



# トピックでAI生成したデータの保存先

---

 **ステータス:** アクティブ（実装仕様書）

 **最終更新:** 2025-12-11

 **用途:** トピックでAI生成したデータの保存先とベクトル（埋め込み）の保存先の説明

## 概要

個別トピックでAIにエンティティ、リレーション、メタデータを生成させた場合の保存先と、ベクトル（埋め込み）の保存先を説明します。

## データの保存先

### 1. トピックメタデータ（キーワード、カテゴリ、重要度など）

**保存先:** SQLite `meetingNotes` テーブル内のトピックデータ

**保存タイミング:** トピック保存時（「保存」ボタンをクリック）

**保存内容:**

- `keywords`: キーワード配列
- `semanticCategory`: セマンティックカテゴリ
- `importance`: 重要度
- `summary`: サマリー

**確認方法:** トピック編集画面で確認可能

---

### 2. エンティティ

**保存先:** SQLite `entities` テーブル

**保存タイミング:** トピック保存時（エンティティが生成されている場合）

**保存内容:**

- `id`: エンティティID (`entity_${timestamp}_${random}`形式)
- `name`: エンティティ名
- `type`: エンティティタイプ（会社、人物、製品など）
- `organizationId`: 組織ID
- `metadata.topicId`: 関連するトピックID
- `aliases`: 別名配列
- `metadata`: その他のメタデータ

**保存処理:** `createEntity()` 関数経由

- Rust API経由で保存を試行
- 失敗時はTauriコマンド経由でSQLiteに保存

#### 確認方法:

- ナレッジグラフページで確認可能
  - SQLiteの`entities`テーブルを直接確認
- 

### 3. リレーション

**保存先:** SQLite `relations` テーブル (`topicRelations`からリネーム済み)

**保存タイミング:** トピック保存時 (リレーションが生成されている場合)

#### 保存内容:

- `id`: リレーションID (`relation_${timestamp}_${random}`形式)
- `topicId`: 関連するトピックID
- `sourceEntityId`: 起点エンティティID
- `targetEntityId`: 終点エンティティID
- `relationType`: リレーションタイプ (子会社、出資、雇用など)
- `description`: 説明
- `organizationId`: 組織ID
- `confidence`: 信頼度

**保存処理:** `createRelation()` 関数経由

- Rust API経由で保存を試行
- 失敗時はTauriコマンド経由でSQLiteに保存


#### 確認方法:

- ナレッジグラフページで確認可能
  - SQLiteの`relations`テーブルを直接確認
- 

### 4. ベクトル (埋め込み) の保存先

#### 4-1. トピック埋め込み

**保存先 (優先順位) :**

1. **ChromaDB** (`useChromaDB`が`true`の場合)  **実装済み**
  - コレクション名: `topics_{organizationId}`
  - ID形式: トピックID (SQLiteの`topics.topicId`と同じ)
  - 保存内容:
    - `embedding`: 統合埋め込みベクトル (1536次元)
    - `metadata`: メタデータ (JSON形式)
      - `topicId`: トピックID
      - `meetingNoteId`: 議事録ID
      - `organizationId`: 組織ID

- **title**: タイトル
- **content**: コンテンツ
- **semanticCategory**: セマンティックカテゴリ
- **keywords**: キーワード (JSON文字列)
- **tags**: タグ (JSON文字列)
- **embeddingModel**: 使用モデル
- **embeddingVersion**: 埋め込みバージョン

## 2. SQLite (ChromaDBが有効な場合)

- テーブル名: **topics** (**topicEmbeddings**から統合済み)
- 保存内容: **メタデータのみ** (ベクトルデータは保存しない)
- **chromaSynced**: 同期状態 (0: 未同期、1: 同期済み)

## 3. SQLite (ChromaDBが無効な場合)

- テーブル名: **topics**
- 保存内容: **メタデータのみ** (ベクトルデータは保存しない)
- **注意**: トピック検索にはSQLiteフォールバックはありません。ChromaDBが無効な場合、検索結果は空になります。

**保存タイミング**: トピック保存時 (**saveTopicEmbeddingAsync()**が自動実行)

---

## 4-2. エンティティ埋め込み

**保存先 (優先順位) :**

### 1. ChromaDB (**useChromaDB**が**true**の場合)

- コレクション名: **entities\_{organizationId}**
- ID: エンティティID
- 保存内容:
  - **combinedEmbedding**: 統合埋め込みベクトル (1536次元)
  - **nameEmbedding**: 名前埋め込み
  - **metadataEmbedding**: メタデータ埋め込み (メタデータがある場合)
  - **metadata**: メタデータ (JSON形式)

### 2. SQLite (ChromaDBが有効な場合)

- テーブル名: **entities** (**entityEmbeddings**は使用されていません)
- 保存内容: **メタデータのみ** (ベクトルデータは保存しない)
- **chromaSynced**: 同期状態 (0: 未同期、1: 同期済み)

### 3. SQLite (ChromaDBが無効な場合)

- テーブル名: **entities**
- 保存内容: **メタデータのみ** (ベクトルデータは保存しない)
- **注意**: **entityEmbeddings**テーブルは現在は使用されていません

**保存タイミング:** エンティティ作成時 (`createEntity()`内で`saveEntityEmbeddingAsync()`が自動実行)

---

#### 4-3. リレーション埋め込み

**保存先 (優先順位) :**

1. **ChromaDB** (`useChromaDB`が`true`の場合)

- コレクション名: `relations_{organizationId}`
- ID: リレーションID
- 保存内容:
  - `combinedEmbedding`: 統合埋め込みベクトル (1536次元)
  - `descriptionEmbedding`: 説明埋め込み
  - `relationTypeEmbedding`: リレーションタイプ埋め込み
  - `metadata`: メタデータ (JSON形式)

2. **SQLite** (ChromaDBが有効な場合)

- テーブル名: `relations` (`relationEmbeddings`は使用されていません)
- 保存内容: **メタデータのみ** (ベクトルデータは保存しない)
- `chromaSynced`: 同期状態 (0: 未同期、1: 同期済み)

3. **SQLite** (ChromaDBが無効な場合)

- テーブル名: `relations`
- 保存内容: **メタデータのみ** (ベクトルデータは保存しない)
- **注意:** `relationEmbeddings`テーブルは現在は使用されていません

**保存タイミング:** リレーション作成時 (`createRelation()`内で`saveRelationEmbeddingAsync()`が自動実行)

---

#### 保存フロー (トピック保存時)

1. トピックを保存  
↓
2. トピックメタデータをSQLiteのmeetingNotesテーブルに保存  
↓
3. トピック埋め込みを生成・保存 (非同期)
  - ChromaDBにベクトルを保存
  - SQLiteにメタデータのみを保存  
↓
4. 【上書き保存】既存のエンティティとリレーションを削除 (非同期)
  - このトピックに関連する既存のリレーションを削除
  - このトピックに関連する既存のエンティティを削除  
↓
5. 新しいエンティティを保存 (非同期)
  - SQLiteのentitiesテーブルに保存

- エンティティ埋め込みを生成・保存（非同期）
- ChromaDBにベクトルを保存
- SQLiteにメタデータのみを保存

↓

6. 新しいリレーションを保存（非同期）
  - SQLiteの`relations`テーブルに保存（`topicRelations`からリネーム済み）
  - リレーション埋め込みを生成・保存（非同期）
  - ChromaDBにベクトルを保存
  - SQLiteにメタデータのみを保存

## 上書き保存の仕組み

トピック保存時は、**上書き保存**が行われます：

1. **既存データの削除**: そのトピックに関連する既存のエンティティとリレーションを削除
2. **新規データの保存**: AI生成された新しいエンティティとリレーションを保存

これにより、前回のデータが残らず、常に最新のAI生成結果が保存されます。

---

## 確認方法

### 1. トピックメタデータの確認

- トピック編集画面で確認
- SQLiteの`meetingNotes`テーブルを確認

### 2. エンティティの確認

- ナレッジグラフページで確認
- SQLiteの`entities`テーブルを確認

### 3. リレーションの確認

- ナレッジグラフページで確認
- SQLiteの`relations`テーブルを確認（`topicRelations`からリネーム済み）

### 4. 埋め込みベクトルの確認

- RAG検索ページの「 埋め込み統計」ボタンで確認
- ブラウザコンソールで`window.printEmbeddingStats()`を実行

---

## 注意事項

1. **非同期処理**: 埋め込み生成は非同期で実行されるため、保存直後はまだ生成中の場合があります
2. **エラー処理**: 埋め込み生成に失敗しても、トピック・エンティティ・リレーションは保存されます
3. **ChromaDB使用時**: ベクトルデータはChromaDBに保存され、SQLiteにはメタデータのみが保存されます

4. **organizationId必須**: 埋め込み生成には**organizationId**が必要です。設定されていない場合は埋め込みが生成されません
5. **テーブル名の変更**:
  - **topicRelations** → **relations** (リネーム済み)
  - **topicEmbeddings** → **topics** (統合済み)
  - **entityEmbeddings**と**relationEmbeddings**は現在は使用されていません
6. **トピック検索**: トピック検索にはSQLiteフォールバックはありません。ChromaDBが無効な場合、検索結果は空になります

## 関連ドキュメント

- [埋め込みベクトルの保存場所](#) - 埋め込みベクトルの保存場所の詳細
- [データベース設計](#) - SQLiteデータベースの設計
- [ChromaDB統合計画](#) - ChromaDB統合の実装計画