OB-POC — Schema Entity Overview

> ***Last reconciled:*** *2026-02-11 — against 77 migrations, 57 DSL verb domains, CLAUDE.md* ***Scope:*** *`"ob-poc"` schema only (226 tables). External schemas (`custody` , `kyc` , `agent` , `teams`) referenced but not detailed.* ***Method:*** *SQL DDL cross-referenced with DSL verb YAML (`rust/config/verbs/*.yaml`) to validate domain groupings.*

Reading Order — The Commercial-to-Operational Flow

This document follows the real business flow from client acquisition through to operational servicing:

```
1. CLIENT GROUP          — Who is the client? (Allianz, BlackRock, Aviva)
       │
       ▼
2. DEALS & COMMERCIALS   — What are we selling? Products, negotiated rate cards
       │
       ▼
3. BOOKING PRINCIPALS    — Who books it, where? Legal entity + jurisdiction + rules
       │
       ▼
4. CONTRACTS & GATE      — Contracted products → CBU subscription (onboarding gate)
       │
       ▼
5. CBU ONBOARDING        — CBU sets subscribe to product sets
       │
       ▼
6. CBU INTERNALS         — Entities with roles, instrument matrix, trading profiles
```

Top-Level Domain Map

```
graph TB
    subgraph "1. Client Group"
        CG[client_group] --> CGE[client_group_entity]
        CGE --> E[entities]
        CG --> CGA[client_group_anchor]
        CG --> CGAL[client_group_alias]
    end

    subgraph "2. Deals & Commercials"
        CG --> D[deals]
        D --> DP[deal_participants]
        D --> DRC[deal_rate_cards]
        DRC --> DRCL[deal_rate_card_lines]
        D --> DC[deal_contracts]
        D --> DON[deal_onboarding_requests]
        D --> FBP[fee_billing_profiles]
    end

    subgraph "3. Booking Principals"
```

```
        LE[legal_entity] --> BP[booking_principal]
        BL[booking_location] --> BP
        BP --> SA[service_availability]
        BP --> CPR[client_principal_relationship]
        CPR --> CG
        CPR --> P[products]
        RS[ruleset] --> R[rule]
    end

    subgraph "4. Contracts & Gate"
        LC[legal_contracts] --> CP[contract_products]
        CP --> CSUB[cbu_subscriptions]
        CSUB --> CBU[cbus]
    end

    subgraph "5. CBU Internals"
        CBU --> CER[cbu_entity_roles]
        CER --> E
        CBU --> CPS[cbu_product_subscriptions]
        CPS --> P
        CBU --> CTP[cbu_trading_profiles]
        CBU --> CMO[cbu_matrix_product_overlay]
        CBU --> CRI[cbu_resource_instances]
    end

    subgraph "6. Product / Service / Resource"
        P --> PS[product_services]
        PS --> S[services]
        S --> SRC[service_resource_capabilities]
        SRC --> SRT[service_resource_types]
    end

    DC --> LC
    DON --> CBU
    SA --> S
    FBP --> DRC
```

Notation

- **Table names** are shown as in the DDL: `"ob-poc".table_name` (schema prefix omitted for readability).
- **PK / FK** are from `ALTER TABLE ... ADD CONSTRAINT ...` blocks.
- **Verb domain** shows which DSL verb YAML file operates on the table (e.g., `cbu.yaml` → 19 verbs).
- Tables with no verb domain are included only when essential to understanding the data structure.

---

1) Client Group — The Apex

Verb domains: `client-group` (23 verbs), `gleif` (16), `ubo` (22), `ownership` (16), `control` (15), `manco-group` (16)

The client group is the apex entity. Everything flows from here — deals, entities, CBUs, UBO discovery. A client group (Allianz, BlackRock, Aviva) aggregates entities discovered via GLEIF research and curated through a review workflow.

```
erDiagram
    client_group ||--o{ client_group_entity : "group_id"
    client_group ||--o{ client_group_relationship : "group_id"
    client_group ||--o{ client_group_alias : "group_id"
    client_group ||--o{ client_group_anchor : "group_id"
    client_group ||--o{ deals : "primary_client_group_id"
    client_group ||--o{ client_principal_relationship : "client_group_id"
    client_group_entity }o--|| entities : "entity_id"
    client_group_entity }o--o| cbus : "cbu_id"
    client_group_entity ||--o{ client_group_entity_roles : "cge_id"
    client_group_relationship }o--|| entities : "parent_entity_id"
    client_group_relationship }o--|| entities : "child_entity_id"
    client_group_anchor }o--|| entities : "anchor_entity_id"

    client_group {
        uuid id PK
        text canonical_name
        text short_code UK
        varchar discovery_status
        varchar discovery_root_lei
        integer entity_count
        integer pending_review_count
    }

    client_group_entity {
        uuid id PK
        uuid group_id FK
        uuid entity_id FK
        uuid cbu_id FK
        text membership_type
        varchar review_status
    }

    client_group_anchor {
        uuid id PK
        uuid group_id FK
        uuid anchor_entity_id FK
        text anchor_role
        text jurisdiction
        float confidence
    }

    client_group_relationship {
        uuid id PK
        uuid group_id FK
        uuid parent_entity_id FK
        uuid child_entity_id FK
        varchar relationship_kind
        varchar review_status
        uuid promoted_to_relationship_id FK
    }

    client_group_alias {
        uuid id PK
        uuid group_id FK
        text alias
        text alias_norm
        boolean is_primary
    }
```

Anchor roles — each group has anchors per jurisdiction for different use cases:

Role	Use Case
`ultimate_parent`	UBO discovery, ownership tracing
`governance_controller`	Session scope, CBU loading
`book_controller`	Regional operations
`operating_controller`	Day-to-day operations
`regulatory_anchor`	Compliance, KYC

Supporting tables:

Table	Purpose
`client_group_alias_embedding`	Versioned embeddings per alias (for "allianz" → Allianz GI resolution)
`client_group_anchor_role`	Anchor role types
`client_group_entity_roles`	GLEIF roles (SUBSIDIARY, ULTIMATE_PARENT) from research phase
`client_group_entity_tag`	Entity classification tags
`client_group_relationship_sources`	Source provenance for relationships

---

2) Deals & Commercials — What Are We Selling?

Verb domains: `deal` (42 verbs), `billing` (17)

A client group has one or more **deals**. Each deal is the commercial origination hub — it carries participants, products, negotiated rate cards, SLAs, and links to contracts. The rate card negotiation is a state machine (`DRAFT → PROPOSED → COUNTER_OFFERED ↔ REVISED → AGREED → SUPERSEDED`).

```
erDiagram
    deals }o--|| client_group : "primary_client_group_id"
    deals ||--o{ deal_participants : "deal_id"
    deals ||--o{ deal_contracts : "deal_id"
    deals ||--o{ deal_rate_cards : "deal_id"
    deals ||--o{ deal_slas : "deal_id"
    deals ||--o{ deal_documents : "deal_id"
    deals ||--o{ deal_ubo_assessments : "deal_id"
    deals ||--o{ deal_onboarding_requests : "deal_id"
    deals ||--o{ deal_events : "deal_id"
    deal_participants }o--|| entities : "entity_id"
    deal_rate_cards }o--|| legal_contracts : "contract_id"
    deal_rate_cards }o--|| products : "product_id"
    deal_rate_cards ||--o{ deal_rate_card_lines : "rate_card_id"
    deal_contracts }o--|| legal_contracts : "contract_id"
    deal_onboarding_requests }o--o| cbus : "cbu_id"
    deal_onboarding_requests }o--|| products : "product_id"

    deals {
        uuid deal_id PK
        varchar deal_name
        varchar deal_reference UK
        uuid primary_client_group_id FK
        varchar sales_owner
        varchar deal_status
        numeric estimated_revenue
        varchar currency_code
    }

    deal_participants {
        uuid deal_participant_id PK
        uuid deal_id FK
        uuid entity_id FK
        varchar participant_role
        boolean is_primary
    }

    deal_rate_cards {
        uuid rate_card_id PK
        uuid deal_id FK
        uuid contract_id FK
        uuid product_id FK
        varchar status
        integer negotiation_round
        uuid superseded_by FK
    }

    deal_rate_card_lines {
        uuid line_id PK
        uuid rate_card_id FK
        varchar fee_type
        varchar fee_subtype
        varchar pricing_model
        numeric rate_value
```

```
        numeric minimum_fee
        numeric maximum_fee
        varchar currency_code
        jsonb tier_brackets
    }

    deal_onboarding_requests {
        uuid request_id PK
        uuid deal_id FK
        uuid cbu_id FK
        uuid product_id FK
        varchar status
    }
```

Deal status state machine: `PROSPECT → QUALIFYING → NEGOTIATING → CONTRACTED → ONBOARDING → ACTIVE → WINDING_DOWN → OFFBOARDED` (any → `CANCELLED`)

Pricing models: `BPS` (basis points on AUM), `FLAT` (fixed fee), `TIERED` (volume-based), `PER_TRANSACTION`, `SPREAD`, `MINIMUM_FEE`

Fee billing — once a deal is active, billing profiles tie rate cards to CBUs:

Table	Purpose
`fee_billing_profiles`	Billing config per deal+contract+rate_card+CBU+product
`fee_billing_account_targets`	Which CBU resource instances to bill (links to `cbu_resource_instances`)
`fee_billing_periods`	Monthly/quarterly billing cycles with status machine
`fee_billing_period_lines`	Calculated fee amounts per target per period

Billing period status: `PENDING → CALCULATING → CALCULATED → REVIEWED → APPROVED → INVOICED → DISPUTED`

---

3) Booking Principals — Who Books It, Where?

Verb domains: `booking-principal` (9 verbs), `booking-location` (3), `client-principal-relationship` (4), `legal-entity` (3), `service-availability` (3), `rule` (3), `ruleset` (3), `contract-pack` (2)

The booking principal is the **policy anchor** — it defines which BNY legal entity, in which jurisdiction, can book which products for which clients. This is the middle layer between commercial (deals) and operational (CBU onboarding).

```
erDiagram
    legal_entity ||--o{ booking_principal : "legal_entity_id"
    legal_entity }o--o| entities : "entity_id"
    booking_location ||--o{ booking_principal : "booking_location_id"
    booking_location }o--o| master_jurisdictions : "jurisdiction_code"
    booking_principal ||--o{ service_availability : "booking_principal_id"
    booking_principal ||--o{ client_principal_relationship : "booking_principal_id"
```

```
service_availability }o--|| services : "service_id"
client_principal_relationship }o--|| products : "product_offering_id"
ruleset ||--o{ rule : "ruleset_id"

legal_entity {
    uuid legal_entity_id PK
    text lei UK
    text name
    text incorporation_jurisdiction
    text status
    uuid entity_id FK
}

booking_location {
    uuid booking_location_id PK
    text country_code
    text region_code
    text_arr regulatory_regime_tags
    varchar jurisdiction_code FK
}

booking_principal {
    uuid booking_principal_id PK
    uuid legal_entity_id FK
    uuid booking_location_id FK
    text principal_code UK
    text book_code
    text status
    timestamptz effective_from
    timestamptz effective_to
}

service_availability {
    uuid service_availability_id PK
    uuid booking_principal_id FK
    uuid service_id FK
    text regulatory_status
    text commercial_status
    text operational_status
    text delivery_model
    timestamptz effective_from
    timestamptz effective_to
}

client_principal_relationship {
    uuid client_principal_relationship_id PK
    uuid client_group_id FK
    uuid booking_principal_id FK
    uuid product_offering_id FK
    text relationship_status
    text contract_ref
    timestamptz onboarded_at
}

ruleset {
    uuid ruleset_id PK
    text owner_type
    uuid owner_id
    text name
    text ruleset_boundary
    integer version
    text status
}

rule {
    uuid rule_id PK
    uuid ruleset_id FK
    text name
    text kind
    jsonb when_expr
    jsonb then_effect
    integer priority
}
```

Three-lane service availability — each principal × service combination has three independent status lanes:

Lane	Values	Purpose
Regulatory	`permitted` / `restricted` / `prohibited`	Can we legally do this here?
Commercial	`offered` / `conditional` / `not_offered`	Do we want to sell this here?
Operational	`supported` / `limited` / `not_supported`	Can we actually deliver this here?

Rule kinds — rules evaluate client+principal+offering context:

Kind	Effect
`deny`	Block: client cannot be booked with this principal
`require_gate`	Gate: requires approval (credit committee, enhanced KYC)
`allow`	Explicit allow (overrides lower-priority denials)
`constrain_principal`	Narrow eligible principals
`select_contract`	Auto-select contract pack template

Ruleset boundaries — rules scoped to `regulatory`, `commercial`, or `operational` domains. Temporal overlap prevention via trigger.

Supporting tables:

Table	Purpose
`client_profile`	Immutable evaluation snapshot (segment, domicile, entity types, risk flags)

`client_classification`	Normalised regulatory classifications (MiFID II, Dodd-Frank, FATCA, CRS)
`eligibility_evaluation`	Immutable audit record of principal selection
`rule_field_dictionary`	Closed-world field registry for rule expression validation
`contract_pack`	Grouped contract package definitions
`contract_template`	Contract template definitions within a pack

---

4) Contracts & Onboarding Gate

Verb domains: `contract` (14 verbs), `contract-pack` (2)

The legal contract is the **onboarding gate** — CBUs can only subscribe to products that are explicitly listed in an active contract. The composite FK on `cbu_subscriptions` enforces this: `(contract_id, product_code) → contract_products`.

```
erDiagram
    legal_contracts ||--o{ contract_products : "contract_id"
    contract_products ||--o{ cbu_subscriptions : "contract_id+product_code"
    cbu_subscriptions }o--|| cbus : "cbu_id"
    deal_contracts }o--|| legal_contracts : "contract_id"

    legal_contracts {
        uuid contract_id PK
        varchar client_label
        varchar contract_reference
        date effective_date
        date termination_date
        varchar status
    }

    contract_products {
        uuid contract_id PK
        varchar product_code PK
        uuid rate_card_id FK
    }

    cbu_subscriptions {
        uuid cbu_id PK
        uuid contract_id PK
        varchar product_code PK
        varchar status
    }
```

The gate in action:

- `contract_products` defines what's contracted (product_code is VARCHAR — "CUSTODY", "FUND_ACCOUNTING", etc.)

- `cbu_subscriptions` has a composite FK to `contract_products(contract_id, product_code)` — you **cannot** subscribe a CBU to a product that isn't in the contract
- Status: `PENDING → ACTIVE → SUSPENDED → TERMINATED`

Note: `legal_contracts.client_label` is a denormalized text field (not UUID FK to client_group). `deal_rate_cards.product_id` is a UUID FK to `products` — these are two different product reference systems that coexist.

---

5) CBU Aggregate — The Operational Unit

Verb domains: `cbu` (19 verbs), `cbu-role-v2` (10), `trading-profile` (47), `cash-sweep` (9), `investment-manager` (7), `pricing-config` (14), `matrix-overlay` (14)

The CBU (Client Business Unit) is the **operational container** for onboarding + KYC scope. CBUs sit under a client group, subscribe to contracted products, and contain entities with roles. Each CBU has a trading profile that materializes into an instrument matrix.

```
erDiagram
    cbus }o--|| entities : "commercial_client_entity_id"
    cbus }o--o| products : "product_id"
    cbus ||--o{ cbu_entity_roles : "cbu_id"
    cbus ||--o{ cbu_product_subscriptions : "cbu_id"
    cbus ||--o{ cbu_trading_profiles : "cbu_id"
    cbus ||--o{ cbu_matrix_product_overlay : "cbu_id"
    cbus ||--o{ cbu_resource_instances : "cbu_id"
    cbus ||--o{ cbu_subscriptions : "cbu_id"
    cbu_entity_roles }o--|| entities : "entity_id"
    cbu_entity_roles }o--|| roles : "role_id"
    cbu_product_subscriptions }o--|| products : "product_id"
    cbu_matrix_product_overlay }o--o| cbu_product_subscriptions : "subscription_id"
    cbu_resource_instances }o--|| service_resource_types : "resource_type_id"

    cbus {
        uuid cbu_id PK
        varchar name
        varchar jurisdiction
        varchar client_type
        varchar cbu_category
        varchar status
        varchar kyc_scope_template
        jsonb risk_context
        jsonb onboarding_context
        uuid commercial_client_entity_id FK
        uuid product_id FK
    }

    cbu_entity_roles {
        uuid cbu_entity_role_id PK
        uuid cbu_id FK
        uuid entity_id FK
        uuid role_id FK
        uuid target_entity_id FK
        numeric ownership_percentage
        numeric authority_limit
        date effective_from
        date effective_to
    }

    cbu_product_subscriptions {
        uuid subscription_id PK
        uuid cbu_id FK
        uuid product_id FK
        varchar status
        date effective_from
        jsonb config
    }

    cbu_trading_profiles {
        uuid profile_id PK
        uuid cbu_id FK
        varchar profile_name
        varchar status
    }

    cbu_matrix_product_overlay {
        uuid overlay_id PK
        uuid cbu_id FK
        uuid subscription_id FK
        uuid instrument_class_id FK
        uuid market_id FK
        varchar currency
        uuid counterparty_entity_id FK
        jsonb additional_services
        varchar status
    }

    cbu_resource_instances {
        uuid instance_id PK
        uuid cbu_id FK
        uuid product_id FK
        uuid service_id FK
        uuid resource_type_id FK
        varchar status
        uuid market_id FK
        varchar currency
    }
```

Entity roles — entities connect to the CBU container with typed roles:

Role Category	Examples
Governance	Depositary, ManCo, Board Director, Auditor

Investment	Investment Manager, Sub-Advisor, Prime Broker
Operations	Transfer Agent, Custodian, Fund Administrator
Ownership	Asset Owner, Beneficial Owner, Shareholder

CBU category (fund type classification):

Category	Examples
`FUND_MANDATE`	UCITS, AIF, hedge fund
`CORPORATE_GROUP`	Corporate treasury, SPV
`PENSION`	Pension fund, sovereign wealth

Instrument matrix — the `cbu_matrix_product_overlay` is keyed by `(cbu_id, instrument_class_id, market_id, currency, counterparty_entity_id)`. It ties each trading cell to a product subscription with overlay config. The external `custody.cbu_instrument_universe` materializes the full trading universe.

CBU child tables:

Table	Verb Domain	Purpose
`cbu_entity_roles`	`cbu-role-v2`	Entity-to-CBU role assignments
`cbu_entity_roles_history`	—	Audit trail of role changes
`cbu_group_members`	`manco-group`	CBU membership in governance groups
`cbu_groups`	`manco-group`	Governance book groups (ManCo, apex parent)
`cbu_product_subscriptions`	`matrix-overlay`	Product subscriptions per CBU
`cbu_trading_profiles`	`trading-profile`	Trading mandate profiles
`cbu_matrix_product_overlay`	`matrix-overlay`	Per-cell instrument/market/currency config
`cbu_resource_instances`	`service-resource`	Provisioned resource instances
`cbu_service_readiness`	—	Computed service readiness status
`cbu_sla_commitments`	`sla`	SLA commitments per CBU
`cbu_lifecycle_instances`	`lifecycle`	Active lifecycle instances
`cbu_subscriptions`	`contract`	Contract+product subscription (onboarding gate)
`cbu_pricing_config`	`pricing-config`	NAV pricing configuration
`cbu_evidence`	`cbu`	Document/attestation evidence links

---

6) Core Entity Model

Verb domains: `entity` (22 verbs), `identifier` (11), `fund` (20), `bods` (9), `regulatory` (5)

All aggregates hang off a canonical `entities` table with a typed taxonomy in `entity_types`. Entity subtypes are modelled as separate satellite tables joined by `entity_id`.

```
erDiagram
    entity_types ||--o{ entity_types : "parent_type_id"
    entity_types ||--o{ entities : "entity_type_id"
    entities ||--o{ entity_names : "entity_id"
    entities ||--o{ entity_identifiers : "entity_id"
    entities ||--o{ entity_proper_persons : "entity_id"
    entities ||--o{ entity_funds : "entity_id"
    entities ||--o{ entity_addresses : "entity_id"
    entities ||--o{ entity_relationships : "from_entity_id"
    entities ||--o{ entity_parent_relationships : "child_entity_id"
    entities ||--o{ cbu_entity_roles : "entity_id"

    entity_types {
        uuid entity_type_id PK
        varchar name UK
        uuid parent_type_id FK
        varchar type_code
        varchar entity_category
        text_arr type_hierarchy_path
    }

    entities {
        uuid entity_id PK
        uuid entity_type_id FK
        varchar name
        varchar external_id
        varchar bods_entity_type
        date founding_date
        date dissolution_date
        boolean is_publicly_listed
        text name_norm
    }

    entity_names {
        uuid name_id PK
        uuid entity_id FK
        varchar name_type
        varchar name_value
    }

    entity_identifiers {
        uuid identifier_id PK
        uuid entity_id FK
        varchar identifier_type
        varchar identifier_value
        varchar issuing_authority
    }
    entity_proper_persons {
        uuid person_id PK
        uuid entity_id FK
        date date_of_birth
        varchar nationality
        varchar country_of_residence
    }

    entity_funds {
        uuid entity_id PK
        varchar fund_type
        varchar gleif_category
        varchar jurisdiction
        uuid parent_fund_id FK
        uuid master_fund_id FK
    }

    entity_parent_relationships {
        uuid relationship_id PK
        uuid child_entity_id FK
        uuid parent_entity_id FK
        varchar relationship_type
        varchar source
    }
```

Entity type hierarchy (self-referencing via `parent_type_id`):

Type Code	Parent	Examples
`NATURAL_PERSON`	ENTITY	Individual directors, UBOs
`LEGAL_ENTITY`	ENTITY	Corporates, funds
`LIMITED_COMPANY`	LEGAL_ENTITY	GLEIF-sourced entities
`FUND`	LEGAL_ENTITY	UCITS, AIF, hedge funds
`TRUST`	LEGAL_ENTITY	Trust vehicles
`PARTNERSHIP`	LEGAL_ENTITY	LP/GP structures
`MANCO`	LEGAL_ENTITY	Management companies

Subtype satellite tables (one per legal form — joined by `entity_id`):

Table	Legal Form	Key Columns
`entity_proper_persons`	Natural persons	date_of_birth, nationality, country_of_residence
`entity_funds`	Funds / vehicles	fund_type, gleif_category, domicile, parent_fund_id, master_fund_id
`entity_limited_companies`	Corporates	incorporation_country, share_capital

`entity_trusts`	Trusts	trust_type, governing_law
`entity_partnerships`	Partnerships	partnership_type
`entity_foundations`	Foundations	foundation_purpose
`entity_cooperatives`	Cooperatives	cooperative_type
`entity_government`	Government bodies	government_level
`entity_manco`	Management companies	manco_type, regulated_by

Supporting tables:

Table	Purpose
`entity_addresses`	Registered / operational addresses
`entity_share_classes`	Share class definitions per entity
`entity_lifecycle_events`	Lifecycle events (incorporation, dissolution)
`entity_bods_links`	Links to BODS statement IDs
`entity_concept_link`	Semantic concept associations (for entity linking)
`entity_feature`	Feature flags for ML/entity linking
`entity_relationships`	Ownership/control edges (from_entity_id → to_entity_id)

---

7) UBO & Ownership Graph

Verb domains: `ubo` (22 verbs), `ownership` (16), `control` (15)

Two layers: (1) candidate entities and proposed relationships in the client group (review workflow), and (2) promoted canonical relationships + per-CBU/case UBO assertions.

```
erDiagram
    ubo_registry }o--|| cbus : "cbu_id"
    ubo_registry }o--|| entities : "subject_entity_id"
    ubo_registry }o--|| entities : "ubo_proper_person_id"
    ubo_registry ||--o{ ubo_evidence : "ubo_id"
    ubo_snapshots }o--|| cbus : "cbu_id"
    entity_relationships }o--|| entities : "from_entity_id"
    entity_relationships }o--|| entities : "to_entity_id"

    ubo_registry {
        uuid ubo_id PK
        uuid cbu_id FK
        uuid subject_entity_id FK
        uuid ubo_proper_person_id FK
        varchar relationship_type
        varchar qualifying_reason
        numeric ownership_percentage
        varchar verification_status
        varchar screening_result
    }

    ubo_snapshots {
        uuid snapshot_id PK
        uuid cbu_id FK
        uuid case_id FK
        jsonb ubos
        jsonb ownership_chains
        jsonb control_relationships
        boolean has_gaps
    }
```

Table	Purpose
`ubo_evidence`	Document/attestation evidence for UBO assertions
`ubo_assertion_log`	Audit log of assertion results per CBU/case
`ubo_snapshot_comparisons`	Diff between two UBO snapshots (added/removed/changed)
`entity_ubos`	Legacy BODS-style UBO records per entity
`control_edges`	Control relationship edges (board, voting)
`entity_relationships_history`	Temporal history of relationship changes

---

8) Product / Service / Resource Model

Verb domains: `product` (2 verbs), `service` (3), `service-resource` (10), `service-pipeline` (14), `delivery` (3), `lifecycle` (16), `sla` (17)

Products compose services; services require resources. CBUs subscribe to products (via contracted product subscriptions), which creates service delivery obligations requiring provisioned resource instances.

```
erDiagram
    products ||--o{ product_services : "product_id"
    product_services }o--|| services : "service_id"
    services ||--o{ service_resource_capabilities : "service_id"
    service_resource_capabilities }o--|| service_resource_types : "resource_id"
    service_resource_types ||--o{ resource_dependencies : "resource_type_id"
    cbus ||--o{ cbu_resource_instances : "cbu_id"
    cbu_resource_instances }o--|| service_resource_types : "resource_type_id"

    products {
        uuid product_id PK
        varchar name UK
        varchar product_code UK
        varchar product_category
        varchar product_family
        boolean requires_kyc
    }

    services {
        uuid service_id PK
        varchar name UK
        varchar service_code UK
        varchar service_category
        text_arr lifecycle_tags
    }

    service_resource_types {
        uuid resource_id PK
        varchar name UK
        varchar resource_code UK
        varchar owner
        boolean per_market
        boolean per_currency
        boolean per_counterparty
        varchar provisioning_strategy
    }
```

Delivery & provisioning tables:

Table	Purpose
`service_intents`	What service a CBU desires (intent → delivery)
`service_delivery_map`	Actual delivery tracking (status, timeline)
`service_availability`	Three-lane service availability per booking principal
`provisioning_requests`	Resource provisioning request tracking
`provisioning_events`	Provisioning event audit trail
`resource_dependencies`	Type-level resource dependency graph
`resource_instance_dependencies`	Instance-level dependency edges
`resource_instance_attributes`	Attribute values on provisioned instances
`resource_attribute_requirements`	Required attributes per resource type

SLA tables:

Table	Purpose
`sla_templates`	SLA definition templates
`sla_measurements`	Measured SLA metrics
`sla_breaches`	Recorded SLA breaches
`cbu_sla_commitments`	SLA commitments per CBU

9) Document & Evidence Model

Verb domains: `document` (13 verbs), `requirement` (10), `attribute` (11), `docs-bundle` (3)

Two document models coexist:

- **Legacy:** `document_catalog` + `document_types` — flat catalog
- **V2 (049):** `documents` → `document_versions` — three-layer model (requirement → document → version)

```
erDiagram
    document_requirements }o--|| entities : "entity_id"
    document_requirements }o--o| documents : "document_id"
    documents ||--o{ document_versions : "document_id"
    documents }o--o| entities : "entity_id"
    documents }o--o| cbus : "cbu_id"
    attribute_registry ||--o{ attribute_observations : "attribute_id"
    attribute_observations }o--|| entities : "entity_id"

    document_requirements {
        uuid requirement_id PK
        uuid entity_id FK
        uuid document_id FK
        varchar doc_type
        varchar status
    }

    documents {
        uuid document_id PK
        uuid entity_id FK
        uuid cbu_id FK
        varchar document_type
        varchar status
    }

    document_versions {
        uuid version_id PK
        uuid document_id FK
        varchar storage_key
        varchar verification_status
        varchar rejection_code
    }

    attribute_registry {
        text id PK
        text display_name
        text category
        text value_type
        jsonb validation_rules
    }
```

Requirement state machine: `missing → requested → received → in_qa → verified` (also: `rejected → retry`, `waived`, `expired`)

Attribute dictionary tables:

Table	Purpose
`attribute_registry`	Central attribute definitions (type, validation, applicability)
`attribute_values_typed`	Typed attribute values per entity
`attribute_observations`	Observed values with source, confidence, supersession chain
`cbu_attr_values`	Attribute values per CBU (with evidence refs)
`document_attribute_links`	Policy-style proof links (document type → attribute)

---

10) Workflow Task Queue

Verb domains: `runbook` (7 verbs)

The task queue provides an async return path for long-running operations (document solicitation, human approvals).

Table	Purpose
`workflow_pending_tasks`	Outbound task tracking (emitted by workflows)
`task_result_queue`	Inbound results (ephemeral, deleted after processing)
`task_result_dlq`	Dead letter queue for failed processing
`workflow_task_events`	Permanent audit trail
`workflow_instances`	Active workflow instances
`workflow_definitions`	Workflow definitions
`staged_runbook`	Staged REPL runbook container
`staged_command`	Individual staged DSL commands
`rejection_reason_codes`	Reference data for document QA rejection reasons

---

11) Screening & KYC Support

Verb domains: `screening` (3 verbs), `kyc-agreement` (4)

The main KYC tables live in the `kyc` schema. These `ob-poc` tables support KYC integration.

Table	Purpose
`screening_lists`	Screening list definitions (sanctions, PEP)
`screening_requirements`	Screening requirements per entity type

`person_pep_status`	PEP status records per person entity
`kyc_service_agreements`	KYC service agreements between CBU and provider
`case_types`	Case type taxonomy (NEW_CLIENT, PERIODIC_REVIEW, etc.)
`risk_ratings`	Risk rating definitions

---

12) BODS + GLEIF

Verb domains: `bods` (9 verbs), `gleif` (16 verbs)

Table	Purpose
`bods_entity_statements`	Entity statements per Open Ownership standard
`bods_person_statements`	Person statements in ownership chains
`bods_ownership_statements`	Ownership/control statements linking persons to entities
`gleif_lei_records`	Cached LEI records from GLEIF API
`gleif_relationships`	Cached GLEIF relationship records (parent/child)
`gleif_sync_log`	GLEIF sync operation audit log

---

13) Reference Taxonomies

Small but load-bearing — drives interpretation, UI grouping, and rule selection.

Table	Verb Domain	Purpose
`roles`	`cbu-role-v2`	Role taxonomy (depositary, IM, director, etc.)
`role_types`	—	Role type classification
`role_categories`	—	Role category grouping
`role_applicable_entity_types`	—	Which entity types can hold which roles
`currencies`	—	Currency reference data
`master_jurisdictions`	`fund`	Jurisdiction definitions
`products`	`product`	Product catalog (CUSTODY, FUND_ACCOUNTING, etc.)
`services`	`service`	Service catalog (SAFEKEEPING, SETTLEMENT, etc.)
`regulators`	`regulatory`	Regulatory body definitions
`rule_field_dictionary`	—	Closed-world field registry for rule validation
`placeholder_kinds`	—	Placeholder entity kinds

`client_types`	—	Client type taxonomy

Schema Statistics

Metric	Count
Total `ob-poc` tables	226
Tables with DSL verb domains	~85
Tables in this document	~150 (essential to data model)
Tables omitted (DSL engine, REPL, semantic search, layout cache)	~76
DSL verb domains	57
Total verb count	~750+
Migrations	77 (001–077 + 072b)

Omitted infrastructure tables (no verb domains, not essential to data model):

- DSL engine: `dsl_verbs`, `dsl_sessions`, `dsl_instances`, `dsl_snapshots`, `dsl_*` (14 tables)
- Semantic search: `verb_pattern_embeddings`, `verb_centroids`, `semantic_match_cache`, `detected_patterns`, `intent_feedback*`
- REPL: `repl_sessions_v2`, `repl_invocation_records`
- BPMN integration: `bpmn_correlations`, `bpmn_job_frames`, `bpmn_parked_tokens`, `bpmn_pending_dispatches`, `expansion_reports`
- Session/layout: `sessions`, `session_scopes`, `session_scope_history`, `session_bookmarks`, `layout_cache`, `layout_config`
- Audit: `sheet_execution_audit`, `cbu_board_controller`, `board_control_evidence`, `cbu_control_anchors`