Tabelle (12)

Nome	Tipo	Schema
database_ref		CREATE TABLE database_ref (id INTEGER PRIMARY KEY AUTOINCREMENT, type TEXT NOT NULL, compression_level INTEGER CHECK (compression_level BETWEEN 0 AND 9), map_size INTEGER, disk_size INTEGER, extra_data TEXT -- JSON stored as TEXT)
id	INTEGER	"id" INTEGER
type	TEXT	"type" TEXT NOT NULL
compression_level	INTEGER	"compression_level" INTEGER CHECK("compression_level" BETWEEN 0 AND 9)
map_size	INTEGER	"map_size" INTEGER
disk_size	INTEGER	"disk_size" INTEGER
extra_data	TEXT	"extra_data" TEXT
manifest		CREATE TABLE manifest (-- inferred single-row manifest: we'll allow multiple manifests but expect one default row id INTEGER PRIMARY KEY AUTOINCREMENT, dsmap_bits INTEGER NOT NULL CHECK (dsmap_bits IN (8,16,32)), modalities TEXT NOT NULL -- JSON stored as TEXT)
id	INTEGER	"id" INTEGER
dsmap_bits	INTEGER	"dsmap_bits" INTEGER NOT NULL CHECK("dsmap_bits" IN (8, 16, 32))
modalities	TEXT	"modalities" TEXT NOT NULL
pretok_strategy		CREATE TABLE pretok_strategy (strategy_name TEXT PRIMARY KEY, modality TEXT, -- JSON stored as TEXT tokenizer_type TEXT, tokenizer_name TEXT, tokenizer_args TEXT, -- JSON stored as TEXT bos_token_id INTEGER, eos_token_id INTEGER, mask_token_id INTEGER, splitter_class TEXT, splitter_init TEXT, -- JSON stored as TEXT splitter_arguments TEXT, -- JSON stored as TEXT chunk_size INTEGER CHECK (chunk_size >= 0), wants_from_db TEXT, -- JSON stored as TEXT (list of required db names) returns TEXT, -- JSON stored as TEXT required_databases TEXT -- JSON stored as TEXT)
strategy_name	TEXT	"strategy_name" TEXT
modality	TEXT	"modality" TEXT
tokenizer_type	TEXT	"tokenizer_type" TEXT
tokenizer_name	TEXT	"tokenizer_name" TEXT
tokenizer_args	TEXT	"tokenizer_args" TEXT
bos_token_id	INTEGER	"bos_token_id" INTEGER
eos_token_id	INTEGER	"eos_token_id" INTEGER
mask_token_id	INTEGER	"mask_token_id" INTEGER
splitter_class	TEXT	"splitter_class" TEXT

Nome	Tipo	Schema
splitter_init	TEXT	"splitter_init" TEXT
splitter_arguments	TEXT	"splitter_arguments" TEXT
chunk_size	INTEGER	"chunk_size" INTEGER CHECK("chunk_size" >= 0)
wants_from_db	TEXT	"wants_from_db" TEXT
returns	TEXT	"returns" TEXT
required_databases	TEXT	"required_databases" TEXT
pretok_strategy_db		CREATE TABLE pretok_strategy_db (id INTEGER PRIMARY KEY AUTOINCREMENT, strategy_name TEXT NOT NULL, submdat_id INTEGER NOT NULL, is_complete INTEGER DEFAULT 0 CHECK (is_complete IN (0,1)), datatype TEXT, is_tokens INTEGER DEFAULT 0 CHECK (is_tokens IN (0,1)), is_chunks INTEGER DEFAULT 0 CHECK (is_chunks IN (0,1)), is_extra TEXT, -- JSON stored as TEXT (nullable) FOREIGN KEY (strategy_name) REFERENCES pretok_strategy(strategy_name) ON DELETE CASCADE, FOREIGN KEY (submdat_id) REFERENCES submdat(id) ON DELETE CASCADE, CHECK ((COALESCE(is_tokens,0) + COALESCE(is_chunks,0) + CASE WHEN is_extra IS NOT NULL THEN 1 ELSE 0 END) = 1))
id	INTEGER	"id" INTEGER
strategy_name	TEXT	"strategy_name" TEXT NOT NULL
submdat_id	INTEGER	"submdat_id" INTEGER NOT NULL
is_complete	INTEGER	"is_complete" INTEGER DEFAULT 0 CHECK("is_complete" IN (0, 1))
datatype	TEXT	"datatype" TEXT
is_tokens	INTEGER	"is_tokens" INTEGER DEFAULT 0 CHECK("is_tokens" IN (0, 1))
is_chunks	INTEGER	"is_chunks" INTEGER DEFAULT 0 CHECK("is_chunks" IN (0, 1))
is_extra	TEXT	"is_extra" TEXT
pretok_strategy_submdat		CREATE TABLE pretok_strategy_submdat (strategy_name TEXT NOT NULL, submdat_id INTEGER NOT NULL, finalized_at TEXT, -- ISO8601 DATETIME stored as TEXT (can be NULL) total_tokens INTEGER CHECK (total_tokens >= 0), max_tokens_per_doc INTEGER CHECK (max_tokens_per_doc >= 0), total_chunks INTEGER CHECK (total_chunks >= 0), max_chunks_per_doc INTEGER CHECK (max_chunks_per_doc >= 0), processed_docs INTEGER CHECK (processed_docs >= 0), PRIMARY KEY (strategy_name, submdat_id), FOREIGN KEY (strategy_name) REFERENCES pretok_strategy(strategy_name) ON DELETE CASCADE, FOREIGN KEY (submdat_id) REFERENCES submdat(id) ON DELETE CASCADE)
strategy_name	TEXT	"strategy_name" TEXT NOT NULL
submdat_id	INTEGER	"submdat_id" INTEGER NOT NULL
finalized_at	TEXT	"finalized_at" TEXT

Nome	Tipo	Schema
total_tokens	INTEGER	"total_tokens" INTEGER CHECK("total_tokens" >= 0)
max_tokens_per_doc	INTEGER	"max_tokens_per_doc" INTEGER CHECK("max_tokens_per_doc" >= 0)
total_chunks	INTEGER	"total_chunks" INTEGER CHECK("total_chunks" >= 0)
max_chunks_per_doc	INTEGER	"max_chunks_per_doc" INTEGER CHECK("max_chunks_per_doc" >= 0)
processed_docs	INTEGER	"processed_docs" INTEGER CHECK("processed_docs" >= 0)
pretok_strategy_submdat_precomputed_length		CREATE TABLE pretok_strategy_submdat_precomputed_length (strategy_name TEXT NOT NULL, submdat_id INTEGER NOT NULL, multiplier INTEGER NOT NULL CHECK (multiplier >= 0), precomputed_length INTEGER CHECK (precomputed_length >= 0), PRIMARY KEY (strategy_name, submdat_id, multiplier), FOREIGN KEY (strategy_name) REFERENCES pretok_strategy(strategy_name) ON DELETE CASCADE, FOREIGN KEY (submdat_id) REFERENCES submdat(id) ON DELETE CASCADE)
strategy_name	TEXT	"strategy_name" TEXT NOT NULL
submdat_id	INTEGER	"submdat_id" INTEGER NOT NULL
multiplier	INTEGER	"multiplier" INTEGER NOT NULL CHECK("multiplier" >= 0)
precomputed_length	INTEGER	"precomputed_length" INTEGER CHECK("precomputed_length" >= 0)
pretok_strategy_view_submdat_precomputed_length		CREATE TABLE pretok_strategy_view_submdat_precomputed_length (strategy_name TEXT NOT NULL, view_name TEXT NOT NULL, submdat_id INTEGER NOT NULL, multiplier INTEGER NOT NULL CHECK (multiplier >= 0), precomputed_length INTEGER CHECK (precomputed_length >= 0), PRIMARY KEY (strategy_name, view_name, submdat_id, multiplier), FOREIGN KEY (strategy_name) REFERENCES pretok_strategy(strategy_name) ON DELETE CASCADE, FOREIGN KEY (view_name) REFERENCES view(view_name) ON DELETE CASCADE, FOREIGN KEY (submdat_id) REFERENCES submdat(id) ON DELETE CASCADE)
strategy_name	TEXT	"strategy_name" TEXT NOT NULL
view_name	TEXT	"view_name" TEXT NOT NULL
submdat_id	INTEGER	"submdat_id" INTEGER NOT NULL
multiplier	INTEGER	"multiplier" INTEGER NOT NULL CHECK("multiplier" >= 0)
precomputed_length	INTEGER	"precomputed_length" INTEGER CHECK("precomputed_length" >= 0)
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
name		"name"
seq		"seq"
submdat		CREATE TABLE submdat (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT NOT NULL, ds_map_id

Nome	Tipo	Schema
		INTEGER NOT NULL, -- progressive id relating to some higher-level mapping data_type TEXT, -- JSON stored as TEXT document_number INTEGER CHECK (document_number >= 0), modality TEXT, -- JSON stored as TEXT data_key TEXT, errors_counters TEXT, -- JSON stored as TEXT (can be NULL) finalized_at TEXT -- ISO8601 DATETIME stored as TEXT (can be NULL))
id	INTEGER	"id" INTEGER
name	TEXT	"name" TEXT NOT NULL
ds_map_id	INTEGER	"ds_map_id" INTEGER NOT NULL
data_type	TEXT	"data_type" TEXT
document_number	INTEGER	"document_number" INTEGER CHECK("document_number" >= 0)
modality	TEXT	"modality" TEXT
data_key	TEXT	"data_key" TEXT
errors_counters	TEXT	"errors_counters" TEXT
finalized_at	TEXT	"finalized_at" TEXT
submdat_database		CREATE TABLE submdat_database (submdat_id INTEGER NOT NULL, database_id INTEGER NOT NULL, raw_data_bytes INTEGER CHECK (raw_data_bytes >= 0), is_data INTEGER DEFAULT 0 CHECK (is_data IN (0,1)), is_meta INTEGER DEFAULT 0 CHECK (is_meta IN (0,1)), is_extra TEXT, -- JSON stored as TEXT when extra metadata is present; NULL otherwise PRIMARY KEY (submdat_id, database_id), FOREIGN KEY (submdat_id) REFERENCES submdat(id) ON DELETE CASCADE, FOREIGN KEY (database_id) REFERENCES database_ref(id) ON DELETE CASCADE, CHECK ((COALESCE(is_data,0) + COALESCE(is_meta,0) + CASE WHEN is_extra IS NOT NULL THEN 1 ELSE 0 END) = 1))
submdat_id	INTEGER	"submdat_id" INTEGER NOT NULL
database_id	INTEGER	"database_id" INTEGER NOT NULL
raw_data_bytes	INTEGER	"raw_data_bytes" INTEGER CHECK("raw_data_bytes" >= 0)
is_data	INTEGER	"is_data" INTEGER DEFAULT 0 CHECK("is_data" IN (0, 1))
is_meta	INTEGER	"is_meta" INTEGER DEFAULT 0 CHECK("is_meta" IN (0, 1))
is_extra	TEXT	"is_extra" TEXT
view		CREATE TABLE view (view_name TEXT PRIMARY KEY, total_documents INTEGER CHECK (total_documents >= 0), shuffle_struct_format TEXT)
view_name	TEXT	"view_name" TEXT
total_documents	INTEGER	"total_documents" INTEGER CHECK("total_documents" >= 0)
shuffle_struct_format	TEXT	"shuffle_struct_format" TEXT
view_submdat		CREATE TABLE view_submdat (view_name TEXT NOT NULL, submdat_id INTEGER NOT NULL, is_skipped INTEGER DEFAULT 0 CHECK (is_skipped IN (0,1)),

Nome	Tipo	Schema
		is_preserved INTEGER DEFAULT 0 CHECK (is_preserved IN (0,1)), is_partial INTEGER DEFAULT 0 CHECK (is_partial IN (0,1)), document_number INTEGER, -- NULL if skipped or preserved, otherwise number of selected documents bytes_criteria INTEGER, -- reference if selection was based on data size PRIMARY KEY (view_name, submdat_id), FOREIGN KEY (view_name) REFERENCES view(view_name) ON DELETE CASCADE, FOREIGN KEY (submdat_id) REFERENCES submdat(id) ON DELETE CASCADE, CHECK ((COALESCE(is_skipped,0) + COALESCE(is_preserved,0) + COALESCE(is_partial, 0)) = 1), -- If skipped or preserved, document_number must be NULL; if partial, document_number must be NOT NULL CHECK (((is_skipped = 1 OR is_preserved = 1) AND document_number IS NULL) OR (is_partial = 1 AND document_number IS NOT NULL AND document_number >= 0)))
view_name	TEXT	"view_name" TEXT NOT NULL
submdat_id	INTEGER	"submdat_id" INTEGER NOT NULL
is_skipped	INTEGER	"is_skipped" INTEGER DEFAULT 0 CHECK("is_skipped" IN (0, 1))
is_preserved	INTEGER	"is_preserved" INTEGER DEFAULT 0 CHECK("is_preserved" IN (0, 1))
is_partial	INTEGER	"is_partial" INTEGER DEFAULT 0 CHECK("is_partial" IN (0, 1))
document_number	INTEGER	"document_number" INTEGER
bytes_criteria	INTEGER	"bytes_criteria" INTEGER

Indici (8)

Nome	Tipo	Schema
idx_dbref_type		CREATE INDEX idx_dbref_type ON database_ref(type)
type		"type"
idx_pretok_strategy_db_strategy		CREATE INDEX idx_pretok_strategy_db_strategy ON pretok_strategy_db(strategy_name)
strategy_name		"strategy_name"
idx_pretok_strategy_modality		CREATE INDEX idx_pretok_strategy_modality ON pretok_strategy(modality)
modality		"modality"
idx_pretok_strategy_submdat		CREATE INDEX idx_pretok_strategy_submdat ON pretok_strategy_submdat(submdat_id)
submdat_id		"submdat_id"
		CREATE INDEX idx_submdat_database_submdat ON

Nome	Tipo	Schema
idx_submdat_database_submdat		submdat_database(submdat_id)
submdat_id		"submdat_id"
idx_submdat_dsmap		CREATE INDEX idx_submdat_dsmap ON submdat(ds_map_id)
ds_map_id		"ds_map_id"
idx_submdat_name		CREATE INDEX idx_submdat_name ON submdat(name)
name		"name"
idx_view_submdat_view		CREATE INDEX idx_view_submdat_view ON view_submdat(view_name)
view_name		"view_name"

Viste (0)

Nome	Tipo	Schema

Triggers (0)

Nome	Tipo	Schema