Documentação técnica — Banco de Dados Vetorial (RAG jurídico) com filtros por perfil das partes

Objetivo

- Definir a modelagem e o script de criação de um "banco vetorial" integrado ao seu domínio jurídico (acórdãos/decisões), suportando:
    - recuperação semântica de trechos de decisões (chunks) e seções (ementa, relatório, fundamentação, dispositivo, votos),
    - filtros estruturados por metadados ricos, incluindo características das partes: gênero, profissão, estado civil, nacionalidade, UF de origem da demanda e UF associada ao documento de identidade (emissor),
    - versionamento, multitenancy, segurança, e rastreabilidade entre o vetorial e o banco relacional principal.

Escopo e premissas

- O repositório vetorial é projetado para RAG (Retrieval-Augmented Generation) e pesquisa semântica híbrida (denso + léxico).
- Integra-se ao seu modelo relacional (já definido) via chaves canônicas, mantendo:
    - acordao_id (UUID da decisão),
    - pessoa_id (UUID da pessoa, quando aplicável),
    - parte_papel_id (requerente, requerido etc.),
    - além de metadados desnormalizados para filtros rápidos (gênero, profissão, estado civil, nacionalidade, UF de origem da demanda, UF de identidade).
- Suporte a múltiplos níveis de indexação:
    1. documento (nível grosso),
    2. seção (ementa/fundamentação/dispositivo…),
    3. chunk (janela deslizante, 256–600 tokens típicos),
    4. menções a entidades (pessoas, leis, órgãos).
- Este documento traz:
    - duas alternativas de implementação: (A) Postgres + pgvector (co-residente ao relacional) e (B) Qdrant (engine dedicada),
    - DDL de referência, índices, partições, e exemplos de consulta com filtros por perfil das partes,
    - MER em ASCII para posicionar as entidades vetoriais e sua relação com o relacional.

Requisitos funcionais (resumo)

- Inserir e versionar embeddings oriundos do JSON tratado (via seu pipeline).
- Consultar chunks relevantes por similaridade e restringir por:
    - tribunal, órgão julgador, classe/assunto, temas, data de julgamento,
    - dados das partes: gênero, profissão, estado civil, nacionalidade,
    - UF origem da demanda e UF emissora do documento (identidade).
- Habilitar busca híbrida (BM25/lexical + vetorial) e re-ranking.
- Garantir multitenancy (tenant_id) e RLS (quando aplicável).

Requisitos não-funcionais (resumo)

- Baixa latência em top-k (típico 10–50).
- Atualizações diárias/contínuas de embeddings sem reindexação cara de toda a coleção.
- Observabilidade (embedding_model, embedding_version).
- Governança de PII (limitar texto sensível e preferir IDs; atributos sensíveis em payloads controlados).

Dimensão e modelo de embeddings

- Dimensão do vetor: parametrizável (ex.: 1536 ou 3072).
- Métrica: cosine (recomendado), opcionalmente inner product.
- Armazenamento de embed_version para compatibilidade futura (migração de modelo sem downtime).

Estratégia de chunking

- Seção → chunk: janela de 256–600 tokens com stride 30–60 tokens (ajuste empírico).
- Preservar offsets: char_start/char_end e token_start/token_end.
- Tipificar chunk.type: ementa, relatório, fundamentação, dispositivo, voto, e outros.

Metadados para filtros

- Atributos de parte:
  - genero_id e genero_nome,
  - profissao_id e profissao_nome,
  - estado_civil_id e estado_civil_nome,
  - nacionalidade_pais_id e nacionalidade_pais_nome,
  - uf_origem_demanda_id e uf_origem_demanda_sigla,
  - uf_identidade_emissor_id e uf_identidade_emissor_sigla,
  - parte_papel_id e parte_papel_nome (requerente, requerido etc.).
- Atributos do acórdão:
  - tribunal_id, orgao_julgador_id, classe_id, assunto_id/temas, data_julgamento, numero_processo, turma/câmara, relator_id.
- Atributos de qualidade:
  - language, toxicity_flag, pii_flag, source_confidence.
- Versionamento/operacional:
  - embedding_model, embedding_version, index_version, created_at, updated_at, is_active, deleted_at.

Opção A — Postgres + pgvector (co-residente ao relacional) Vantagens

- Integridade referencial natural com o banco relacional.
- Uma stack operacional (backup, RLS, transações).
- Busca híbrida com GIN/TSVector e pg_trgm. Considerações
- Tuning de HNSW/IVFFLAT e manutenção de VACUUM/ANALYZE.
- Para grandes coleções, particionar por mês/tenant ou por tribunal.

DDL de referência (PostgreSQL 15+ com pgvector) Pré-requisitos:

- Extensões vector, pg_trgm, unaccent (para híbrido).

```
-- Extensões
CREATE EXTENSION IF NOT EXISTS vector;
CREATE EXTENSION IF NOT EXISTS pg_trgm;
CREATE EXTENSION IF NOT EXISTS unaccent;

-- Namespace lógico (útil para isolar coleções/escopos de embeddings)
CREATE TABLE IF NOT EXISTS vec_namespace (
  id                uuid PRIMARY KEY,
  name              text NOT NULL UNIQUE,
  description       text,
  embedding_dim     int  NOT NULL,
  metric            text NOT NULL DEFAULT 'cosine', -- 'cosine' | 'ip'
| 'l2'
  embedding_model   text NOT NULL,
  embedding_version text NOT NULL,
  created_at        timestamptz NOT NULL DEFAULT now()
);

-- Documentos (1 por acórdão/decisão)
CREATE TABLE IF NOT EXISTS vec_document (
  id                uuid PRIMARY KEY,
  tenant_id         uuid NOT NULL,
  namespace_id      uuid NOT NULL REFERENCES vec_namespace(id) ON
DELETE RESTRICT,
  acordao_id        uuid NOT NULL, -- FK lógica para o relacional
  tribunal_id       uuid,
  orgao_julgador_id uuid,
  classe_id         uuid,
  numero_processo   text,
  data_julgamento   date,
  title             text,  -- título/ementa resumida
  language          text DEFAULT 'pt',
  hash_canonical    text UNIQUE, -- para deduplicação
  is_active         boolean NOT NULL DEFAULT true,
  created_at        timestamptz NOT NULL DEFAULT now(),
  updated_at        timestamptz NOT NULL DEFAULT now()
);

-- Seções do documento (ementa, relatório, fundamentação, dispositivo,
voto)
CREATE TABLE IF NOT EXISTS vec_section (
  id                uuid PRIMARY KEY,
  document_id       uuid NOT NULL REFERENCES vec_document(id) ON
DELETE CASCADE,
  section_type      text NOT NULL, --
'ementa','relatorio','fundamentacao','dispositivo','voto','outros'
  char_start        int,
  char_end          int,
  text              text NOT NULL,
  text_lex          tsvector,      -- p/ híbrido (pré-processado)
  created_at        timestamptz NOT NULL DEFAULT now()
);

-- Chunks vetoriais (unidade de recuperação)
CREATE TABLE IF NOT EXISTS vec_chunk (
  id                          uuid PRIMARY KEY,
```

```
  tenant_id                   uuid NOT NULL,
  namespace_id                uuid NOT NULL REFERENCES
vec_namespace(id) ON DELETE RESTRICT,
  document_id                 uuid NOT NULL REFERENCES
vec_document(id) ON DELETE CASCADE,
  section_id                  uuid REFERENCES vec_section(id) ON
DELETE CASCADE,
  acordao_id                  uuid, -- referência lógica ao relacional
  chunk_seq                   int  NOT NULL,
  char_start                  int,
  char_end                    int,
  token_start                 int,
  token_end                   int,
  text                        text NOT NULL,
  text_lex                    tsvector,       -- p/ híbrido
  embedding                   vector,         -- dimensão definida no
namespace
  embedding_model             text NOT NULL,
  embedding_version           text NOT NULL,
  index_version               int  NOT NULL DEFAULT 1,
  -- Metadados de parte (desnormalizados para filtro rápido)
  parte_papel_id              uuid,
  parte_papel_nome            text,
  pessoa_id                   uuid,
  genero_id                   uuid,
  genero_nome                 text,
  profissao_id                uuid,
  profissao_nome              text,
  estado_civil_id             uuid,
  estado_civil_nome           text,
  nacionalidade_pais_id       uuid,
  nacionalidade_pais_nome     text,
  uf_origem_demanda_id       uuid,
  uf_origem_demanda_sigla    text,
  uf_identidade_emissor_id   uuid,
  uf_identidade_emissor_sigla text,
  -- Demais metadados úteis de filtro
  tribunal_id                 uuid,
  orgao_julgador_id           uuid,
  classe_id                   uuid,
  assunto_principal_id        uuid,
  assunto_principal_nome      text,
  data_julgamento             date,
  language                    text DEFAULT 'pt',
  pii_flag                    boolean DEFAULT false,
  toxicity_flag               boolean DEFAULT false,
  is_active                   boolean NOT NULL DEFAULT true,
  created_at                  timestamptz NOT NULL DEFAULT now(),
  updated_at                  timestamptz NOT NULL DEFAULT now(),
  UNIQUE (document_id, chunk_seq)
);

-- Menções a entidades (opcional, útil para perguntas do tipo "sobre
Fulano" ou "Lei X")
CREATE TABLE IF NOT EXISTS vec_entity_mention (
  id                uuid PRIMARY KEY,
  tenant_id         uuid NOT NULL,
  namespace_id      uuid NOT NULL REFERENCES vec_namespace(id) ON
DELETE RESTRICT,
  document_id       uuid NOT NULL REFERENCES vec_document(id) ON
DELETE CASCADE,
```

```
  section_id        uuid REFERENCES vec_section(id) ON DELETE CASCADE,
  acordao_id        uuid,
  entity_type       text NOT NULL,     --
'pessoa','lei','orgao','tema','precedente'
  entity_id         uuid,              -- FK lógica para relacional
(quando houver)
  entity_name       text NOT NULL,     -- forma canônica
  mention_text      text NOT NULL,     -- forma mencionada
  char_start        int,
  char_end          int,
  embedding         vector,            -- embedding da entidade/menção
(opcional)
  embedding_model   text,
  embedding_version text,
  created_at        timestamptz NOT NULL DEFAULT now()
);

-- Índices léxicos (híbrido)
CREATE INDEX IF NOT EXISTS idx_vec_section_text_lex
  ON vec_section USING GIN (text_lex);

CREATE INDEX IF NOT EXISTS idx_vec_chunk_text_lex
  ON vec_chunk USING GIN (text_lex);

-- Índices por filtros comuns
CREATE INDEX IF NOT EXISTS idx_vec_chunk_filters
  ON vec_chunk (tenant_id, namespace_id, is_active, data_julgamento);

CREATE INDEX IF NOT EXISTS idx_vec_chunk_partes
  ON vec_chunk (parte_papel_nome, genero_nome, estado_civil_nome,
profissao_nome);

CREATE INDEX IF NOT EXISTS idx_vec_chunk_ufs
  ON vec_chunk (uf_origem_demanda_sigla, uf_identidade_emissor_sigla);

-- Vetorial (HNSW ou IVFFLAT). Escolha uma estratégia conforme volume.
-- HNSW (melhor recall; custo maior em inserção e RAM):
CREATE INDEX IF NOT EXISTS hnsw_vec_chunk_embedding
  ON vec_chunk USING hnsw (embedding vector_cosine_ops)
  WITH (m = 16, ef_construction = 64);

-- Alternativa: IVFFLAT (boa para grandes coleções + lotes):
-- CREATE INDEX IF NOT EXISTS ivf_vec_chunk_embedding
--   ON vec_chunk USING ivfflat (embedding vector_cosine_ops)
--   WITH (lists = 200);

-- Opcional: RLS por tenant
-- ALTER TABLE vec_document ENABLE ROW LEVEL SECURITY;
-- CREATE POLICY tenant_isolation ON vec_document
--   USING (tenant_id = current_setting('app.tenant_id')::uuid);
-- Repetir para vec_section/vec_chunk/vec_entity_mention
```

Boas práticas operacionais no Postgres

- Particionamento:
  - Por tenant_id (hash) e/ou por data_julgamento (range mensal) em vec_chunk.
  - Ganho: manutenção e paralelismo de índices HNSW/IVF melhores.
- Atualizações de embeddings:

    - Mantenha embedding_version e index_version. Ao trocar de modelo, grave novo versionamento sem derrubar o anterior; um job de limpeza remove embeddings antigos quando seguro.
- Híbrido:
    - Popular text_lex com to_tsvector('portuguese', unaccent(text)).
    - Combinar pré-filtro léxico + similaridade vetorial melhora precisão e custo.
- VACUUM/ANALYZE:
    - Automatizar via autovacuum tunado; reindex periódico conforme churn.

Consultas de exemplo (pgvector)

1. Recuperação semântica com filtros por perfil das partes e UF:

```
-- Suponha :q_emb seja o vetor de consulta (dimension compatível)
-- Filtro: requerente mulher, profissão 'Professora', nacionalidade
brasileira,
-- UF de origem da demanda = 'SP', julgados em 2023.
SELECT
  c.id,
  c.document_id,
  c.section_id,
  c.chunk_seq,
  c.text,
  (c.embedding <=> :q_emb) AS distance,
  c.parte_papel_nome,
  c.genero_nome,
  c.profissao_nome,
  c.nacionalidade_pais_nome,
  c.uf_origem_demanda_sigla,
  c.data_julgamento
FROM vec_chunk c
WHERE c.tenant_id = :tenant
  AND c.namespace_id = :ns
  AND c.is_active = true
  AND c.parte_papel_nome = 'requerente'
  AND c.genero_nome = 'Feminino'
  AND c.profissao_nome ILIKE 'Professora%'
  AND c.nacionalidade_pais_nome = 'Brasil'
  AND c.uf_origem_demanda_sigla = 'SP'
  AND c.data_julgamento >= DATE '2023-01-01'
  AND c.data_julgamento <  DATE '2024-01-01'
ORDER BY c.embedding <=> :q_emb
LIMIT 20;
```

2. Híbrido (pré-filtro lexical por termos + top-k vetorial):

```
WITH lexical AS (
  SELECT id
  FROM vec_chunk
  WHERE tenant_id = :tenant
    AND namespace_id = :ns
    AND to_tsvector('portuguese', unaccent(text)) @@
plainto_tsquery('portuguese', unaccent(:q_text))
    AND is_active = true
  LIMIT 200
)
SELECT
```

```
  c.id, c.document_id, c.section_id, c.chunk_seq, c.text,
  (c.embedding <=> :q_emb) AS distance
FROM vec_chunk c
JOIN lexical l ON l.id = c.id
WHERE c.genero_nome IN ('Feminino','Masculino') -- ex. filtro amplo
ORDER BY c.embedding <=> :q_emb
LIMIT 20;
```

3. Consulta por seção da decisão (ex.: só fundamentação):

```
SELECT c.*
FROM vec_chunk c
JOIN vec_section s ON s.id = c.section_id
WHERE c.tenant_id = :tenant
  AND c.namespace_id = :ns
  AND s.section_type = 'fundamentacao'
ORDER BY c.embedding <=> :q_emb
LIMIT 20;
```

Opção B — Qdrant (engine vetorial dedicada) Vantagens

- Alta performance nativa com HNSW, quantização, filtros em payload.
- Escala horizontal simples, gerenciamento de collections. Considerações
- Integração referencial lógica (IDs/UUIDs), não há FK “forte” relacional.
- Backup/restore e versionamento operacionais próprios.

Especificação de collections (Qdrant)

- Collection: legal_chunks
  - Vetor principal: embedding (size = 1536 ou 3072), distance = Cosine.
  - Payload (metadados filtráveis):
    - tenant_id, namespace_id, document_id, section_id, acordao_id
    - chunk_seq, section_type, language
    - parte_papel_nome, genero_nome, profissao_nome, estado_civil_nome
    - nacionalidade_pais_nome, uf_origem_demanda_sigla, uf_identidade_emissor_sigla
    - tribunal_id, orgao_julgador_id, classe_id, assunto_principal_id
    - data_julgamento (como string ISO-8601 ou inteiro YYYYMMDD)
    - pii_flag, toxicity_flag, is_active
    - embedding_model, embedding_version, index_version
- Collection: legal_entities (opcional)
  - Vetor para nomes canônicos ou descrições de entidades.
  - Payload com entity_type, entity_name, entity_id, acordao_id/document_id.

Exemplo de criação (Qdrant API – JSON):

```
{
  "create_collection": {
    "collection_name": "legal_chunks",
    "vectors": {
      "size": 1536,
```

```
      "distance": "Cosine"
    },
    "hnsw_config": {
      "m": 16,
      "ef_construct": 64
    },
    "optimizers_config": {
      "default_segment_number": 4
    },
    "quantization_config": {
      "scalar": {
        "type": "int8",
        "always_ram": true
      }
    }
  }
}
```

Exemplo de upsert de ponto (Qdrant):

```
{
  "points": [
    {
      "id": "c9e5f1a2-...-001",
      "vector": [0.0123, -0.0456, ...],
      "payload": {
        "tenant_id": "9f3e...",
        "namespace_id": "56ac...",
        "document_id": "d1...",
        "section_id": "s1...",
        "acordao_id": "a1...",
        "chunk_seq": 12,
        "section_type": "fundamentacao",
        "text": "Trecho do chunk...",
        "parte_papel_nome": "requerente",
        "genero_nome": "Feminino",
        "profissao_nome": "Professora",
        "estado_civil_nome": "Casada",
        "nacionalidade_pais_nome": "Brasil",
        "uf_origem_demanda_sigla": "SP",
        "uf_identidade_emissor_sigla": "SP",
        "tribunal_id": "t1...",
        "orgao_julgador_id": "o1...",
        "classe_id": "c1...",
        "assunto_principal_id": "as1...",
        "data_julgamento": "2023-05-18",
        "language": "pt",
        "embedding_model": "text-embedding-3-small",
        "embedding_version": "2024-01",
        "is_active": true
      }
    }
  ]
}
```

Busca com filtros (Qdrant):

```
{
  "vector": [0.11, -0.02, ...],
  "limit": 20,
```

```
  "filter": {
    "must": [
      { "key": "tenant_id", "match": { "value": "9f3e..." } },
      { "key": "namespace_id", "match": { "value": "56ac..." } },
      { "key": "is_active", "match": { "value": true } },
      { "key": "parte_papel_nome", "match": { "value": "requerente" }
},
      { "key": "genero_nome", "match": { "value": "Feminino" } },
      { "key": "profissao_nome", "match": { "value": "Professora" } },
      { "key": "nacionalidade_pais_nome", "match": { "value": "Brasil"
} },
      { "key": "uf_origem_demanda_sigla", "match": { "value": "SP" }
},
      {
        "key": "data_julgamento",
        "range": { "gte": "2023-01-01", "lt": "2024-01-01" }
      }
    ]
  }
}
```

Integração com o relacional (chaves e rastreabilidade)

- Armazene acordao_id, document_id (derivado do acórdão), section_id e pessoa_id (quando chunk exibe narrativa relacionada a uma parte específica).
- Desnormalize para filtros rápidos: genero_nome, profissao_nome, etc., mas mantenha IDs quando existirem dimensões (para posterior join/checagem).
- Em pipelines, ao transformar JSON → relacional → vetorial:
    - vincule IDs gerados no relacional (acórdão/pessoa/parte/UF/país) e replique-os para o payload vetorial.

Segurança e privacidade

- Evite colocar identificadores sensíveis no texto do chunk. Prefira IDs e atributos categóricos.
- Se o texto original contiver PII, aplique máscara/redação no pipeline e marque pii_flag.
- Em Postgres, use RLS por tenant_id; em Qdrant, implemente gates na camada de API de busca.

Particionamento e performance

- Postgres:
    - Particionar vec_chunk por tenant_id (hash) e/ou por data_julgamento (range mensal).
    - Escolher HNSW para recall melhor; usar IVFFLAT para inserções massivas e footprint menor.
- Qdrant:
    - Ajustar ef_search por consulta (pós-filtragem) para modular recall/latência.
    - Usar quantização (int8) para reduzir memória; avaliar impacto de qualidade.

Governança de versões

- Campos embedding_model, embedding_version e index_version guiarão:
    - rotas de reprocessamento (roll-forward),
    - convivência de coleções com embeddings velhos e novos,
    - política de descarte de versões descontinuadas.

Política de atualização e deleção

- Preferir soft delete (is_active=false, deleted_at) e compactações periódicas.
- Ao atualizar um chunk (mudança de chunking/embedding), gere novo registro, mantendo o antigo até a reindexação completa.

Exemplos de ingestão (Postgres)

1. Inserir namespace:

```
INSERT INTO vec_namespace (id, name, description, embedding_dim,
metric, embedding_model, embedding_version)
VALUES (:id, :name, :desc, :dim, 'cosine', :model, :version)
ON CONFLICT (name) DO UPDATE
SET embedding_dim = EXCLUDED.embedding_dim,
    embedding_model = EXCLUDED.embedding_model,
    embedding_version = EXCLUDED.embedding_version;
```

2. Inserir documento, seção, chunk:

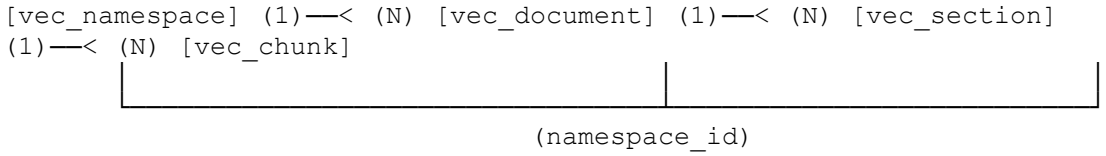
```
INSERT INTO vec_document (id, tenant_id, namespace_id, acordao_id,
tribunal_id, orgao_julgador_id,
                          classe_id, numero_processo, data_julgamento,
title, language, hash_canonical)
VALUES (:doc_id, :tenant, :ns, :acordao_id, :tribunal_id, :orgao_id,
:classe_id, :nproc, :data_julg, :title, 'pt', :hash)
ON CONFLICT (hash_canonical) DO UPDATE SET updated_at = now()
RETURNING id;

INSERT INTO vec_section (id, document_id, section_type, char_start,
char_end, text, text_lex)
VALUES (:sec_id, :doc_id, :stype, :cstart, :cend, :text,
        to_tsvector('portuguese', unaccent(:text)));

INSERT INTO vec_chunk (id, tenant_id, namespace_id, document_id,
section_id, acordao_id,
                       chunk_seq, char_start, char_end, token_start,
token_end,
                       text, text_lex, embedding, embedding_model,
embedding_version, index_version,
                       parte_papel_id, parte_papel_nome, pessoa_id,
genero_id, genero_nome,
                       profissao_id, profissao_nome, estado_civil_id,
estado_civil_nome,
                       nacionalidade_pais_id, nacionalidade_pais_nome,
                       uf_origem_demanda_id, uf_origem_demanda_sigla,
                       uf_identidade_emissor_id,
uf_identidade_emissor_sigla,
                       tribunal_id, orgao_julgador_id, classe_id,
assunto_principal_id, assunto_principal_nome,
                       data_julgamento, language, pii_flag,
toxicity_flag, is_active)
```

```
VALUES (:chunk_id, :tenant, :ns, :doc_id, :sec_id, :acordao_id,
        :seq, :cstart, :cend, :tstart, :tend,
        :text, to_tsvector('portuguese', unaccent(:text)), :emb,
:model, :emb_ver, 1,
        :papel_id, :papel_nome, :pessoa_id, :genero_id, :genero_nome,
        :prof_id, :prof_nome, :ec_id, :ec_nome,
        :pais_id, :pais_nome,
        :uf_origem_id, :uf_origem_sigla,
        :uf_id_emissor, :uf_sigla_emissor,
        :tribunal_id, :orgao_id, :classe_id, :assunto_id,
:assunto_nome,
        :data_julg, 'pt', :pii, :tox, true);
```

MER (ASCII) — visão vetorial e vínculos principais




```
[vec_chunk] --(N:1)--> [vec_document]
[vec_chunk] --(N:1)--> [vec_section]
[vec_chunk] .. contém metadados desnormalizados para filtros:
   - parte_papel_nome, genero_nome, profissao_nome, estado_civil_nome,
     nacionalidade_pais_nome, uf_origem_demanda_sigla,
uf_identidade_emissor_sigla,
     tribunal_id, orgao_julgador_id, classe_id, assunto_principal_id,
data_julgamento

[vec_entity_mention] (opcional)
   - (N:1) vec_document, (N:1) vec_section, (N:1) vec_namespace
   - entity_type, entity_id, entity_name, embedding (opcional)

Relação lógica com o relacional:
[vec_document.acordao_id] → [acordao.id]
[vec_chunk.pessoa_id] → [pessoa.id] (quando aplicável)
[vec_chunk.parte_papel_id] → [dim_parte_papel.id] (lógico)
```

Estratégia de “coverage” para perguntas de negócio

- Distribuir filtros em colunas explícitas (tipadas) + nomes desnormalizados.
- Para consultas frequentes, criar índices compostos direcionados (por ex., parte_papel_nome + genero_nome + data_julgamento).
- Materialized views (Postgres) podem acelerar painéis (ex.: perfis por UF e período).

Checklist de decisão: Postgres pgvector vs Qdrant

- Volume e QPS altos, com filtros complexos → Qdrant tende a escalar melhor nativamente.
- Co-localização com seu relacional, transações ACID, RLS integrada → Postgres + pgvector.
- Híbrido denso + léxico na mesma engine → Postgres facilita (GIN + vector).
- Equipe já domina Postgres? O overhead operacional cai.

Boas práticas finais

- Evitar “over-metadata”: priorize filtros que serão usados.
- Controlar cardinalidades: profissao_nome, assunto_principal_nome e orgao_julgador podem ter alta variedade; indexes seletivos ajudam.
- Reprocessamento: sempre popular embedding_version e manter jobs de limpeza planejados.
- Observabilidade: registrar latência por consulta, top-k, tamanho do pré-filtro lexical, ef_search (Qdrant) e uso de memória.

Se quiser, adapto este modelo:

- Para outro SGBD vetorial (Weaviate, Pinecone) com payloads equivalentes.
- Para duplicar o repositório (doc-level + chunk-level) em coleções distintas.
- Com stored procedures e funções utilitárias (Postgres) para upsert e manutenção de índices (rebuild/ANALYZE).