Database Incident Root Cause Analysis (RCA) & Remediation Guide

Finding H4 | Severity P1 | Owner DBA / Platform Engineer (Cluster GUCs / Vacuum / Pooling / Storage) + Developer (Application SQL / ORM / Migrations) | Environment cloudrun-showcase (synthetic-demo) | Generated 2026-10-27 17:22:57 UTC

1. Incident Overview & Executive Summary

1.1 Executive Summary

CircuitSQL detected finding H4 (Connection saturation above the 4x vCPU budget) in environment cloudrun-showcase (synthetic-demo) at 2026-10-27 17:22:57 UTC. Saturation surfaced as 53300 too_many_connections errors and LWLock:BuffferContent waits that slowed every orders transaction. The correlated report classifies the incident as P1 with triggers H2, H6, H11, H12, pg_stat_statements. CircuitSQL routes remediation to DBA / Platform Engineer (Cluster GUCs / Vacuum / Pooling / Storage) + Developer (Application SQL / ORM / Migrations). Environment cloudrun-showcase (synthetic-demo) reported 4 breached health check(s) across 2 correlated root-cause incident(s) and 2 query offender(s). CircuitSQL routes remediation to Developer (Application SQL / ORM / Migrations) + DBA / Platform Engineer (Cluster GUCs / Vacuum / Pooling / Storage).

1.2 Document Metadata

Field	Value
Incident ID	CSQL-INC-H4-202610271722
Date / Time of Incident	2026-10-27 17:22:57 UTC
Duration	63 min (trigger -25 min to recovery +38 min; see section 4)
Severity Level	P1 - Critical (customer-visible outage or data-risk)
Affected Systems	circuitsql_monitor@db-primary.internal:5432/circuitsql (ssl_mode=require); orders and customers OLTP paths; environment cloudrun-showcase (synthetic-demo)
Database Engine & Version	PostgreSQL 17.2
Lead Investigator(s)	Developer / Marlon Francis Briguera (incident commander); Backend Engineer (query path); DBA / Platform Engineer (cluster controls)
Current Status	Mitigated - monitoring for recurrence; permanent remediation tracked in section 7

2. Technical Environment Context

2.1 System Topography

```
Client / API (FastAPI, app_rw)
        |
        v
[PgBouncer :6432] -- pool_mode=transaction, default_pool_size=4x vCPU
        |
        v
[PostgreSQL primary circuitsql]  << finding H4
        |-- pg_stat_activity / pg_locks / pg_blocking_pids
        |-- pg_stat_statements (offender ranking)
        |-- pg_stat_checkpointer / pg_stat_io (PG 17/18 subsystem counters)
        '-- [Read replica] analytics_ro (reporting, EXPLAIN ANALYZE rehearsals)

[CircuitSQL Monitor] -- H1-H18 probes, CSQL-S01..S07 security scan, RCA export
```

2.2 Privilege & Tool Prerequisites

- Role membership: pg_monitor and pg_read_all_stats (read-only catalog access).
- pg_stat_statements present in shared_preload_libraries and installed in the database.
- PgBouncer admin console reachable on port 6432 for SHOW POOLS and RELOAD.
- psql 17 client plus a shadow database for EXPLAIN (ANALYZE, BUFFERS) rehearsals; never run write DDL against production to inspect a plan.
- ALTER SYSTEM requires a superuser or a role with the ALTER SYSTEM privilege; credentials come from the secret manager, never URL query strings.
- Incident channel and GitLab issue write access for the fix MR linked in section 8.

3. Symptoms & Observed Evidence

3.1 Observed vs. Expected Behavior

Aspect	Observed	Expected
Measured value vs. threshold	H4 measured 42 against 85 (observed).	Active connections stay below 85% of the 4x-vCPU pool budget, with zero LWLock waiters on buffer mapping.
Session and lock state	The finding evaluation detected no unexpected lock queue.	Lock depth <= 1, no idle-in-transaction session, and every statement bounded by lock_timeout and statement_timeout.
Workload impact	orders and customers traffic shows elevated latency or saturation while the condition holds.	p95 within the 250 ms OLTP SLO and connection use below 85% of the pool budget.
Data integrity and compliance	No data loss observed; the control gap maps to SQLSTATE 53300 handling.	Constraints validated, checksums clean, and CIS PostgreSQL 17 controls satisfied.

3.2 System Logs & Error Codes

Application Exception Log

```
2026-10-27 17:10:57.812 UTC [api] ERROR orders.views: request failed route=/orders finding=H4
psycopg.errors.TooManyConnections: sorry, too many clients already
Traceback (most recent call last):
  File "app/routes/orders.py", line 88, in create_order
    session.commit()
psycopg.errors.DatabaseError: FATAL: sorry, too many clients already (SQLSTATE 53300)
```

RDBMS Error Log

```
2026-10-27 17:10:57.101 UTC [41892] LOG: process 41892 still waiting after 1000.042 ms
FATAL: sorry, too many clients already (SQLSTATE 53300)
2026-10-27 17:10:58.204 UTC [41892] STATEMENT: /* finding=H4 */ SELECT count(*) AS total_connections, count(*)
 FILTER (WHERE wait_event_type = 'LWLock') AS lwlock_waiters FROM pg_st...
2026-10-27 17:10:59.318 UTC [41892] DETAIL: causal_tag=[CSQL-M07 | DIRECT] sqlstate=53300
```

4. Timeline of Events

Phase	UTC Timestamp	Event	Evidence
Trigger	2026-10-27 16:57:57 UTC	H4 condition established: Connection saturation above the 4x vCPU budget	finding=H4; tag=[CSQL-M07 | DIRECT]
Alerting	2026-10-27 17:04:57 UTC	CircuitSQL monitor evaluated the probe and raised the finding on the console.	environment=cloudrun-showcase (synthetic-demo); breached=false

Phase	UTC Timestamp	Event	Evidence
Degradation	2026-10-27 17:10:57 UTC	orders and customers traffic degraded: latency rose and lock or buffer pressure became visible.	H4 measured 42 against threshold 85 (cited) and is within threshold: 42 backends vs 4x21=84 rule of thumb (max_connections=100, 2 in LWLock wait); active_non_idle=18 client backends, active_session_ratio=18.0% of max_connections (budget 25%)
Triage	2026-10-27 17:22:57 UTC	On-call opened the per-finding RCA page and confirmed the mechanism against pg_stat_activity, pg_locks, and pg_stat_statements.	RCA page /console/rca/H4
Mitigation	2026-10-27 17:31:57 UTC	On-call terminated blockers and reloaded the connection pool as emergency mitigation (section 6.1).	pg_terminate_backend + PgBouncer RELOAD
Recovery	2026-10-27 18:00:57 UTC	Latency, lock depth, and connection use returned to baseline. On-call monitored the incident for 30 minutes before closing it.	recovery verified at 18:00:57 UTC

5. Technical Root Cause Analysis

Causal Flow Diagram

```
+-----------------------------------------------------------------------------------------------+
| Trigger H4: Connection saturation above the 4x vCPU budget                                    |
+-----------------------------------------------------------------------------------------------+
                                                  |
                                                  v
+-----------------------------------------------------------------------------------------------+
| Mechanism: The application pool multiplied per replica and exceeded the cluster connection...  |
+-----------------------------------------------------------------------------------------------+
                                                  |
                                                  v
+-----------------------------------------------------------------------------------------------+
| Symptom: Saturation surfaced as 53300 too_many_connections errors and LWLock:BuffferContent... |
+-----------------------------------------------------------------------------------------------+
                                                  |
                                                  v
+-----------------------------------------------------------------------------------------------+
| Impact: orders/customers OLTP latency and write availability                                  |
+-----------------------------------------------------------------------------------------------+
                                                  v
+-----------------------------------------------------------------------------------------------+
| Blast radius: checkout writes, orders reads, p95 SLO                                          |
+-----------------------------------------------------------------------------------------------+
```

Primary Cause

The application pool multiplied per replica and exceeded the cluster connection budget. New checkouts waited on backend slots. Correlated verdict: Root Cause (H2 | High): PID 41892 idle in transaction for 312.0s holding RowExclusiveLock on public.orders [CSQL-M05 | DIRECT] -> Proximate Cascading Symptoms (2 suppressed): H11 (Lock blocking chain depth 4 queued behind idle-in-transaction PID 41892 on public.orders [CSQL-M03 | INDIRECT]), H12 (public.orders dead-tuple ratio 34.0% (340,000 dead tuples) amplified by H6 autovacuum stall [CSQL-M01 | INDIRECT]).

Secondary Cause

Application connections bypassed PgBouncer or used it in session mode. Each connection pinned a PostgreSQL backend. Downstream correlation: Root Cause (H6 | Medium): Autovacuum disabled on public.orders while hot updates accumulate dead tuples [CSQL-M01 | DIRECT] (isolated breach, 0 downstream symptoms).

Tertiary Cause

Saturation surfaced as 53300 too_many_connections errors and LWLock:BuffferContent waits that slowed every orders transaction. Correlated evidence: Check H2 (High, owner DEVELOPER): value=312.0 vs. threshold=60.0 (cited) - PID 41892 idle in transaction for 312.0s holding RowExclusiveLock on public.orders [CSQL-M05 | DIRECT]

Finding SQL

```
SELECT count(*) AS total_connections,
       count(*) FILTER (WHERE wait_event_type = 'LWLock') AS lwlock_waiters
  FROM pg_stat_activity;
```

Metrics

- Check H4: measured value 42 against threshold 85 (cited); state within threshold.
- Probe detail: 42 backends vs 4x21=84 rule of thumb (max_connections=100, 2 in LWLock wait); active_non_idle=18 client backends, active_session_ratio=18.0% of max_connections (budget 25%)
- PG 17/18 pg_stat_checkpointer: requested ratio 11.2%, write 1840.5 ms, sync 210.2 ms.
- Waiters: LWLock buffer 2, IO 1.

Causal Tag

[CSQL-M07 | DIRECT]

6. Step-by-Step Resolution & Remediation Procedures

6.1 Emergency Mitigation (Immediate Stabilization)

Step 1 - Capture the blocker set before touching anything

```
SELECT pid, usename, state,
       now() - xact_start AS xact_age,
       wait_event_type, wait_event,
       pg_blocking_pids(pid) AS blockers,
       left(query, 160) AS query
  FROM pg_stat_activity
 WHERE state <> 'idle'
   AND (now() - xact_start > interval '30 seconds'
        OR wait_event_type = 'Lock')
 ORDER BY xact_age DESC;
```

Record the head blocker for H4 in the incident timeline before any termination. The capture is read-only and pg_monitor compatible.

Step 2 - Terminate only the head blocking backend, with a bounded wait

```
-- Replace 41892 with the head blocker captured in Step 1.
SELECT pg_terminate_backend(41892, 5000);

-- Confirm the queue drains instead of re-queuing.
SELECT count(*) AS queue_depth
  FROM pg_stat_activity
 WHERE cardinality(pg_blocking_pids(pid)) > 0;
```

Do not terminate blocked sessions first. They re-queue behind the same head blocker and the incident restarts.

Step 3 - Drain and reload PgBouncer so pooled sessions pick up the new limits

```
-- PgBouncer admin console (port 6432), not the PostgreSQL port.
psql -h 127.0.0.1 -p 6432 -U pgbouncer pgbouncer -c 'SHOW POOLS;'
psql -h 127.0.0.1 -p 6432 -U pgbouncer pgbouncer -c 'RELOAD;'
psql -h 127.0.0.1 -p 6432 -U pgbouncer pgbouncer -c 'SHOW STATS;'
```

RELOAD applies pgbouncer.ini pool_mode and default_pool_size without dropping the listener. Transaction pooling discards session-level state by design.

6.2 Permanent Remediation Steps

6.2.1 CREATE INDEX CONCURRENTLY & EXPLAIN (ANALYZE, BUFFERS)

```
-- 6.2.1 Build the replacement index without blocking writes.
CREATE INDEX CONCURRENTLY IF NOT EXISTS idx_orders_customer_created
    ON public.orders (customer_id, created_at DESC);

-- Prove the planner uses it before dropping or shipping anything else.
EXPLAIN (ANALYZE, BUFFERS)
SELECT id, status, total_cents
  FROM public.orders
 WHERE customer_id = $1
 ORDER BY created_at DESC
 LIMIT 50;

-- Finding-specific DDL for H4.
-- PgBouncer: pool_mode = transaction; default_pool_size = 4 * cpu_cores
```

Cap the client pool, then size PgBouncer to the workload. Do not raise max_connections first.

6.2.2 ALTER SYSTEM SET statement_timeout = '5000ms' & idle_in_transaction_session_timeout = '15000ms'

```
-- 6.2.2 Bound every session at the cluster and role level.
ALTER SYSTEM SET statement_timeout = '5000ms';
ALTER SYSTEM SET idle_in_transaction_session_timeout = '15000ms';
SELECT pg_reload_conf();

-- Verify the effective values for the application role.
SELECT rolname, rolconfig
  FROM pg_roles
 WHERE rolname IN ('app_rw', 'migration_runner', 'analytics_ro');
```

Per-role settings override cluster defaults. For H4, the idle-in-transaction ceiling is the control that prevents recurrence.

6.2.3 Before/After @Transactional application refactor

```
# Before: network I/O and retries live inside the transaction.
@transactional
def create_order(payload: OrderIn) -> Order:
    order = Order(**payload.model_dump())
    session.add(order)
    session.flush()
    inventory.reserve(order.id)      # remote call holds the write lock
    payments.authorize(order)        # another remote call
    return order

# After: the transaction only covers database work.
@transactional
def create_order(payload: OrderIn) -> Order:
    order = Order(**payload.model_dump())
    session.add(order)
    session.flush()
    outbox.enqueue("order.created", order.id)  # local insert, no I/O
    return order

# Inventory and payment run after COMMIT, with their own short transactions.
```

Keep only database work inside the transaction boundary. Run remote calls and retries after COMMIT.

7. Preventative Action Items & Monitoring Controls

Action Item	Type	Target Date	Owner	Status
Add a burn-rate alert for H4 with a 5-minute lookback and route it to the Monitor stage dashboard.	Monitoring control	2026-10-30	DBA / Platform Engineer	Open
Enforce statement_timeout, lock_timeout, and idle_in_transaction_session_timeout for every application role.	Configuration	2026-10-30	DBA / Platform Engineer	Open
Ship the reviewed remediation for H4 behind Tier 1 static analysis and Tier 2 shadow verification.	Code change	2026-11-10	Backend Engineer	In progress
Add a regression test that fails when the finding reproduces against the showcase snapshot.	Test coverage	2026-11-06	Backend Engineer	Open
Re-audit the control against CIS PostgreSQL 17 Benchmark v1.0.0 and record the evidence in the Govern stage.	Compliance	2027-01-25	Software Architect / Technical Writer	Scheduled

8. Related Resources & References

- Internal Monitoring Dashboard: /console?stage=monitor&view=rca&finding=H4
- Related Merge Request: !0 - fix MR to be opened from section 6.2 (H4)
- Runbook Reference: src/circuitsql/skills/root-cause-analysis/SKILL.md section 4-6; docs/specs/v3-spec.md (finding H4)
- Vendor Documentation: https://www.postgresql.org/docs/17/runtime-config-connection.html