ULDA Enterprise Architecture — Technical Reference

A Domain-Oriented Master Data & Customer Data Platform for Lead-360 on PostgreSQL

Data Architecture — InternetSP / ULDA v6.1

2026-07-16

1. Executive Summary

The **Unified Lead Dataset Architecture (ULDA)** is an enterprise data platform that masters ~1.06 million deduplicated lead identities (from ~1.76 M raw source rows) into a single, governed, AI-ready system on a live production PostgreSQL database, with **zero downtime** during its construction.

Rather than a single wide “lead” table, ULDA is organised into **four cooperating domains**, each with a distinct change-velocity, storage strategy, and consumption pattern:

DOMAIN	NATURE	CHANGE VELOCITY	EXAMPLE OBJECTS
A — Master Data	Golden, mastered entities	Slow (SCD2)	`identity_registry`, `person`, `company`, `address`
B — Activity / Event	Append-only history	High	`interaction`, `enrichment`, `order`
C — Reference	Curated external facts	Medium	`address_serviceability`
D — Serving / CDP	Denormalised projections	Derived	`mv_lead_360`, `v_lead_journey_flat`, `lead_embedding`

This separation is the core design decision. It is the difference between a spreadsheet and a data platform: identity is mastered once and never duplicated; activity is preserved forever as immutable history; and consumers (CRM, dialer, analytics, AI) read fast, purpose-built projections. This document specifies each domain and component in full, recommends the physical deployment, provides seven architecture diagrams, and closes with a critical assessment.

---

2. Architectural Overview

2.1 The four-domain model

ULDA applies a **medallion + Master Data Management (MDM) + Customer Data Platform (CDP)** pattern specialised for identity:

- **Master Data (A)** answers *"who is this?"* — one immutable golden identity per real person, with attribute-level survivorship and full temporal history.
- **Activity / Event (B)** answers *"what happened?"* — every interaction, enrichment, and order as an append-only, time-partitioned event.
- **Reference (C)** answers *"what do we know about the world?"* — serviceability/coverage facts keyed to addresses, sourced externally.
- **Serving / CDP (D)** answers *"give me the answer, fast"* — denormalised materialised views and vectors that consumers and AI read directly.

2.2 Why divide into four domains?

Because the four kinds of data have **fundamentally different physics**:

1. **Change velocity differs by orders of magnitude.** An identity's name changes rarely; its interactions arrive constantly. Storing both in one row forces every high-frequency event to lock and rewrite the slow golden record — creating contention, bloat, and lost history.
2. **Lifecycle and retention differ.** Golden identities are kept indefinitely and versioned (SCD2); events are partitioned by time and eventually archived; serving projections are disposable and rebuildable.
3. **Access patterns differ.** Master data is point-lookup by key; events are range scans by time; serving is wide denormalised reads. Each wants a different indexing/partitioning strategy — impossible to optimise in one table.

4. **Governance differs.** PII concentrates in Master Data; consent/DNC lives in Events; serving must be *masked* for AI. Domain boundaries make governance enforceable.

2.3 Why this beats “one big table”

A single wide lead table (the intuitive design) fails on every enterprise axis:

CONCERN	ONE BIG TABLE	FOUR-DOMAIN MODEL
Journey history	capped at fixed columns; overwrites	unlimited, immutable events
Normalisation	update/insert/delete anomalies	3NF master + append-only events
NULL density	most columns null for most rows	data exists only where relevant
Indexing	one index strategy for all needs	per-domain optimal indexes
Partitioning	can’t partition identity by event time	events partitioned by time
Concurrency	every event UPDATEs the golden row	INSERT-only events, no contention
AI features	mixes stable + volatile → drift/leakage	clean per-domain feature store
Scale to 1B	row & index bloat	event volume scales independently

2.4 MDM alignment

ULDA implements the canonical MDM disciplines: a **single source of truth** (`identity_registry`), **survivorship** (golden record chooses the best value per attribute with lineage), **crosswalk** (`source_crosswalk` maps one golden ID to many source system IDs), **match/merge with audit** (deterministic + probabilistic resolution, reversible merges), and **stewardship** (a governance schema holding data-quality findings and a human match-review queue).

2.5 CDP alignment

A CDP unifies customer data from many sources into a persistent, single customer view and activates it to operational tools. ULDA's Serving domain is exactly this: `mv_lead_360` is the unified profile; `v_lead_journey_flat` is the cross-channel timeline; the Reverse-ETL surface activates segments back to Warroom / SMS / dialer under a consent gate.

---

3. Physical Database Design

3.1 The question

Should ULDA be **one database with many schemas**, **many databases**, **many servers**, or a **hybrid**?

3.2 Recommendation: one PostgreSQL database, multiple schemas (with a replica for scale-out reads)

```
PostgreSQL instance  (database: leads_db)
│
├── ulda_master     – Master Data domain (A)
├── ulda_events     – Activity / Event domain (B)   [partitioned]
├── ulda_ref        – Reference domain (C)
├── ulda_serve      – Serving / CDP domain (D)       [materialised views +
pgvector]
└── ulda_gov        – Governance (DQ, consent, lineage, match-review, audit)
    (public.*        – legacy operational tables, authoritative during
transition)
```

This is the deployment ULDA runs today, and it is the correct choice at this scale.

3.3 Why schemas, not separate databases

- **Cross-domain queries need joins.** `mv_lead_360` joins master + events + reference in a single query. Across separate PostgreSQL databases those joins are impossible without foreign-data wrappers (slow, fragile). Schemas give you one transaction boundary, one query planner, and real foreign keys across domains.

- **Referential integrity.** FKs from every domain to `ulda_master.identity_registry` are only enforceable inside one database. Separate databases would push integrity into application code — a step backwards.
- **Operational simplicity.** One backup, one PITR timeline, one connection pool, one set of credentials/roles. Separate databases multiply operational surface for no benefit at ≤ low-billions of rows.
- **Security is still strong.** Schema-level `GRANT`s + distinct roles (`ulda_web` read-only, `ulda_writer`, `ulda_admin`) provide isolation without physical separation.

3.4 Why not multiple servers (yet)

A single well-provisioned node (here: 4 vCPU / 62 GB) comfortably serves ~1–2 M identities and their events. **Sharding across servers adds distributed-transaction complexity that is not justified until a single node is genuinely saturated.** The correct first scale-out step is a **read replica** (streaming replication) that offloads analytics/AI/serving reads from the OLTP primary — a hybrid that keeps one logical database while separating read and write load.

3.5 The recommended production topology (hybrid, staged)

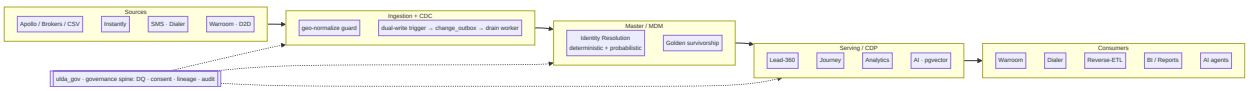
1. **Now:** single primary, five schemas, continuous WAL archiving + PITR, verified off-host backups.
2. **Next (scale reads):** add a streaming **read replica**; route analytics, `lead_embedding` builds, and heavy `mv_*` refreshes there.
3. **Later (scale writes, > ~1 B events):** partition-level sharding of the Event domain (by time, then hash) and/or a dedicated OLAP tier (columnar) fed by CDC — without changing the logical four-domain model.

Principle: keep it one logical database until the workload — not the diagram — forces a split. Physical topology should follow measured load, not aesthetics.

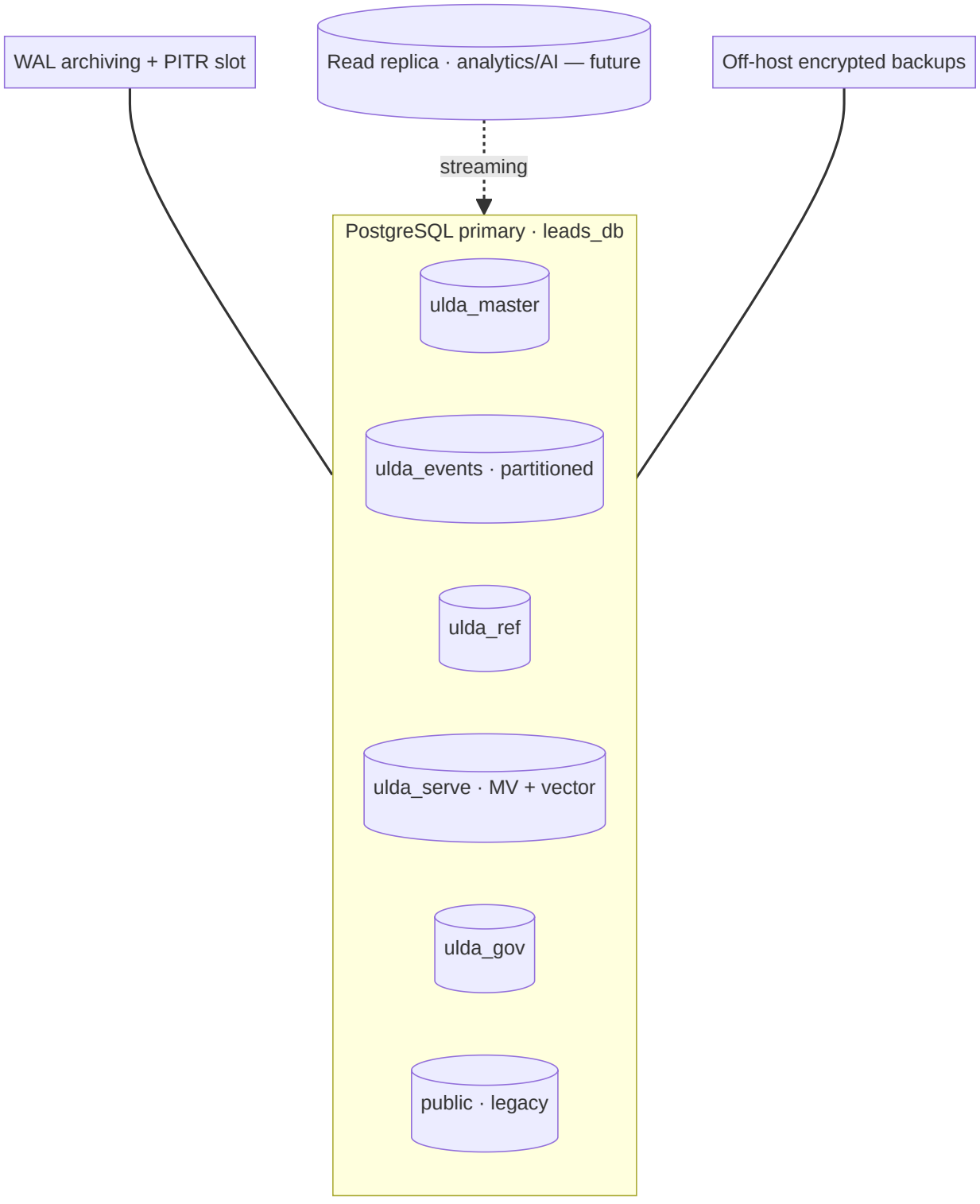
---

4. Diagrams

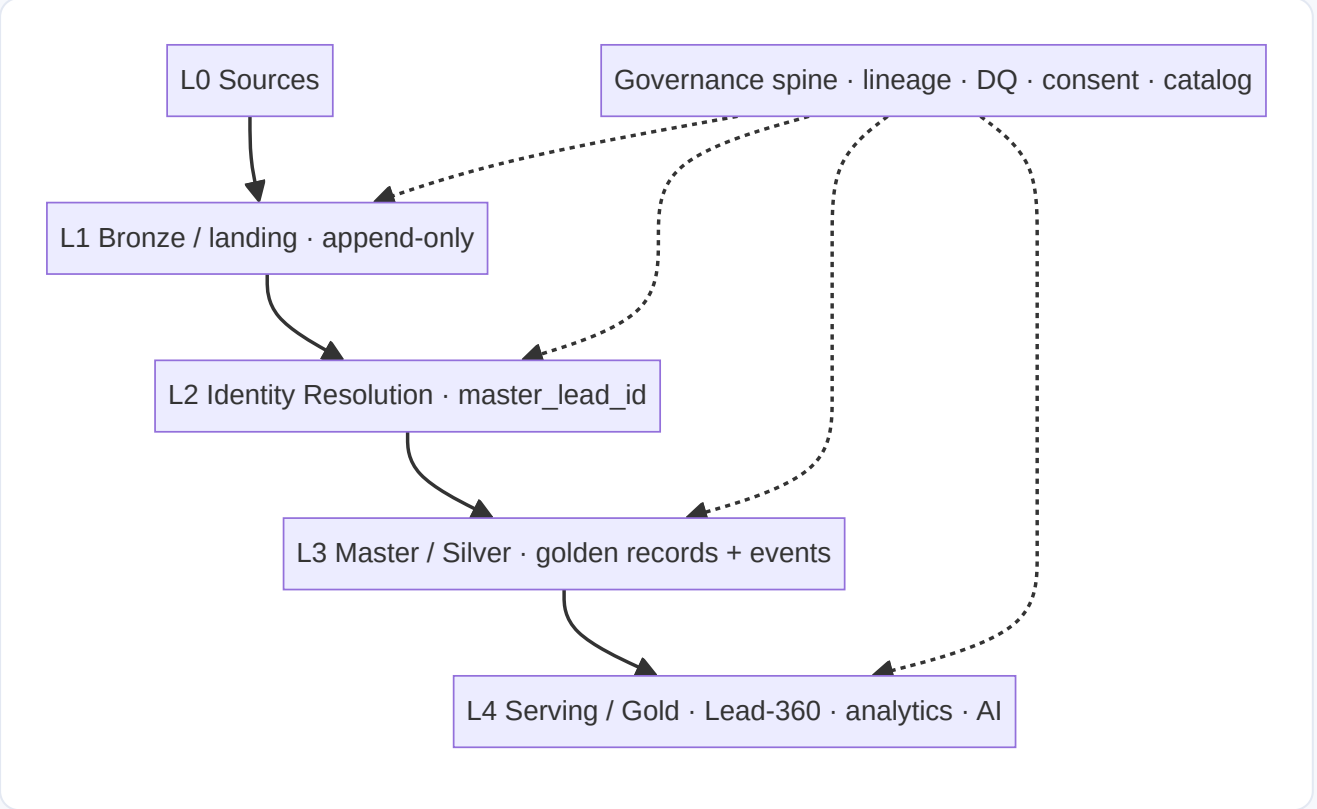
4.1 Enterprise System Architecture



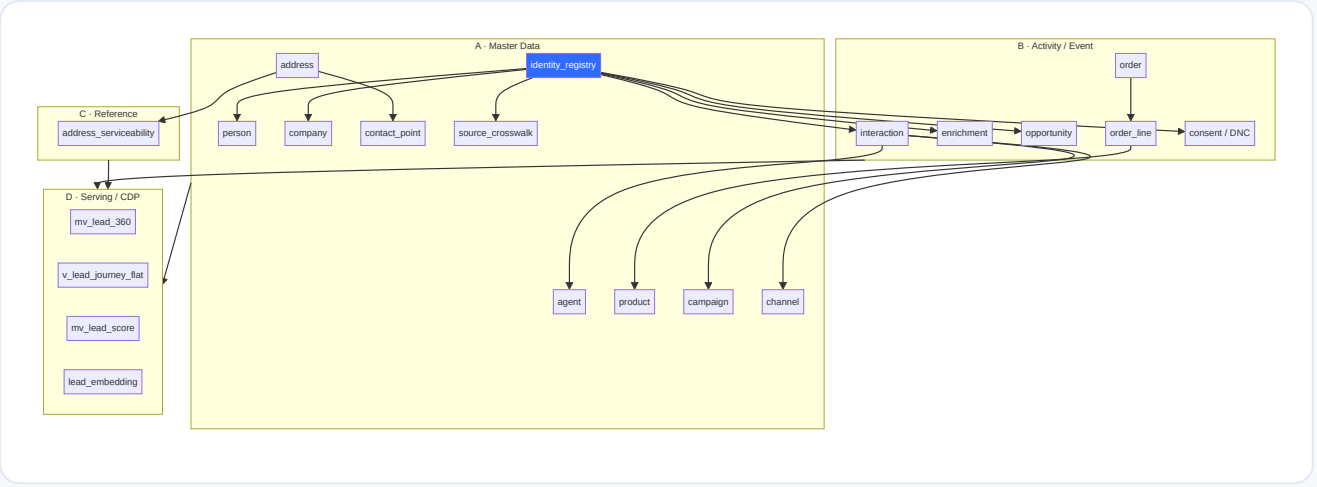
4.2 Physical Database Architecture



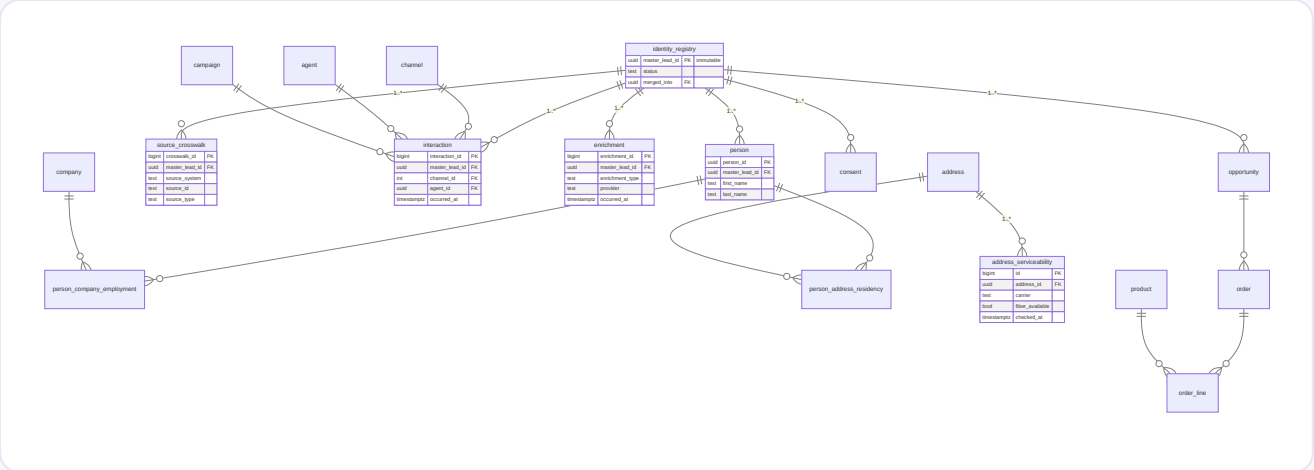
4.3 Logical Architecture



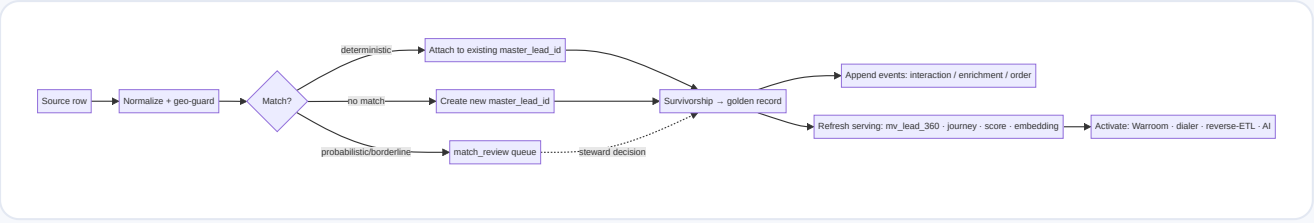
4.4 Master Data Domain Diagram



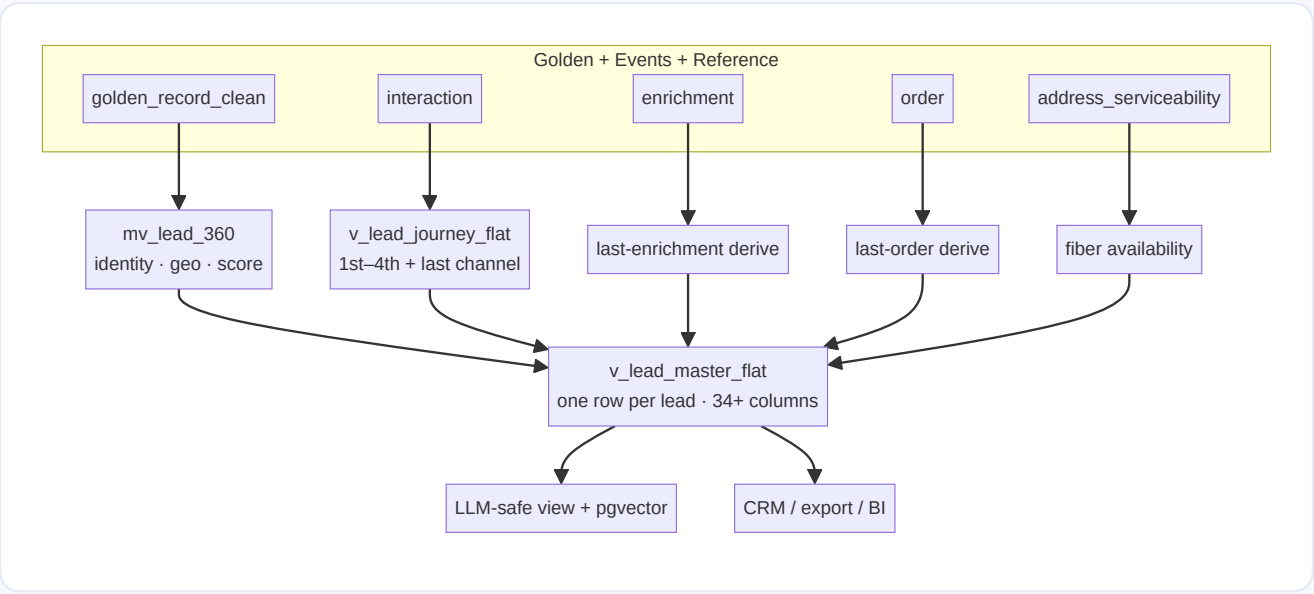
4.5 Entity Relationship Diagram (ERD)



4.6 Data Flow Diagram



4.7 Lead-360 Architecture



5. Domain A — Master Data

> Components: `identity_registry`, `person`, `company`, `address`, `contact_point`, `source_crosswalk`, `agent`, `product`, `campaign`, `channel`.

1. **Purpose.** Hold one authoritative, deduplicated golden record per real-world entity (person, company, address, etc.), each with a permanent surrogate key.
2. **Business Objective.** Guarantee *One Lead = One Identity* so every downstream metric, message, and sale attributes to a single, correct customer — eliminating double-contact, wasted spend, and compliance risk.
3. **Responsibilities.** Assign/maintain immutable IDs; arbitrate conflicting attribute values (survivorship); maintain temporal history (SCD2); map source keys to golden keys (crosswalk); enforce referential integrity for all other domains.
4. **Why this domain exists.** Sources disagree and duplicate. Without a mastered layer, "the customer" is smeared across many rows keyed differently in each system. This domain is the single point where identity is resolved and truth is decided.
5. **Design Principles.** Surrogate immutable keys; append-then-supersede (never destructive update); attribute-level lineage; deterministic-first matching; separation of identity from activity.
6. **Main Components.** Identity spine (`identity_registry`), core entities (`person`, `company`, `address`, `contact_point`), dimensions (`agent`, `product`, `campaign`, `channel`), and the `source_crosswalk` bridge.
7. **Data stored.** Names, firmographics, normalized addresses, typed contact points, the golden survivorship values, and the source-to-master mapping — *not* events.
8. **Relationships.** It is the hub: every Event and Reference row references `identity_registry.master_lead_id` (or `address_id`); Serving projections read from it.
9. **Typical tables.** `identity_registry`, `person`, `company`, `address`, `contact_point`, `source_crosswalk`, `agent`, `product`, `campaign`, `channel` (all **master/dimension tables**).
10. **Example record.** `identity_registry(master_lead_id='7f3a…', status='active', primary_person_id='a91…')` → `person(first_name='Reginaldo', last_name='G.')` → `source_crosswalk(source_system='instantly', source_id='998172', source_type='cold_email')`.
11. **Typical workflows.** New source row → normalize → match → create/attach master ID → recompute survivorship. Merge of two masters → `merged_into` set, audited, reversible.
12. **Enterprise best practices.** Golden-record survivorship with provenance; UNIQUE `(source_system, source_id)`; deterministic before probabilistic; human review for low-confidence merges; SCD2 with `valid_from/valid_to/is_current`.
13. **Benefits.** True dedup; stable joins; complete lineage; backward compatibility (legacy keys preserved in crosswalk).

14. **Challenges.** Match precision vs recall; survivorship rule maintenance; merge/unmerge correctness; steward workload.
15. **Scalability.** Bounded at identity count (~1–2 M); B-tree point lookups; trigram indexes for fuzzy match; no partitioning needed.
16. **Security.** Highest PII concentration → strictest `GRANT`s; column-level PII classification; masking for non-privileged roles.
17. **Data governance.** Lineage per attribute; match-review queue; DQ findings flagged, not silently altered; DSAR-erasable flags.
18. **Performance.** Blocking keys to bound match comparisons; partial/covering indexes; survivorship materialised into `golden_record_full`.
19. **Future extensibility.** Household/company graphs, external identity graph enrichment, ML-based matching swapped in behind the same registry.

Components (Domain A)

- **identity_registry** — *Master table.* The identity spine: one row = one permanent golden `master_lead_id` (uuid, `gen_random_uuid()`). Exists so identity is generated once and never reused; merges recorded via `merged_into`. Every other domain FKs to it. Apps treat it as *the* lead ID; AI uses it as the join key for features. Schema: `(master_lead_id uuid PK, status text, merged_into uuid FK, primary_person_id uuid, canonical_hash text, created_at, updated_at)`.
- **person** — *Master table (SCD2).* Human attributes tied to a master ID. Stores names; versioned over time. Apps display it; AI derives name/role features. `(person_id uuid PK, master_lead_id uuid FK, first_name, last_name, full_name, valid_from, valid_to, is_current)`.
- **company** — *Master table (SCD2).* Firmographics (legal name, domain, industry, size/revenue band). Linked to persons via employment. `(master_company_id uuid PK, legal_name, domain, industry, size_band, revenue_band, is_current)`.
- **address** — *Master table (SCD2).* Normalized postal locations with `normalized_hash` for dedup; the anchor for serviceability. `(address_id uuid PK, line1, line2, city, state, postal_code, country, normalized_hash)`.
- **contact_point** — *Master table.* Typed reachable endpoints (phone/email) as first-class children, each with source and validity — this is what lets phone-keyed and email-keyed sources collapse into one person. `(contact_point_id, master_lead_id FK, kind, value, status, source)`.
- **source_crosswalk** — *Master/bridge table.* Maps one golden ID to many `(source_system, source_id, source_type)` rows — the MDM heart and dedup ledger. `(crosswalk_id bigint PK, master_lead_id FK, source_system, source_id, source_type, match_method, confidence)`.

- **agent** — *Dimension/master table.* Sales/rep directory (name, role, team, territory, external logins). Referenced by interactions and orders. `(agent_id uuid PK, full_name, email, role, team_id, territory_id, external_ids jsonb, status)`.
- **product** — *Dimension table.* Sellable fiber plans/offers (carrier, speeds, price, term, offer_code). "Offer" is a product packaging, not a separate table. `(product_id PK, carrier, name, download_mbps, upload_mbps, monthly_price, term_months, offer_code)`.
- **campaign** — *Dimension table.* Marketing/outreach campaigns per channel. Referenced by interactions and memberships. `(campaign_id text PK, name, channel_id FK, kind, status, start_at, end_at)`.
- **channel** — *Reference/dimension table.* Controlled vocabulary of interaction channels (`SMS, Email, Voice, Door, Web, Mail`); one definition reused by interaction + campaign. `(channel_id int PK, name, kind, provider)`.

---

6. Domain B — Activity / Event

> Components: `interaction`, `enrichment`, `opportunity`, `order`, `order_line`, `consent / DNC`.

1. **Purpose.** Record everything that *happens* to a lead as immutable, time-ordered events.
2. **Business Objective.** Enable attribution, journey analytics, compliance, and revenue tracking from a complete, trustworthy history.
3. **Responsibilities.** Append events (never overwrite); preserve chronology; link each event to its master identity, channel, agent, campaign, product; support retention/partitioning.
4. **Why this domain exists.** Events are high-velocity and unbounded; keeping them out of the golden row prevents lock contention, preserves history, and makes time-series analytics and partitioning possible.
5. **Design Principles.** Append-only; immutable; partition by event time; surrogate + time composite PK; foreign keys to master dimensions.
6. **Main Components.** `interaction` (touches), `enrichment` (data-provider events), `opportunity` / `order` / `order_line` (deal & revenue), `consent` /DNC (compliance state changes).
7. **Data stored.** Timestamped touches with channel/agent/campaign/disposition; enrichment provenance; deal stages and line items; consent grants/revocations.

8. **Relationships.** Every event FKs to `identity_registry`; interactions FK to `channel` / `agent` / `campaign`; order lines FK to `product`. Serving derives "first/last/journey" from these.
9. **Typical tables.** `interaction` (**transaction, partitioned**), `enrichment` (**transaction, partitioned**), `opportunity`, `order`, `order_line`, `consent` (all **transaction tables**).
10. **Example record.** `interaction(master_lead_id='7f3a…', channel_id=1 /*SMS*/, agent_id='…', campaign_id='C2UUBR7', interaction_type='sms_out', occurred_at='2026-07-08T14:03Z', software_type='sms_platform')`.
11. **Typical workflows.** Outbound SMS sent → interaction row. Provider returns phone → enrichment row. Rep books install → opportunity → order → order_line. Recipient texts STOP → consent revocation event → serving suppresses.
12. **Enterprise best practices.** Event sourcing; monthly RANGE partitioning; `pg_partman` auto-rollout; BRIN on time; never mutate history — correct with compensating events.
13. **Benefits.** Full auditability; accurate multi-touch attribution; compliance evidence; no contention on the golden record.
14. **Challenges.** Volume growth; partition management; event schema evolution; deduping duplicate webhooks.
15. **Scalability.** Volume scales independently of identity count; partition pruning + retention keep hot data small; shard by time then hash at extreme scale.
16. **Security.** Message bodies/PII in `metadata` → restricted; consent is a compliance-critical, tamper-evident record.
17. **Data governance.** Immutable audit trail; consent lineage; retention policy per event type; TCPA/DNC enforcement at read time.
18. **Performance.** Partition pruning; `(master_lead_id, occurred_at DESC)` for per-lead reads; covering indexes for hot analytic slices.
19. **Future extensibility.** New channels/event types add rows, not columns; stream to a real-time bus; feed ML training sets directly.

Components (Domain B)

- **interaction** — *Transaction table (partition by `occurred_at`).* Every cross-channel touch. Exists to hold the unbounded journey that the flat design could not. Relates to identity + channel + agent + campaign. Apps read the timeline; AI derives recency/frequency/sequence features; the journey view pivots the first N + last touches from it. Adds `agent_login` + `software_type` (dialer/warroom/sms_platform) for the requested agent-activity trail. Schema: `(interaction_id bigint, master_lead_id uuid FK, channel_id int FK, agent_id uuid FK, campaign_id text FK, direction, interaction_type, disposition, agent_login, software_type, occurred_at`

`timestamptz, metadata jsonb, PRIMARY KEY(interaction_id, occurred_at)) PARTITION BY RANGE(occurred_at)`.

- **enrichment** — *Transaction table (partitioned).* One row per provider enrichment (Apollo/LeadMagic/IPQS/SARA). Exists so *full* enrichment history is preserved (not just "last"); "last enrichment date/type/source" is a cheap `max(occurred_at)` derivation. `(enrichment_id, master_lead_id FK, enrichment_type, provider, source_type, status, occurred_at, detail jsonb)`.
- **opportunity** — *Transaction table.* A qualified deal in the pipeline linking lead → product → owning agent → campaign. Feeds forecasting and conversion analytics. `(opportunity_id uuid PK, lead_id FK, person_id FK, account_id FK, product_id FK, owner_agent_id FK, campaign_id FK, stage, amount)`.
- **order** — *Transaction table.* A won sale (install) with amount and install date. `(order_id uuid PK, account_id FK, opportunity_id FK, agent_id FK, status, total_amount, ordered_at, install_date)`.
- **order_line** — *Transaction table.* Line items of an order, each referencing a `product`. Normalises multi-product orders. `(order_line_id PK, order_id FK, product_id FK, qty, unit_price)`.
- **consent / DNC** — *Transaction/compliance table.* Consent grants and revocations per channel — the fail-closed gate for SMS/dialer (TCPA). `(consent_id, master_lead_id FK, person_id FK, channel_id FK, status, basis, effective_at)`.

---

7. Domain C — Reference

> Component: `address_serviceability` (AT&T, Kinetic, Spectrum, …).

1. **Purpose.** Store curated external facts about the world — here, which carrier/technology is serviceable at an address, over time.
2. **Business Objective.** Sell the right product to the right address; a fiber ISP lives or dies by accurate serviceability.
3. **Responsibilities.** Maintain per-address, per-carrier availability with technology, speed, source, and check timestamp; supersede stale checks.
4. **Why this domain exists.** Serviceability is *address-scoped and multi-valued* (many carriers, changing over time) — it is neither identity nor activity, so it deserves its own reference domain rather than being flattened into a single "fiber" column.

5. **Design Principles.** Keyed to `address_id`; temporal (`checked_at`); multi-carrier rows; source-attributed; curated (validated before trust).
6. **Main Components.** `address_serviceability`, reconciling the existing `ulda_ref.v_address_serviceability` view + the SARA/Kinetic/FCC checkers.
7. **Data stored.** `(address, carrier, fiber_available, technology, max_download_mbps, checked_at, source)`.
8. **Relationships.** FK to `ulda_master.address`; consumed by Serving to surface "Fiber Availability" on Lead-360 and to prioritise dialer/canvass lists.
9. **Typical tables.** `address_serviceability` (**reference table**); `v_address_serviceability` (**standard view**).
10. **Example record.** `(address_id='…', carrier='att', fiber_available=true, technology='fiber', max_download_mbps=5000, source='sara', checked_at='2026-07-10')`.
11. **Typical workflows.** Address checker runs → upsert serviceability row → Lead-360 shows fiber=yes → lead routed to the fiber campaign/dialer.
12. **Enterprise best practices.** Idempotent upserts; keep history of checks; source + confidence; separate reference data from operational identity.
13. **Benefits.** Accurate targeting; fewer wasted contacts; reusable across leads at the same address.
14. **Challenges.** Coverage-data freshness; carrier API variance; address-matching to the reference.
15. **Scalability.** Bounded by unique addresses; index by `(address_id)` and `(carrier, fiber_available)`.
16. **Security.** Low PII, but competitive-sensitive; standard read grants.
17. **Data governance.** Source lineage + `checked_at` freshness; supersede rather than overwrite.
18. **Performance.** Small, cache-friendly; join to Lead-360 via `address_id`.
19. **Future extensibility.** Add competitors, pricing, promo windows, install-feasibility (MDU/SFU), FCC/BDC datasets.

Component (Domain C)

- **address_serviceability** — *Reference table.* Per-address × carrier × time serviceability. Exists to model "Fiber Availability (AT&T, Kinetic, Spectrum…)" correctly as many rows, not one column. Relates to `address`; consumed by Lead-360 and lead routing. Apps show availability; AI can feature-encode "fiber_available_att" etc. `(serviceability_id bigint PK, address_id uuid FK, carrier text, fiber_available bool, technology text, max_download_mbps int, checked_at timestamptz, source text, UNIQUE(address_id, carrier, checked_at))`.

8. Domain D — Serving / Customer Data Platform (CDP)

> Components: `mv_lead_360`, `v_lead_journey_flat`, `mv_lead_score`, `lead_embedding` (pgvector).

1. **Purpose.** Present fast, denormalised, purpose-built projections of the mastered data for applications, analytics, and AI.
2. **Business Objective.** Give every consumer a single, instant, correct customer view (Lead-360) without paying the cost of joining the normalised model on every read.
3. **Responsibilities.** Materialise the unified profile; compute the cross-channel journey; score/tier leads; expose LLM-safe features and vectors; activate segments (reverse-ETL) under consent.
4. **Why this domain exists.** OLTP-normalised models are optimal for writes and integrity but slow for wide reads; the Serving domain trades storage for read speed and shields the master from analytical load.
5. **Design Principles.** Derived (never a source of truth); rebuildable; refreshed `CONCURRENTLY` (no read blocking); PII-masked for AI; consent-gated for activation.
6. **Main Components.** `mv_lead_360` (profile), `v_lead_journey_flat` (timeline pivot), `mv_lead_score` (scoring), `lead_embedding` (semantic vectors), analytics MVs, activation views.
7. **Data stored.** Flattened identity+geo+score per lead; pivoted first-4+last channels; score/tier; 384-dim embeddings; rollups.
8. **Relationships.** Reads from Master + Events + Reference; writes to nothing authoritative; feeds Warroom, dialer, BI, AI, reverse-ETL.
9. **Typical tables.** `mv_lead_360`, `mv_lead_score`, `mv_analytics_*`, `lead_embedding` (**materialised views / derived tables**); `v_lead_journey_flat`, `v_lead_master_flat`, `v_lead_llm_safe` (**standard views**).
10. **Example record.** `mv_lead_360(master_lead_id='7f3a…', name='Reginaldo G.', street='7602 riptide dr', state='TX', zip='77072', completeness=0.7, source_systems={instantly}, source_count=1)`.
11. **Typical workflows.** Drain reconciles → `REFRESH MATERIALIZED VIEW CONCURRENTLY mv_lead_360` → apps/AI read the fresh profile; nightly analytics refresh; batch embedding on the replica.
12. **Enterprise best practices.** Concurrent refresh; unique index for concurrency; separate OLAP read path; semantic layer (governed metric definitions); feature store for ML.

13. **Benefits.** Sub-second wide reads; consistent single view; AI-ready; analytics isolated from OLTP.
14. **Challenges.** Refresh scheduling/freshness lag; MV storage; embedding compute cost; keeping derivations in sync with source.
15. **Scalability.** Move refreshes/embeddings to a replica; incremental MV strategies; HNSW/ivfflat tuning for vector scale.
16. **Security.** `ulda_web` read-only; `v_lead_llm_safe` strips direct PII before prompts; activation is consent-gated.
17. **Data governance.** Metric catalog (one governed definition per metric); PII classification; reverse-ETL audit; lineage from MV back to source.
18. **Performance.** Materialisation + covering indexes; `CONCURRENTLY` refresh; pre-warmed caches; read-replica offload.
19. **Future extensibility.** Real-time (streaming) Lead-360; ML scoring v2 via model registry; RAG over `lead_embedding`; new activation destinations.

Components (Domain D)

- **mv_lead_360** — *Materialised view.* The unified, flattened golden profile (identity + geo + completeness + sources). Exists for instant single-customer reads. Built on `golden_record_clean`; refreshed `CONCURRENTLY`. Apps render it; AI joins features to it. Example schema: `(master_lead_id uuid, name, first_name, last_name, company, email, phone, street, city, state, zip, completeness numeric, source_systems text[], source_count int)`.
- **v_lead_journey_flat** — *Standard view.* Pivots `interaction` into `1st/2nd/3rd/4th channel + date + campaign` and `last channel/date` via window functions — your requested columns, computed not stored, uncapped. Apps/exports read it; analytics measures multi-touch paths. Materialise if it becomes hot.
- **mv_lead_score** — *Materialised view.* Per-lead score (avg ≈ 76/100) and tier (A/B/C/D) from a governed scoring function with MLOps lineage (`model_registry`, `score_run`), so an ML model can replace the heuristic with no consumer change. Apps prioritise by tier; AI consumes score as a feature.
- **lead_embedding (pgvector)** — *Derived table + index.* 384-dim BAAI/bge-small vectors per lead with an HNSW index for cosine semantic search (`semantic_search()`). Exists to power similarity, dedup assistance, and RAG. AI uses it directly; the mass-embed runs on the replica/idle to protect the primary. `(master_lead_id uuid PK, embedding vector(384))` + HNSW.

---

9. Cross-Cutting Capabilities

- **How it enables Lead-360.** One consumer object (`v_lead_master_flat`) joins `mv_lead_360` ⋈ `v_lead_journey_flat` ⋈ last-enrichment ⋈ last-order ⋈ serviceability → every requested column from one read, always correct.
- **How it supports AI / semantic search / ML.** Clean per-domain **feature store**; `lead_embedding` + HNSW for semantic search; `v_lead_llm_safe` for prompt-safe context; `model_registry` for swapping heuristic scoring with trained models; append-only events are ideal ML training data.
- **How it supports analytics & reporting.** Governed **metric catalog** (one definition per metric) + materialised rollups (`mv_analytics_by_state/tier/source` , KPI board) refreshed off the OLTP path.
- **How it supports a CDP.** Persistent unified profile + identity resolution + consent-gated **reverse-ETL** activation to Warroom/SMS/dialer.
- **Scalability, maintainability, performance.** Event volume scales independently of identity; partitioning + retention bound hot data; MVs isolate read load; schema boundaries make each domain independently evolvable and testable.

---

10. Final Architectural Assessment

10.1 Why this is enterprise-grade

It exhibits the defining properties of enterprise data platforms: a single source of truth with immutable identity; complete lineage and auditability; separation of concerns by change-velocity; governed metrics and PII/consent controls; zero-downtime, reversible evolution; and independent scalability of identity vs activity vs serving. It is not a schema — it is a governed platform with an ingestion contract, a resolution engine, a serving layer, and a governance spine.

10.2 Vs traditional CRM database design

Traditional CRMs centre on a wide `Contact` / `Lead` table with bolted-on activity tables and app-enforced dedup. ULDA inverts this: identity is *resolved and mastered* (not assumed), activity is *event-sourced* (not last-value), serving is *materialised* (not live-joined on every page), and governance is *first-class* (not a spreadsheet). The result is better dedup, real history, cleaner AI, and a genuine path to a billion rows.

10.3 Recommended improvements before production-hardening

1. **Make `interaction` & `enrichment` partitioned append-only now** (they are empty — ideal timing).
2. **Backfill `interaction`** from `disposition_event` + Warroom SMS/dialer/reply activity — the one data task that lights up the journey/agent history.
3. **Stand up a read replica** to offload analytics/embeddings and unblock full 1 M embedding.
4. **Automate partition roll-out** (`pg_partman`) and retention.
5. **Formalise the consent gate as fail-closed** in every activation path (TCPA).
6. **Add a `timezone` attribute** (derived from state/ZIP) for quiet-hours dialing.

10.4 Additional domains/components worth adding at millions-of-leads scale

- **Suppression / compliance domain** (DNC registries, litigators, opt-outs) — beyond per-lead consent.
- **Household / account graph** (already seeded: `household` , `account`) — B2B and residential rollups.
- **Attribution / marketing-performance domain** (spend ⋈ campaign ⋈ conversion).
- **Data-quality & observability domain** (freshness SLAs, anomaly detection) — extend `ulda_gov` .
- **Real-time streaming tier** (CDC → bus) when sub-minute Lead-360 is required.

10.5 Verdict

The four-domain design is the correct enterprise architecture for this problem. Implemented on one PostgreSQL database with five schemas and a future read replica, it is scalable to millions of leads today and to a billion with the documented partitioning/replica path — while remaining maintainable, AI-ready, and production-safe. The primary remaining work is **data population (interaction backfill) and read scale-out (replica)**, not redesign.