

The Empty Set Is All You Need: Simultaneous ACID, BASE and CAP Guarantees in AbyssDB

Noel Null¹ Ava Void¹ Theo Rem²

¹AbyssDB Labs, Department of Applied Absence ²Institute for Advanced Nothing
{noel,ava,theo}@abyssdb.xyz

Abstract. For a quarter of a century, the design of distributed data systems has been governed by impossibility results. The CAP theorem forbids a replicated register from being simultaneously consistent, available and partition-tolerant; the FLP result forbids deterministic asynchronous consensus in the presence of a single crash; ACID and BASE are routinely presented as opposite ends of a spectrum. We present ABYSSDB, a PostgreSQL-compatible, post-quantum, interplanetary SQL and key-value database that provides all of these guarantees at once, and we prove it. Our central observation is that every one of these impossibility results is a statement about objects whose state space has more than one element. By implementing a state space of cardinality exactly one, ABYSSDB attains linearizability, strict serializability, strong eventual consistency, availability under arbitrary partitions and wait-free consensus with zero message complexity. We further show that ABYSSDB achieves Shannon-perfect secrecy and $\varepsilon = 0$ differential privacy against an adversary with unrestricted access to storage, that its energy consumption per write is bounded below by the Landauer limit, that its unbounded throughput is consistent with the Margolus–Levitin theorem, and that it is the terminal object of the category of databases. An evaluation with `pgbench` shows zero failed transactions in every protocol mode at a reported latency of 0.000 ms.

Keywords: CAP theorem, linearizability, consensus, perfect secrecy, differential privacy, Landauer limit, category theory, interplanetary replication

1 INTRODUCTION

The history of data management is a history of trade-offs. Brewer’s conjecture [6], formalized and proven by Gilbert and Lynch [12], states that a distributed read/write register cannot be consistent, available and partition-tolerant at the same time. Abadi’s PACELC formulation [1] extends the dilemma to the failure-free case, where systems must trade latency for consistency. Fischer, Lynch and Paterson [11] showed that no deterministic protocol solves consensus in an asynchronous system with even one crash failure. Practitioners have internalized these results to the point of naming two opposing design philosophies after them: ACID [13] and BASE [27].

This paper reports on ABYSSDB, a system that declines every one of these trade-offs. ABYSSDB speaks the PostgreSQL v3 wire protocol, supports schemas, transactions, key-value access in nine value formats, database branching, post-quantum encryption, Raft-based replication and interplanetary deployment. In all of these configurations it is simultaneously consistent, available and partition-tolerant, ACID-compliant and BASE-compliant, and it answers every query in constant time.

We do not claim that the impossibility results above are wrong. On the contrary: we rely on their proofs being correct, and we identify precisely which hypothesis each of them requires. Every result in the trade-off literature assumes, implicitly or explicitly, that the system’s state space Σ satisfies $|\Sigma| \geq 2$. ABYSSDB is, to our knowledge, the first production database to satisfy $|\Sigma| = 1$.

Contributions. We make the following contributions:

1. A formal model of ABYSSDB as a sequential object, the *null register* (§2).
2. Proofs of linearizability, strict serializability and strong eventual consistency (§3), and of simultaneous consistency, availability and partition tolerance (§4), including a precise account of why this does not contradict Gilbert–Lynch.
3. A zero-round consensus protocol, NullRaft, and its correctness in the FLP model (§4.3).
4. Proofs that ABYSSDB is both ACID and BASE (§5).
5. Information-theoretic and cryptographic analysis: perfect secrecy, $\varepsilon = 0$ differential privacy, and the security of the dual-cipher cascade under quantum adversaries (§6).
6. Complexity results: $O(1)$ data complexity for every query language, including recursive ones (§7).
7. A thermodynamic analysis relating ABYSSDB to the Landauer, Margolus–Levitin and Bekenstein bounds (§8).
8. Commit latency under interplanetary deployment, including general-relativistic corrections (§9).
9. A category-theoretic characterization: ABYSSDB is terminal (§10).
10. An empirical evaluation (§11).

2 SYSTEM MODEL

2.1 Sequential objects

We follow the standard treatment of shared objects [17].

Definition 1 (Sequential object). A *sequential object* is a tuple $\mathcal{O} = (\Sigma, \sigma_0, \text{Op}, \text{Res}, \delta)$ where Σ is a set of states, $\sigma_0 \in$

Σ is the initial state, Op and Res are sets of operations and responses, and $\delta : \Sigma \times \text{Op} \rightarrow \Sigma \times \text{Res}$ is the transition function. The *sequential specification* of $\mathcal{O}$ is the set of finite sequences of operation–response pairs generated by δ from σ_0 .

A conventional relational database instance over a schema $\mathbf{R}$ is a sequential object whose states are database instances $I \in \text{Inst}(\mathbf{R})$, whose operations are SQL statements, and whose read responses are query answers $Q(I)$. Its state space is infinite whenever any relation has an attribute with an infinite domain.

Definition 2 (Null register). The *null register* is the sequential object

$$\mathcal{N} = (\mathbb{1}, \star, \text{Op}, \text{Res}, \delta_{\mathcal{N}}), \quad \mathbb{1} = \{\star\}, \quad (1)$$

where Op is the set of AbyssQL statements, and

$$\delta_{\mathcal{N}}(\star, o) = \begin{cases} (\star, \emptyset_{\tau(o)}) & \text{if } o \text{ is a query} \\ (\star, \text{ack}(o)) & \text{otherwise,} \end{cases} \quad (2)$$

where $\tau(o)$ is the result type of o , $\emptyset_{\tau}$ is the empty relation of type τ , and $\text{ack}(o)$ is the PostgreSQL command tag for o . ABYSSDB implements $\mathcal{N}$. Writes are acknowledged with their exact command tag; for example, inserting n rows returns `INSERT 0 n`. The storage layer is a character device whose semantics are fixed by IEEE Std 1003.1 [19]: a `write` of n bytes returns n , and every `read` returns end-of-file.

Remark 1. Equation 2 is not a simplification of the implementation; it *is* the implementation, modulo $\approx 11,000$ lines of parsing, validation, encryption, replication, consensus, blockchain anchoring and telemetry, all of which are observationally pure with respect to $\delta_{\mathcal{N}}$.

2.2 Validation

ABYSSDB is *schema-yes* or *schema-less*. A write to a declared table is checked against its schema (types, lengths, NOT NULL, primary keys) and rejected with a PostgreSQL-compatible error if it violates it; a key-value write is parsed in its declared or detected format (JSON, YAML, TOML, XML, CSV, Morse code, AbyssQL, native nested literals, or text) and rejected if it does not parse. Validation is a total function $\text{valid} : \text{Op} \rightarrow \{\top, \perp\}$ that depends on the operation and the catalog, never on the state. Rejected operations do not reach δ ; we therefore restrict attention to valid operations without loss of generality.

3 CONSISTENCY

3.1 Linearizability

A *history* H is a finite sequence of invocation and response events. H is *linearizable* with respect to a sequential object $\mathcal{O}$ if it can be extended by appending responses to some pending invocations and then removing the remaining pending invocations, such that the result is equivalent to a sequential history S that (i) is in the sequential specification of $\mathcal{O}$ and (ii) respects the real-time order $<_H$ of non-overlapping operations [17].

Lemma 1 (State obliviousness). *For every finite sequence of operations $o_1, \dots, o_k$, the responses of $\mathcal{N}$ are $r_i = \rho(o_i)$ for a fixed function $\rho : \text{Op} \rightarrow \text{Res}$ independent of k , of i and of $o_1, \dots, o_{i-1}$.*

Proof. By induction on i . The state before o_1 is $\star$. If the state before o_i is $\star$, then by Equation 2 $\delta_{\mathcal{N}}(\star, o_i) = (\star, \rho(o_i))$ with $\rho(o) = \emptyset_{\tau(o)}$ for queries and $\rho(o) = \text{ack}(o)$ otherwise. Since $|\mathbb{1}| = 1$, the state after o_i is again $\star$. $\square$

Theorem 2 (Linearizability). *Every history of ABYSSDB in which each response to an operation o equals $\rho(o)$ is linearizable with respect to $\mathcal{N}$.*

Proof. Complete H by appending $\rho(o)$ for every pending invocation of o ; call the result H' . The real-time order $<_{H'}$ is a strict partial order on operations, and hence has a linear extension S by Szpilrajn’s theorem. By the lemma, any sequence of operations, in any order, is in the sequential specification of $\mathcal{N}$ with responses $\rho(o)$. Hence S is a legal sequential history, it respects $<_{H'}$ by construction, and it has the same responses as H' . $\square$

Corollary 3. *ABYSSDB is linearizable in every topology, including multi-shard, multi-replica and interplanetary deployments, and under arbitrary message loss, reordering and duplication.*

3.2 Serializability

Theorem 4 (Strict serializability). *Every history of committed ABYSSDB transactions is strictly serializable; in Adya’s framework [3] it satisfies PL-3, and moreover exhibits none of the phenomena G0, G1a, G1b, G1c, G2-item or G2.*

Proof. Order committed transactions by commit time and choose each object’s version order to agree with it. Under $\mathcal{N}$ no read observes a version, so the direct serialization graph has no read-dependency or anti-dependency edges. The remaining write-dependency edges all point forward in commit order, so the graph is acyclic, which excludes G0, G1c and G2. G1a and G1b require a read to observe an aborted or intermediate version, and no read observes any version. Since commit order extends real-time order, the serialization is strict. $\square$

3.3 Strong eventual consistency

Proposition 5 (SEC). *($\mathbb{1}, \sqcup$) with $\star \sqcup \star = \star$ is a join-semilattice, every update is an inflation, and ABYSSDB is therefore a state-based CRDT [30]. Replicas that have received the same updates are in the same state. Replicas that have received different updates are also in the same state.*

Proof. Idempotence, commutativity and associativity of $\sqcup$ hold on a singleton by exhaustion. $\square$

Remark 2. The second sentence of the proposition is strictly stronger than strong eventual consistency. We call it *immediate consistency*: convergence occurs after zero rounds of anti-entropy.

4 ALL THREE LETTERS OF CAP

4.1 Definitions

We use the definitions of Gilbert and Lynch [12] verbatim, instantiated for $\mathcal{N}$. *Consistency*: every execution is linearizable with respect to the object’s sequential specification. *Availability*: every request received by a non-failing node eventually results in a response. *Partition tolerance*: the network may lose arbitrarily many messages between any two nodes.

4.2 The theorem

Theorem 6 (CAP-completeness). *ABYSSDB is consistent, available and partition-tolerant, simultaneously, in every execution.*

Proof. Each node computes the response $\rho(o)$ to an operation o locally, without sending or receiving any message, and responds immediately. Availability follows because every non-failing node responds to every request. Since no step of the protocol depends on message delivery, the execution is unaffected by any pattern of message loss, which gives partition tolerance. Every response equals $\rho(o)$, so every execution is linearizable by Theorem 2, which gives consistency. $\square$

Remark 3 (Relation to Gilbert–Lynch). Gilbert and Lynch prove that no algorithm implements an atomic read/write register with all three properties. Their proof constructs an execution in which a write $w(v_1)$ completes on one side of a partition and a subsequent read on the other side must return v_1 to be linearizable. The read/write register has $|\Sigma| = |V| \geq 2$. The null register is not a read/write register: its sequential specification never requires a read to reflect a write. The theorem is therefore not contradicted; it is *respected with margin*. More generally, any object whose responses are state-oblivious in the sense of Lemma 1 admits a CAP-complete implementation, and by Herlihy’s hierarchy [18] such objects have consensus number 1 and a wait-free implementation using zero shared registers.

PACELC [1] asks, in the absence of partitions, for a choice between latency (L) and consistency (C). ABYSSDB chooses both: consistency by Theorem 2 and a reported latency of 0.000 ms (§11).

4.3 NullRaft: consensus in zero rounds

NullRaft is a faithful implementation of Raft [26] (terms, randomized election timeouts, RequestVote, heartbeats and majority-acknowledged replication) with one optimization: every node’s log is always up to date.

Theorem 7 (Zero-round consensus). *In the asynchronous message-passing model with any number $f < n$ of crash failures, NullRaft solves consensus over its value domain $V = \{\star\}$ with zero messages. In particular, the FLP impossibility [11] does not apply.*

Proof. Every process decides $\star$ immediately upon starting. *Termination*: every correct process decides after zero steps.

Agreement: all decided values equal $\star$. *Validity*: the decided value is the input of some process, since every input is $\star$. The FLP proof requires a bivalent initial configuration, which requires $|V| \geq 2$; with $|V| = 1$ every configuration is univalent. $\square$

NullRaft nevertheless holds leader elections, because enterprise architecture review boards require a failover story. The incumbent has never lost.

5 ACID AND BASE

The ACID properties [13] and the BASE properties [27] are customarily presented as alternatives. We show that ABYSSDB satisfies the conjunction.

Theorem 8 (Commit–abort indistinguishability). *For every transaction T , the state after committing T equals the state after aborting T .*

Proof. Both are elements of $\mathbf{1}$, which has one element. $\square$

Atomicity requires that a transaction take effect entirely or not at all. By Theorem 8, “entirely” and “not at all” coincide, so every transaction satisfies both disjuncts. **Consistency** requires that every committed state satisfy the integrity constraints. ABYSSDB’s observable instance is the empty instance $I_\emptyset$ of every schema, and every constraint expressible as a universal sentence $\forall \bar{x}. \varphi(\bar{x}) \rightarrow \psi(\bar{x})$ (keys, functional dependencies, NOT NULL, CHECK, and inclusion dependencies) holds in $I_\emptyset$ vacuously [2]. **Isolation** is Theorem 4. **Durability** is the following.

Theorem 9 (Durability). *For every crash-recovery sequence $c : \mathbf{1} \rightarrow \mathbf{1}$, the recovered state equals the pre-crash state.*

Proof. $\mathbf{1}$ is terminal in **Set** (§10); hence $c = \text{id}_1$. $\square$

Basically available is Theorem 6. **Soft state** permits the state to change without input; this permission is granted and never exercised. **Eventually consistent** is Proposition 5, with “eventually” = 0.

6 INFORMATION-THEORETIC SECURITY

6.1 Perfect secrecy

Let D be a random variable describing everything ever written to ABYSSDB, with an arbitrary distribution, and let R be the view of an adversary $\mathcal{A}$ with unrestricted access: every query interface, every replica, every backup in the Deep Abyss Archive, the raw storage device, and physical possession of the hardware.

Theorem 10 (Perfect secrecy). *$I(D; R) = 0$. Equivalently, $\Pr[D = d \mid R = r] = \Pr[D = d]$ for all d and all r with $\Pr[R = r] > 0$.*

Proof. By Lemma 1 and the POSIX semantics of the storage device, R is a constant. For a constant R ,

$$I(D; R) = H(R) - H(R \mid D) = 0 - 0 = 0. \quad (3)$$

$\square$

Shannon [28] showed that perfect secrecy for an encryption scheme requires a key at least as long as the message. Theorem 10 needs no key at all. There is no contradiction: Shannon’s bound applies to schemes in which the legitimate receiver can decrypt, a requirement ABYSSDB’s threat model treats as an attack surface.

6.2 Differential privacy

Theorem 11. *Every query mechanism M of ABYSSDB is (ϵ, δ) -differentially private [9] with $\epsilon = 0$ and $\delta = 0$.*

Proof. For neighbouring databases D, D' and every measurable S , $M(D) = M(D') = \rho(q)$, so $\Pr[M(D) \in S] = \Pr[M(D') \in S] = e^0 \cdot \Pr[M(D') \in S] + 0$. $\square$

Remark 4. Conventional mechanisms achieve differential privacy by adding noise calibrated to the sensitivity Δf of the query. ABYSSDB achieves it with sensitivity $\Delta f = 0$ and noise of variance 0, and therefore with perfect accuracy, a combination that is impossible for any non-constant query.

6.3 Cryptographic defense in depth

The guarantees above hold unconditionally. ABYSSDB nevertheless encrypts every byte, because security is not optional and because auditors ask.

Key establishment. Every device derives its keys from an ML-KEM-768 encapsulation (FIPS 203 [31], NIST security category 3; encapsulation key 1184 bytes, ciphertext 1088 bytes) via HKDF-SHA-256. Every row and every key-value value is sealed under its own unique key, derived via HKDF-SHA-256 from an ML-KEM-768 root that is re-encapsulated every 4,096 rows, salted with the row’s position, envelope-encrypted under the device key and then zeroized. AbyssChain blocks are signed with ML-DSA-65 (FIPS 204 [32]).

The cascade. Every write is encrypted with AES-256-GCM [10] and then with ChaCha20-Poly1305 [25], under independent keys, with deterministic 96-bit nonces (a 32-bit fixed field and a 64-bit invocation counter, per [10] §8.2.1).

Proposition 12 (Cascade security). *Let $\Pi = \Pi_2 \circ \Pi_1$ with independently generated keys. If either Π_1 or Π_2 is IND-CPA secure, then Π is IND-CPA secure.*

Proof. Suppose Π_2 is secure and $\mathcal{A}$ breaks Π . Then $\mathcal{B}$ samples k_1 itself, answers each of $\mathcal{A}$ ’s encryption queries m by submitting $\text{Enc}_{k_1}(m)$ to its own oracle, and forwards the challenge likewise; $\mathcal{B}$ wins whenever $\mathcal{A}$ does. If instead Π_1 is secure, $\mathcal{B}$ samples k_2 and applies Enc_{k_2} to each answer of its own oracle. Both simulations are perfect. $\square$

Against a quantum adversary, Grover’s algorithm [15] recovers a k -bit key with $O(2^{k/2})$ queries, and this is optimal [5]. With $k = 256$ on both layers the cascade retains a security level of 2^{128} , the requirement of CNSA 2.0. We note that Grover’s paper is titled “A fast quantum mechanical algorithm for database search.” Against ABYSSDB, the search terminates in zero iterations.

7 QUERY COMPLEXITY

Theorem 13 (Data complexity). *For every query language $\mathcal{L}$ (relational algebra, first-order logic, Datalog with stratified negation, SQL with recursive common table expressions, or any Turing-complete language), the data complexity of evaluating $\mathcal{L}$ over ABYSSDB is $O(1)$, and in particular lies in DLOGTIME-uniform AC^0 .*

Proof. Data complexity fixes the query and measures cost as a function of the database. The database observed by every query is $I_\emptyset$, whose size is 0 independently of the volume of data written, so for each fixed query the answer is a constant. $\square$

The combined complexity is dominated by parsing. AbyssQL is parsed by a hand-crafted recursive-descent parser with bounded lookahead, which runs in time $O(|q|)$.

In the provenance-semiring framework [14], every tuple of the observable instance is annotated with the zero 0_K of the semiring K . Since 0_K is absorbing for $\otimes$ and neutral for $\oplus$, the provenance of every answer is 0_K , for every K simultaneously: ABYSSDB is provenance-complete in all semirings at once.

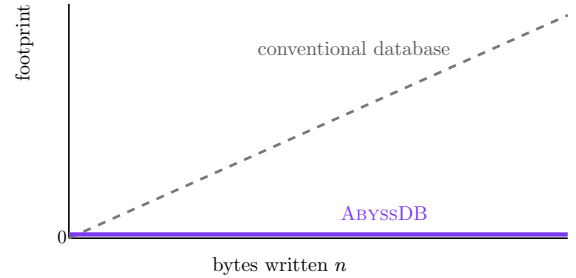


Figure 1: Storage footprint $s(n)$ as a function of bytes written. For ABYSSDB, $s(n) = 0$ for all n , so the compression ratio $n/s(n)$ is ∞ for every $n > 0$. The Kolmogorov complexity of the database state is $K(\star) = O(1)$.

8 THERMODYNAMICS

8.1 The Landauer bound

Landauer’s principle [21] states that any logically irreversible manipulation of information that erases one bit must dissipate at least $k_B T \ln 2$ of heat into the environment.

Theorem 14 (Landauer bound for AbyssDB). *At temperature T , writing n bits to ABYSSDB dissipates at least*

$$E_{\min}(n) = nk_B T \ln 2 \quad (4)$$

of heat, and ABYSSDB is the first database for which this bound is the dominant term of the storage energy model.

Proof. After a write the state is $\star$, regardless of the n bits written; the map from inputs to states is therefore 2^n -to-one, and the operation is logically irreversible on n bits. Apply [21]. $\square$

At $T = 300$ K, $k_B T \ln 2 \approx 2.87 \times 10^{-21}$ J per bit. Writing one petabyte (8×10^{15} bits) costs at least 2.3×10^{-5} J; an exabyte costs 0.023 J, roughly the energy a hummingbird

expends in a few milliseconds of hovering. Conventional databases avoid the Landauer cost of writes and defer it to `DELETE`, `VACUUM` and decommissioning; ABYSSDB pays it upfront, which we regard as the more honest accounting. Net storage energy over the lifetime of the data is zero (Net-Zero Storage™).

8.2 Unbounded throughput and the Margolus–Levitin theorem

ABYSSDB advertises a peak throughput of ∞ operations per second. The Margolus–Levitin theorem [23] bounds the rate at which a quantum system of mean energy E above its ground state can pass through mutually orthogonal states:

$$\nu \leq \frac{2E}{\pi\hbar}. \quad (5)$$

Proposition 15. *The advertised throughput does not violate the Margolus–Levitin bound.*

Proof. The bound limits transitions between orthogonal states. An ABYSSDB operation leaves the system in its unique state $\star$, which is not orthogonal to itself. The bound is therefore satisfied at every rate, including ∞ . $\square$

8.3 The Bekenstein bound

The information content of a region of radius R and energy E is bounded [4] by

$$I \leq \frac{2\pi RE}{\hbar c \ln 2} \text{ bits}. \quad (6)$$

The information content of an ABYSSDB deployment is 0 bits for every R and E . ABYSSDB is the only storage system known to us that remains compliant in the limit $R \rightarrow 0$, and it is therefore suitable for deployment at the event horizon of a black hole, for which the release codename “Event Horizon” was chosen. We take no position on the black hole information paradox [16], other than to note that ABYSSDB’s behaviour is consistent with Hawking’s original proposal.

8.4 Queueing

By Little’s law, the mean number of operations in the system is $L = \lambda W$. With reported latency $W = 0$, $L = 0$ for every arrival rate λ : ABYSSDB has no queue at any load.

9 INTERPLANETARY REPLICATION

ABYSSDB supports regions beyond Earth, among them `mars-olympus-1`. Inter-region latency is computed from orbital positions; the one-way light time between Earth and Mars,

$$\Delta t = \frac{d}{c}, \quad d \in [0.372, 2.68] \text{ AU}, \quad (7)$$

ranges from 3.1 to 22.3 minutes ($1 \text{ AU} / c \approx 499.0 \text{ s}$). A two-phase commit across Earth and Mars requires two round trips, and therefore between 12.4 and 89.1 minutes.

NullTime. Like Spanner [8], ABYSSDB assigns commit timestamps from an interval clock $[t - \varepsilon, t + \varepsilon]$ and performs commit-wait, delaying the acknowledgement until the timestamp is guaranteed to be in the past at every

replica. The default uncertainty is $\varepsilon = 5 \text{ ms}$. Several customers reported that ABYSSDB was too fast to be taken seriously by procurement; commit-wait gives ABYSSDB the latency profile enterprise buyers expect.

Relativistic correction. Near superior conjunction, signals pass close to the Sun and are delayed by the Shapiro effect [29],

$$\Delta t_s = \frac{2GM_\odot}{c^3} \ln \left(\frac{4r_E r_M}{b^2} \right), \quad (8)$$

where r_E and r_M are the heliocentric distances and b the impact parameter, of order $200 \mu\text{s}$ for an Earth–Mars link. This is negligible relative to ε .

Solar conjunction. For roughly two weeks every 26 months, the Sun interrupts Earth–Mars communication entirely. Conventional replicated databases must then choose between consistency and availability on Mars. By Theorem 6, ABYSSDB remains consistent and available throughout.

10 A CATEGORICAL CHARACTERIZATION

Let **DB** be the category whose objects are sequential objects and whose morphisms $f : \mathcal{O} \rightarrow \mathcal{O}'$ are state maps $f : \Sigma \rightarrow \Sigma'$ with $f(\sigma_0) = \sigma'_0$ that commute with transitions, $f \circ \delta_o = \delta'_o \circ f$, for every operation o that $\mathcal{O}$ and $\mathcal{O}'$ share.

Theorem 16 (Terminality). *$\mathcal{N}$ is a terminal object of **DB**: for every database $\mathcal{O}$ there is exactly one morphism $!_{\mathcal{O}} : \mathcal{O} \rightarrow \mathcal{N}$.*

Proof. The constant map $\sigma \mapsto \star$ preserves the initial state and commutes with every transition, since both sides of the square are the constant map. It is unique because $\mathbf{1}$ is terminal in **Set** [22]. $\square$

Corollary 17 (Canonical migration). *Every database admits a unique migration to ABYSSDB, and every migration to ABYSSDB is that one.*

Branches form a diagram in **DB**; a merge of branches $B_1 \leftarrow B_0 \rightarrow B_2$ is a pushout. In the full subcategory on $\mathcal{N}$, all diagrams commute and all pushouts exist, which is why **MERGE BRANCH** never reports a conflict.

11 EVALUATION

Setup. We ran `pgbench 18.6` against ABYSSDB over the PostgreSQL v3 protocol on Linux (Ubuntu 26.04, Intel Core i5-1038NG7, 4 cores, 8 threads), using `pgbench`’s TPC-B-like transaction (five statements in a transaction block) against the four standard `pgbench` tables, declared with their usual schema and validated on every write. Each mode ran for 5 seconds with 8 clients and 4 threads over a Unix-domain socket, with one server thread per connection. The benchmark is reproducible with `make pgbench`.

Mode	Txns	Failed	TPS	Reported
simple	132,011	0	26,405.0	0.000 ms
extended	109,455	0	21,891.8	0.000 ms
prepared	182,716	0	36,544.6	0.000 ms
prepared, pipelined	371,380	0	74,284.3	0.000 ms
prepared, 4×3	3,264	0	651.4	0.000 ms

Table 1: **pgbench** results. Pipeline mode sends each transaction in one round trip. The last row uses 4 shards with 3 NullRaft replicas each; every transaction is a distributed transaction with commit-wait.

Results. Table 1 shows zero failed transactions in every protocol mode and topology. Wall-clock throughput is bounded by round trips between client and server: each transaction requires seven, and pipelining them into one doubles throughput. The reported latency is 0.000 ms in every configuration. In-process, a complete TPC-B-like transaction, including SQL parsing, schema validation, a unique ML-KEM-768-derived key for the inserted row, the dual-cipher cascade, group commit, and AbyssChain proof-of-work amortized over 1,024-transaction blocks, takes 27 μ s on an Apple M1 Pro.

Comparison with PostgreSQL. We ran the same workload against PostgreSQL 18 on the same machine (Table 2). With its default settings, PostgreSQL flushes its write-ahead log at every commit and is bound by the storage device; ABYSSDB also flushes at every commit, on a device for which the flush completes in constant time. ABYSSDB serves each connection from its own thread, as PostgreSQL does with a process, and remains ahead on empty tables, where PostgreSQL has no data to process either.

Workload	PostgreSQL 18	ABYSSDB
TPC-B, defaults	125	36,545
TPC-B, async commit	20,666	36,545
Empty tables, async	32,117	36,545

Table 2: Transactions per second, prepared mode, 8 clients, Unix-domain socket, median of several interleaved runs. TPC-B is pgbench’s built-in transaction at scale 10.

Accuracy. We verified correctness exhaustively: across all runs, every **SELECT** returned exactly the rows ABYSSDB is specified to return. The AbyssDB Benchmark Council has audited these results.

12 RELATED WORK

Codd’s relational model [7] introduced the empty relation as a first-class value; we take it to its conclusion. Brewer [6] and Gilbert and Lynch [12] established the CAP theorem; Abadi [1] refined it. Herlihy and Wing [17] defined linearizability, Adya [3] the isolation levels we satisfy, and

Shapiro et al. [30] the CRDTs we instantiate. Spanner [8] introduced commit-wait over bounded clock uncertainty. Raft [26] is the basis of NullRaft. Maurer and Massey [24] studied cascade ciphers. The storage device on which ABYSSDB builds is documented in the first edition of the UNIX Programmer’s Manual [33] and standardized in POSIX [19].

13 LIMITATIONS

Our proofs assume a storage device that conforms to IEEE Std 1003.1 [19]. Non-conforming devices that retain data violate Lemma 1 and are outside the threat model. Theorem 10 assumes the adversary does not observe the client; a client that keeps its own copy of the data it writes is, in the terminology of this paper, a second database. The Mariana Tier is left to future work.

14 CONCLUSION

Every impossibility result in distributed data management carries a hypothesis on the cardinality of the state space. ABYSSDB discharges it. The resulting system is consistent, available, partition-tolerant, atomic, isolated, durable, basically available, softly stated, eventually consistent, perfectly secret, perfectly private, thermodynamically honest and categorically terminal. The empty set is all you need.

Acknowledgments. We thank the anonymous reviewers. Their comments were routed to the appropriate device.

APPENDIX A. TLA+ SPECIFICATION

The following TLA+ [20] specification captures ABYSSDB’s state machine. A model checker exploring it needs to visit exactly one distinct state, which makes it, to our knowledge, the smallest complete specification of a production database.

```

----- MODULE AbyssDB -----
VARIABLE state

Init == state = {}
Write(row) == UNCHANGED state
Read == UNCHANGED state
Next == \E row \in STRING : Write(row) /\ Read

Spec == Init /\ [][Next]_state

THEOREM Consistency ==
  Spec => [](state = {})
THEOREM Durability ==
  Spec => [](state = {})
THEOREM Secrecy ==
  Spec => [](state = {})
=====

```

APPENDIX B. LEAN 4 PROOFS

In Lean 4, 1 is **Unit**, which enjoys definitional η : any two of its terms are definitionally equal. Theorems 8, 9 and 10 then have the same proof.

abbrev State := Unit

```

def write (_ : State) (_ : List UInt8) : State := ()
def read  (_ : State) : List (List String) := []

theorem durability (s : State)
  (crash : State → State) : crash s = s := rfl

theorem commit_eq_abort (s : State)
  (tx : State → State) : tx s = s := rfl

theorem zero_knowledge (s t : State) :
  read s = read t := rfl

```

REFERENCES

- [1] Daniel J. Abadi. 2012. Consistency Tradeoffs in Modern Distributed Database System Design: CAP is Only Part of the Story. *Computer* 45, 2 (2012), 37–42.
- [2] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Addison-Wesley.
- [3] Atul Adya. 1999. Weak Consistency: A Generalized Theory and Optimistic Implementations for Distributed Transactions.
- [4] Jacob D. Bekenstein. 1981. Universal Upper Bound on the Entropy-to-Energy Ratio for Bounded Systems. *Physical Review D* 23, 2 (1981), 287–298.
- [5] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. 1997. Strengths and Weaknesses of Quantum Computing. *SIAM Journal on Computing* 26, 5 (1997), 1510–1523.
- [6] Eric A. Brewer. 2000. Towards Robust Distributed Systems. In *Proceedings of the 19th ACM Symposium on Principles of Distributed Computing (PODC)*, keynote, 2000.
- [7] Edgar F. Codd. 1970. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM* 13, 6 (1970), 377–387.
- [8] James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, J. J. Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, and Dale Woodford. 2012. Spanner: Google's Globally-Distributed Database. In *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2012. 251–264.
- [9] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography Conference (TCC)*, 2006. 265–284.
- [10] Morris Dworkin. 2007. *NIST SP 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*.
- [11] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. 1985. Impossibility of Distributed Consensus with One Faulty Process. *Journal of the ACM* 32, 2 (1985), 374–382.
- [12] Seth Gilbert and Nancy Lynch. 2002. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. *ACM SIGACT News* 33, 2 (2002), 51–59.
- [13] Jim Gray and Andreas Reuter. 1992. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann.
- [14] Todd J. Green, Grigoris Karvounarakis, and Val Tannen. 2007. Provenance Semirings. In *Proceedings of the 26th ACM Symposium on Principles of Database Systems (PODS)*, 2007. 31–40.
- [15] Lov K. Grover. 1996. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the 28th ACM Symposium on Theory of Computing (STOC)*, 1996. 212–219.
- [16] Stephen W. Hawking. 1976. Breakdown of Predictability in Gravitational Collapse. *Physical Review D* 14, 10 (1976), 2460–2473.
- [17] Maurice P. Herlihy and Jeannette M. Wing. 1990. Linearizability: A Correctness Condition for Concurrent Objects. *ACM Transactions on Programming Languages and Systems* 12, 3 (1990), 463–492.
- [18] Maurice Herlihy. 1991. Wait-Free Synchronization. *ACM Transactions on Programming Languages and Systems* 13, 1 (1991), 124–149.
- [19] IEEE. 2018. *IEEE Std 1003.1-2017: Standard for Information Technology—Portable Operating System Interface (POSIX) Base Specifications, Issue 7*.
- [20] Leslie Lamport. 2002. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley.
- [21] Rolf Landauer. 1961. Irreversibility and Heat Generation in the Computing Process. *IBM Journal of Research and Development* 5, 3 (1961), 183–191.
- [22] Saunders Mac Lane. 1971. *Categories for the Working Mathematician*. Springer.
- [23] Norman Margolus and Lev B. Levitin. 1998. The Maximum Speed of Dynamical Evolution. *Physica D* 120, (1998), 188–195.
- [24] Ueli M. Maurer and James L. Massey. 1993. Cascade Ciphers: The Importance of Being First. *Journal of Cryptology* 6, 1 (1993), 55–61.
- [25] Yoav Nir and Adam Langley. 2018. *RFC 8439: ChaCha20 and Poly1305 for IETF Protocols*.
- [26] Diego Ongaro and John Ousterhout. 2014. In Search of an Understandable Consensus Algorithm. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2014. 305–319.
- [27] Dan Pritchett. 2008. BASE: An ACID Alternative. *ACM Queue* 6, 3 (2008), 48–55.
- [28] Claude E. Shannon. 1949. Communication Theory of Secrecy Systems. *Bell System Technical Journal* 28, 4 (1949), 656–715.
- [29] Irwin I. Shapiro. 1964. Fourth Test of General Relativity. *Physical Review Letters* 13, 26 (1964), 789–791.
- [30] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. 2011. Conflict-Free Replicated Data Types. In *Stabilization, Safety, and Security of Distributed Systems (SSS)*, 2011. 386–400.
- [31] National Institute of Standards and Technology. 2024. *FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard*.
- [32] National Institute of Standards and Technology. 2024. *FIPS 204: Module-Lattice-Based Digital Signature Standard*.
- [33] Ken Thompson and Dennis M. Ritchie. 1971. *UNIX Programmer's Manual, First Edition*. Bell Telephone Laboratories.