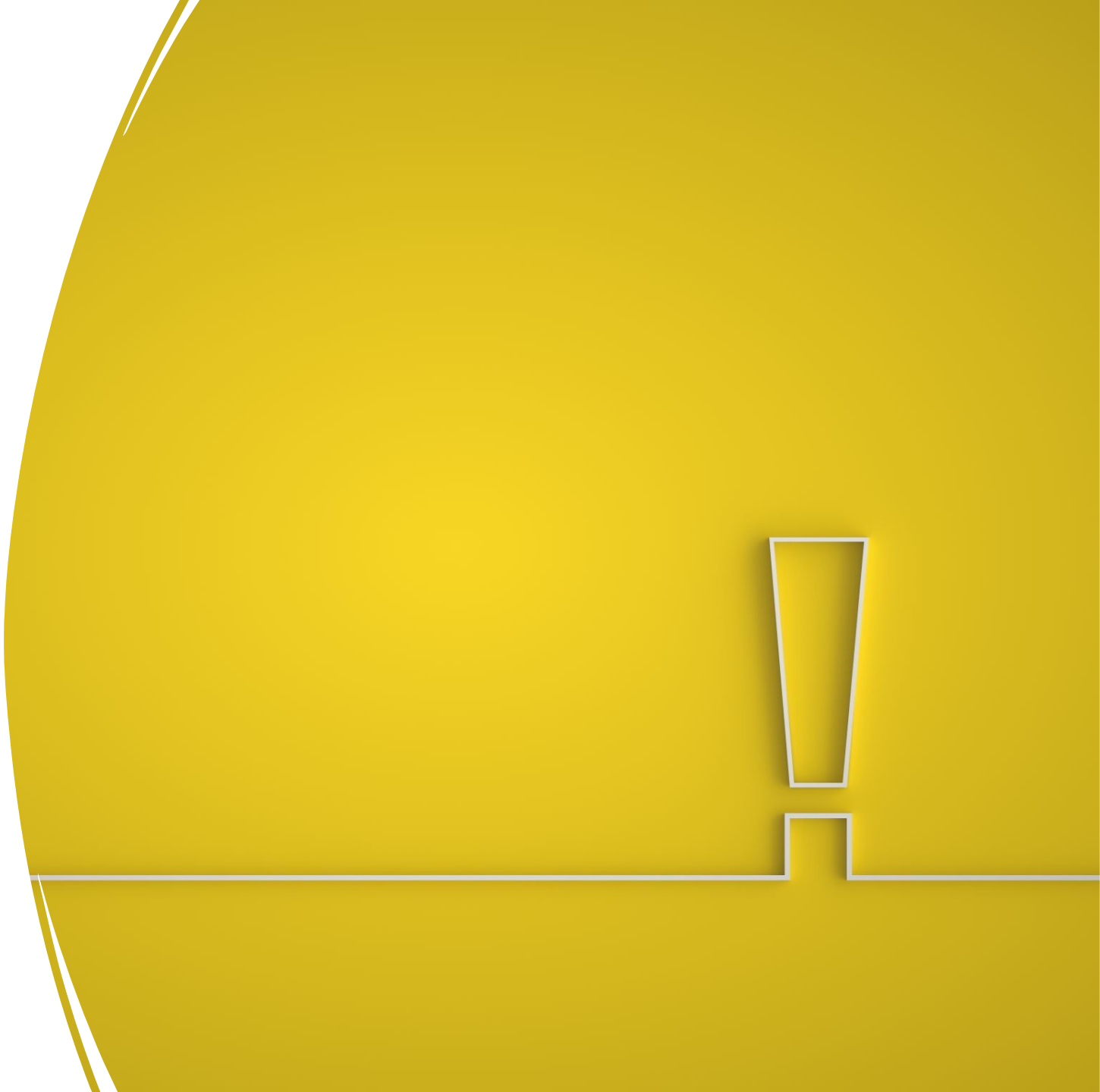
Use Systemd to multiplex

(Semi-)Arbitrary numbers of concurrent PgBouncers

Read Replica Scaling Challenges

Many Replicas: Why?

- Reduce load on the primary
- Redundancy
- Different purposes:
  - HA
  - Load Balancing fast queries
  - Reporting/Analytics
  - Delayed Standby



Read Replica Problems: Management

- Working with replicas at scale
- Manual
- Inconsistent

Replica Management

- Automate tool chains
- Managed Postgres + APIs
- Centralize Monitoring & Data Collection

Read Replica Problems: Different Query Workloads

Replicas have different needs

Replicas have different performance

Solutions for separate query workloads

Look at each machine

- Can't just look at primary
- Pg_stat_* and pg_statio_* views
- Pg_stat_statements, auto_analyze

Tuning/analysis over time/trending

Central point for information

- Can write back to primary to give stats history
- Third-party services

Read Replica Problems: Lag

Locking can cause lag

Need up to date information

Solutions for lag - tuning

- Statement timeouts
- Max standby archive delay
- Max standby streaming delay

Read Replica Problems: SAVEPOINT

Application issues SAVEPOINTS

SAVEPOINTS have to be looked up on disk

Causes lag

Solutions for SAVEPOINT

Underlying app changes

Review subtransactions

Read Replica Problems: Long running analytics

Logical replication not able to keep up

Queries can impact primary

Solutions for long analytics

Hot_standby_feedback

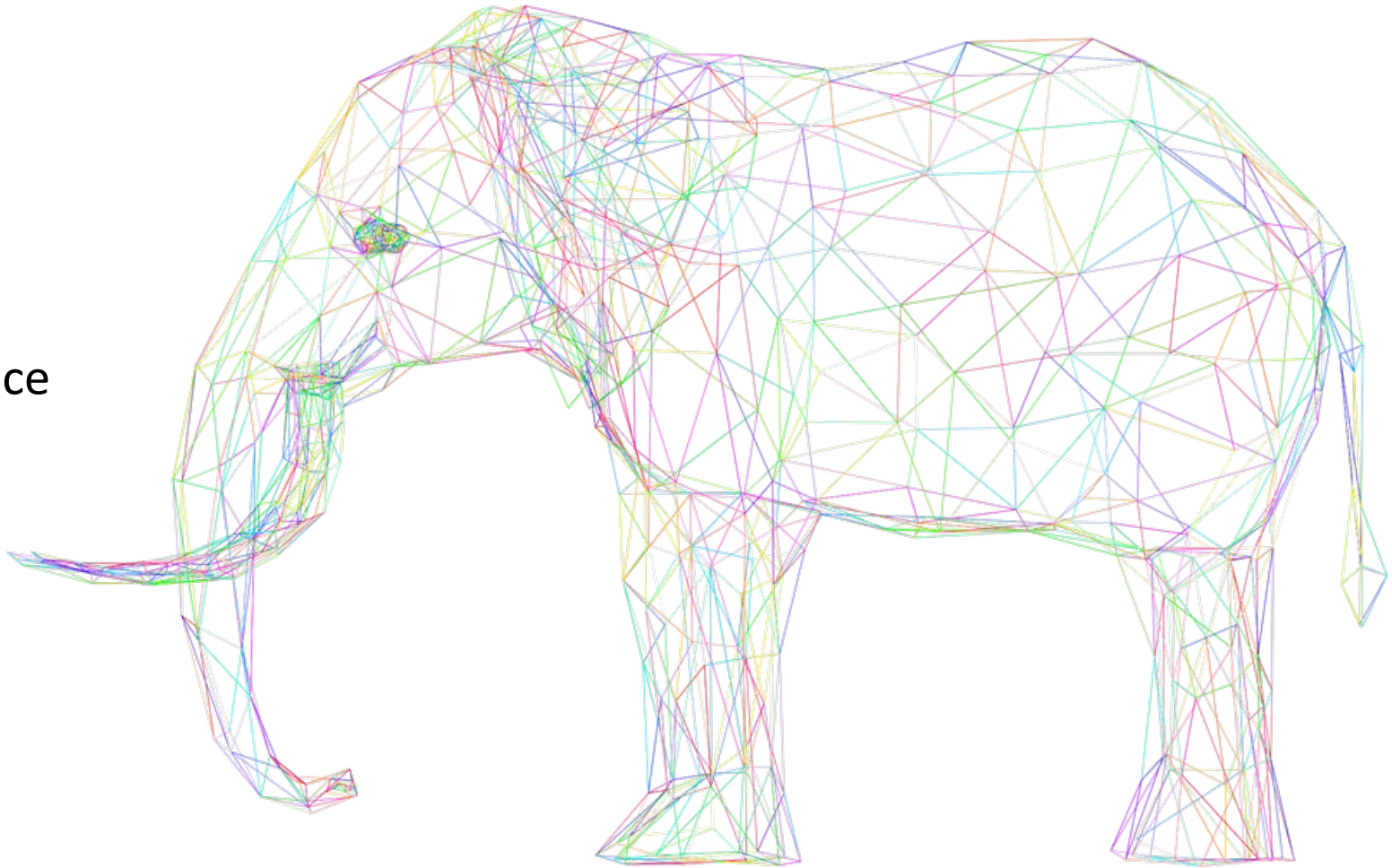
- =off

Archive only replica

- Not streaming
- Dedicating specific replicas to analytics only

How did we tame the Mastodon?

- Minimize locks
- Be smart about indexing
- Per table vacuum tuning
- Partition if you can
- Design schema to minimize space
- Look at multi pgBouncer
- Planned replicas



Thanks to

- Elizabeth Christensen
- Greg Sabino Mullane
- Keith Fiske

crunchydata

Questions?