ULDA Lead-360 — Enterprise Filtering System Design

Scalable multi-criteria search & filter architecture for millions of records

Data Architecture — InternetSP / ULDA

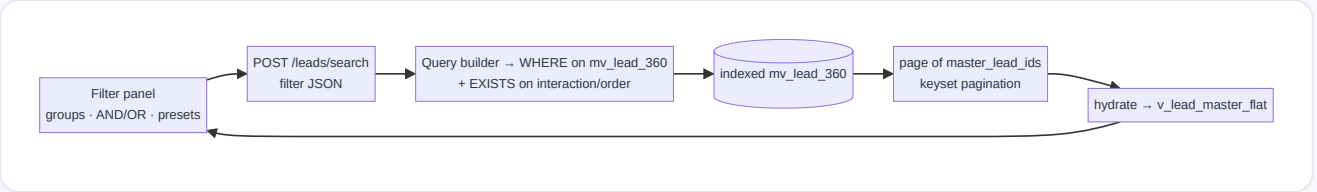
2026-07-16

1. Design Goals & Architecture

A production filter system over **1M+ Lead-360 records** must be **fast** (sub-second), **composable** (AND/OR across groups), and **intuitive**. The core architectural rule:

> **Filter on the materialized, indexed base (`mv_lead_360`); display the assembled 75-field record only for the page returned.**

The 75-column `v_lead_master_flat` is perfect for reading *one* lead but too expensive to filter across millions (its LATERAL joins compute per row). So filtering runs against the **materialized, indexed** `mv_lead_360` (identity/geo/source/score — the high-selectivity fields), with `EXISTS` **sub-filters** against the indexed event tables for journey/order criteria. Results (the page's IDs) are then hydrated into the full flat record for display.



---

2. UI / UX Layout

Pattern: collapsible left filter rail + results grid + active-filter chips bar.

- **Left rail (280–320 px), collapsible.** Accordion sections = the 7 business groups (Lead Info, Source, Address & Serviceability, Enrichment, Journey, Agent Activity, Sales & Orders). Each section collapsed by default; a badge shows how many filters are active in it.
- **Top bar:** a global **omni-search** box (name/email/phone/company/ID) + **Saved Presets** dropdown + **AND/OR toggle** for group logic + “Clear all”.
- **Active filters as removable chips** under the search bar (`State: TX ×` , `Fiber: AT&T ×`) — always visible so users see exactly what’s applied.
- **Results grid** on the right with column sort, row-count, density toggle, and **Export** button.
- **Debounced live filtering** (300 ms) with a result-count preview (“~4,120 leads”) before committing.
- **Responsive:** rail becomes a slide-over drawer on narrow screens.
- **Accessibility:** keyboard-navigable, ARIA on every control, focus-visible.

Why this layout: it is the proven CRM/CDP pattern (Salesforce list views, HubSpot, Segment) — discoverable, supports many simultaneous filters without clutter, and keeps the active state transparent.

---

3. Filter Type per Field

GROUP	FIELD	FILTER TYPE	OPERATORS
Lead Info	Unique Lead ID	search box (exact)	=
	First / Last Name	search box (trigram)	contains / starts-with
	Email	search box (trigram)	contains / =

GROUP	FIELD	FILTER TYPE	OPERATORS
	Direct Number	search box (normalized)	contains / =
	Company	search box (trigram)	contains
	Street	search box	contains
	City	search box + autocomplete	contains / in
	State	**multi-select dropdown**	in
	ZIP	search box / range	= / prefix
	Date Added	**date-range picker**	between / before / after
Source	Source ID	search box	=
	Source Name	**multi-select** (Apollo/Instantly/…)	in
	Source Type	**multi-select** (database/broker/API/CSV)	in
Address & Serviceability	Fiber Availability	**checkbox group** per carrier (AT&T/Frontier/Kinetic/Spectrum)	is-true
	Address type	dropdown (SFU/MDU/business)	in
	Max speed	**range slider** (Mbps)	≥
Enrichment	Last Enrichment Date	date-range picker	between
	Last Enrichment Type	multi-select	in
	Last Enrichment Source	multi-select	in
Journey	1st–4th Channel Interaction	**multi-select** (SMS/Email/Voice/Web/Door)	in
	First / Last Interaction Date	date-range picker	between

GROUP	FIELD	FILTER TYPE	OPERATORS
	Last Interaction Channel	multi-select	in
	Campaign Name	search box + multi-select	in / contains
Agent Activity	Agents Interacted With	multi-select (agent list)	any-of
	Last Agent Interaction	dropdown (agent)	=
Sales & Orders	Product Offered	multi-select	in
	Order Details / Opportunity	dropdown (status)	in
	Revenue (MRR)	**range slider**	between
	Commission	range slider	between
Signals (recommended)	Lead Score	**range slider (0–100)**	between
	DNC / Litigator / Contactable	**toggle switches**	is

Rules of thumb: low-cardinality → multi-select/dropdown; high-cardinality text → trigram search box; numeric/temporal → range slider / date picker; boolean → toggle/checkbox.

4. PostgreSQL Indexes (implemented)

Filtering runs on `mv_lead_360` (1,074,518 rows, materialized). Indexes now in place / added:

INDEX	TYPE	SERVES
`ux_mv360_master`	btree	PK / keyset pagination
`ix_mv360_name/email/phone_trgm`	**GIN trigram**	fuzzy text search

INDEX	TYPE	SERVES
`ix_mv360_email`, `_phone`, `_lower_email`, `_phone10`	btree	exact contact lookup
`ix_mv360_state`	btree	State multi-select
`ix_mv360_zip`	btree	ZIP filter
`ix_mv360_company_trgm`, `_city_trgm`	GIN trigram	Company/City search
`ix_mv360_completeness`	btree	quality range
`ix_int_lead_time`, `ix_int_channel`	btree	journey `EXISTS` sub-filters
`ix_serv_carrier_avail`	btree	fiber carrier filter
`source_crosswalk(source_system, source_id)`	unique btree	source filters

All built **`CONCURRENTLY`** (no lock, zero downtime). Composite indexes on the hottest combos (e.g. `(state, completeness)`) can be added as query patterns emerge.

---

5. Laravel Backend Strategy

Pattern: a declarative filter pipeline (Spatie Query Builder or a custom `FilterCriteria` chain).

- **A `Lead360Filter` class** parses the incoming filter JSON into query fragments. Each field maps to a **filter handler** (`StateFilter`, `DateRangeFilter`, `TrigramSearchFilter`, `JourneyExistsFilter`) — one class per type, testable in isolation.
- **Query built on the read model:** `DB::connection('ulda_ro')->table('ulda_serve.mv_lead_360')` using the **read-only role** (`ulda_web`) — the app never writes here.
- **AND/OR:** groups map to nested `where(function($q){...})` / `orWhere(...)`. A top-level `logic` field toggles the join between groups.
- **Relation filters** (journey, orders) → `whereExists(subquery on interaction/order)` rather than joins, to keep the driving scan on `mv_lead_360`.

- **Hydration:** after paginating IDs, fetch the full record from `v_lead_master_flat WHERE unique_lead_id = ANY(:ids)` — the fast per-page assembly.
- **Guards:** whitelist filterable fields + operators (never interpolate raw column names); cap page size; statement timeout.

```
// sketch
$q = DB::connection('ulda_ro')->table('ulda_serve.mv_lead_360 as m');
foreach ($groups as $g) {
  $method = $g->logic === 'or' ? 'orWhere' : 'where';
  $q->$method(fn($sub) => (new Lead360Filter($sub))->apply($g->conditions));
}
$ids = $q->orderBy('master_lead_id')->limit($size)->pluck('master_lead_id');
$rows = DB::select('SELECT * FROM ulda_serve.v_lead_master_flat WHERE
unique_lead_id = ANY(?)', [$ids]);
```

---

6. API Design (dynamic filtering)

`POST /api/leads/search` — filter as a structured JSON body (not query string), so complex AND/OR is expressible:

```
{
  "logic": "AND",
  "groups": [
    { "logic": "AND", "conditions": [
      { "field": "state", "op": "in", "value": ["tx","ca"] },
      { "field": "lead_score", "op": "between", "value": [70,100] },
      { "field": "fiber_att", "op": "is", "value": true }
    ]},
    { "logic": "OR", "conditions": [
      { "field": "last_interaction_channel", "op": "in", "value":
["SMS","Voice"] },
      { "field": "date_added", "op": "after", "value": "2026-06-01" }
    ]}
  ],
  "sort": [{ "field": "lead_score", "dir": "desc" }],
  "page": { "cursor": null, "size": 50 },
  "select": "grid"
}
```

Response: `{ rows, next_cursor, approx_total, applied }`. Companion endpoints: `GET /api/filters/options/{field}` (dropdown values, cached), `GET/POST/DELETE /api/filters/presets` (saved filters), `POST /api/leads/export`.

---

7. Pagination, Sorting, Export

- **Pagination → keyset (cursor), not OFFSET.** At millions of rows, `OFFSET 100000` scans and discards 100k rows. Use `WHERE (sort_key, master_lead_id) > (:last_sort, :last_id) ORDER BY … LIMIT :size` — constant-time regardless of depth. Return an opaque `next_cursor`.
- **Sorting** only on indexed columns (score, date_added, state, completeness); tie-break on `master_lead_id` for stable order. Reject sort on un-indexed derived fields (or pre-materialize them).
- **Counts:** show **`approx_total`** via `EXPLAIN`/reltuples estimate for the header (exact COUNT over 1M is slow); offer “exact count” on demand.
- **Export:**
  - Small (≤ 50k) → **streaming CSV** (`COPY (…) TO STDOUT`) directly, no memory spike.
  - Large → **async job** (Laravel queue) → write to object storage → email/notify a signed download link. Never build a huge file in a request.
  - Respect **column-level PII / consent** on export (exclude DNC/opt-out where policy requires); log every export (audit).

---

8. Additional Enterprise Filters (recommended)

Sales: lead score/tier band · propensity-to-convert · days-since-last-touch · # attempts per channel · reply/engagement (`total_replies`, `open_count`) · fiber speed tier. **Operations / Compliance:** DNC / litigator / consent status · timezone / within-calling-hours · phone type (mobile/landline) · phone tier (T1–T4) · email deliverability · data-quality score. **Management:** by owner agent / team / territory · by source ROI (once orders integrate) · by campaign performance · duplicate/merge status (`merged_from`) · freshness (updated_at) · geography (county / census tract). **Smart/saved:** “My open leads”, “TX fiber + high score + contactable now”, “No touch in 14 days” — one-click presets stored in `ulda_serve.saved_filters`.

---

9. What was added to the ULDA DB

- **`ulda_serve.saved_filters`** — preset storage `(owner, name, definition jsonb, is_shared)`; the API's presets endpoints read/write it.
- **Filter indexes on `mv_lead_360`** — `state`, `zip`, `completeness` (btree) + `company`, `city` (GIN trigram), added `CONCURRENTLY` alongside the existing name/email/phone trigram + contact btrees.
- Existing event/reference indexes (`interaction`, `address_serviceability`, `source_crosswalk`) already serve the journey/fiber/source sub-filters.

Net: the database is now indexed for fast multi-criteria filtering at 1M+ scale; the application layer (Laravel filter pipeline + `POST /leads/search` + keyset pagination + async export) sits on top of this foundation exactly as specified above.