

# THIẾT KẾ DATABASE Blog & Video Platform

NGÀY 4-7: THIẾT KẾ DATABASE SCHEMA  
Complete Database Design Document

<b>Phiên bản:</b>	1.0
<b>Ngày tạo:</b>	27/11/2025
<b>Tác giả:</b>	Claude & Development Team
<b>Số bảng:</b>	16 tables
<b>Số triggers:</b>	15+ triggers

# MỤC LỤC

<b>1.</b>	ERD - Entity Relationship Diagram	4
<b>2.</b>	CHI TIẾT 16 BẢNG CHÍNH	6
	2.1 users - Tài khoản người dùng	6
	2.2 posts - Bài viết blog	7
	2.3 videos - Video files	8
	2.4 categories - Danh mục	9
	2.5 tags - Tags	9
	2.6 comments - Bình luận	10
	2.7 likes - Lượt thích	11
	2.8 bookmarks & folders - Lưu bài viết	11
	2.9 sessions - Phiên đăng nhập	12
	2.10 Analytics tables - Thống kê	13
<b>3.</b>	INDEXES - Tối ưu hóa truy vấn	15
<b>4.</b>	TRIGGERS - Tự động hóa	17
<b>5.</b>	MIGRATION SCRIPT	20
<b>6.</b>	DATA DICTIONARY	23

# 1. ERD - ENTITY RELATIONSHIP DIAGRAM

Database schema bao gồm 16 bảng chính được tổ chức theo các nhóm chức năng:

- Core Entities: users, posts, videos
- Content Organization: categories, tags, post\_categories, post\_tags
- User Engagement: comments, likes, bookmarks, bookmark\_folders
- Authentication: sessions
- Analytics: post\_views, video\_views
- Admin & Logs: activity\_logs, search\_queries

Key Relationships:

- users (1) → (N) posts: Một user tạo nhiều bài viết
- posts (1) → (0..1) videos: Mỗi post tối đa 1 video
- posts (N) ↔ (N) categories: Many-to-many qua post\_categories
- posts (N) ↔ (N) tags: Many-to-many qua post\_tags (max 5 tags)
- posts (1) → (N) comments: 1-level reply support
- users (1) → (N) bookmarks → (N) posts: Organized in folders

Đặc điểm kỹ thuật:

- Sử dụng UUID cho tất cả primary keys
- Soft delete với deleted\_at timestamp
- Denormalized counters (view\_count, like\_count) cho performance
- Vector embeddings (1536 dimensions) cho AI semantic search
- Triggers tự động update counters và timestamps
- Indexes được tối ưu cho các query patterns phổ biến

Parent Table	Child Table	Type	On Delete
users	posts	1:N	CASCADE
users	comments	1:N	CASCADE
users	likes	1:N	CASCADE
users	bookmarks	1:N	CASCADE
posts	videos	1:0..1	SET NULL
posts	comments	1:N	CASCADE
posts	likes	1:N	CASCADE
posts	post_views	1:N	CASCADE
videos	video_views	1:N	CASCADE
comments	comments	1:N	CASCADE (reply)
bookmark_folders	bookmarks	1:N	CASCADE

## 2. CHI TIẾT 16 BẢNG CHÍNH

### 2.1 Bảng USERS

Mục đích: Lưu trữ tài khoản người dùng và thông tin xác thực

Primary Key: id (UUID)

Indexes: email, username, email\_verified, created\_at

Các trường quan trọng:

- email, username: UNIQUE, dùng để đăng nhập
- password\_hash: Bcrypt hash (cost factor 12)
- email\_verified: Phải TRUE mới được tạo bài viết (BR-01)
- spam\_score: Tăng khi bị báo cáo spam, auto-block ở 5+
- social\_links: JSONB chứa links mạng xã hội
- deleted\_at: Soft delete timestamp

Constraints:

- username: Chỉ chấp nhận a-z, 0-9, \_, -
- email: Format email hợp lệ
- spam\_score: 0-100 range

```
CREATE TABLE users (  
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),  
  email VARCHAR(255) NOT NULL UNIQUE,  
  username VARCHAR(50) NOT NULL UNIQUE,  
  password_hash VARCHAR(255) NOT NULL,  
  full_name VARCHAR(100),  
  bio TEXT,  
  avatar_url VARCHAR(500),  
  social_links JSONB DEFAULT '{}',  
  email_verified BOOLEAN DEFAULT FALSE,  
  is_active BOOLEAN DEFAULT TRUE,  
  is_admin BOOLEAN DEFAULT FALSE,  
  spam_score INTEGER DEFAULT 0,  
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
  deleted_at TIMESTAMP  
);
```

### 2.2 Bảng POSTS

Mục đích: Bài viết blog với rich text content và metadata

Primary Key: id (UUID)

Foreign Keys: author\_id → users, video\_id → videos

Các trường quan trọng:

- title: 10-200 ký tự (BR-02)
- slug: URL-friendly, unique per author (auto-generated)
- content: HTML content, min 50 chars (BR-02)
- status: draft, published, archived

- visibility: public, private, unlisted
- embedding: vector(1536) cho AI semantic search
- \*\_count: Denormalized counters (updated by triggers)

Indexes đặc biệt:

- Full-text search: GIN index trên title + content
- Vector search: IVFFlat index cho similarity search
- Unique constraint: (author\_id, slug)

## 2.3 Bảng VIDEOS

Mục đích: Video files với encoding status và HLS URLs

Primary Key: id (UUID)

Foreign Key: post\_id → posts

Processing Workflow:

- Status: uploading → processing → ready/failed
- Max file size: 2GB (BR-02)
- Max duration: 30 minutes (BR-02)
- Retry limit: 3 attempts (BR-04)

Các trường quan trọng:

- status: uploading, processing, ready, failed, cancelled
- raw\_file\_path: MinIO path to original (deleted after encode)
- hls\_master\_url: Path to master.m3u8 playlist
- available\_qualities: JSONB array ["1080p", "720p", ...]
- retry\_count: Auto-increment on failure (BR-04)
- error\_message: Log của lỗi cuối cùng

## 2.4 Tổng quan các bảng còn lại

Bảng	Mô tả	Key Features
categories	Danh mục bài viết	Unique name, color coding, post_count
tags	Tags nhiều-nhiều	Unique name, usage_count, trigram search
post_categories	Junction table	Many-to-many posts ↔ categories
post_tags	Junction table	Many-to-many posts ↔ tags (max 5)
comments	Bình luận	1-level reply, sensitive words flagging
likes	Lượt thích	Toggle on/off, unique per user+post
bookmark_folders	Thư mục bookmark	User-created, default "Read Later"
bookmarks	Lưu bài viết	Organized in folders, unique per user+post
sessions	Phiên đăng nhập	JWT tokens, auto-expire, device tracking
post_views	Lượt xem bài viết	Session-based, unique per day
video_views	Lượt xem video	Watch duration, completion rate

activity_logs	Audit trail	Admin actions, JSONB metadata
search_queries	Log tìm kiếm	Analytics cho search patterns

## 3. INDEXES - TỐI ƯU HÓA TRUY VẤN

Database có 60+ indexes được thiết kế để tối ưu hóa các query patterns phổ biến. Indexes được nhóm theo loại và mục đích sử dụng.

Loại Index	Mục đích	Ví dụ	Query Pattern
B-tree (Default)	Lookup nhanh, range queries	idx_users_email	WHERE email = ?
Partial Index	Index chỉ subset data	WHERE deleted_at IS NULL	Loại bỏ soft-deleted
GIN	Full-text search, JSONB	idx_posts_fulltext	to_tsvector search
IVFFlat	Vector similarity	idx_posts_embedding	pgvector cosine distance
Trigram (GIN)	Fuzzy text search	idx_tags_name_trgm	LIKE "%react%"
Composite	Multi-column queries	(user_id, post_id)	Unique constraints

### 3.1 Các Indexes Quan Trọng Nhất

#### 1. Posts Fulltext Search

```
CREATE INDEX idx_posts_fulltext ON posts
USING GIN (to_tsvector('english', title || ' ' || content));
```

→ Query: WHERE to\_tsvector(...) @@ plainto\_tsquery('react hooks')

#### 2. Posts Vector Similarity

```
CREATE INDEX idx_posts_embedding ON posts
USING ivfflat (embedding vector_cosine_ops) WITH (lists = 100);
```

→ Query: ORDER BY embedding <=> \$query\_embedding LIMIT 10

#### 3. Post Views Unique Constraint

```
UNIQUE (post_id, session_id, DATE(viewed_at))
```

→ Đảm bảo mỗi session chỉ count 1 view/ngày (BR-06.2)

#### 4. Partial Indexes cho Soft Delete

```
CREATE INDEX idx_users_email ON users(email) WHERE deleted_at IS NULL;
```

→ Loại bỏ deleted records khỏi index, tiết kiệm space

#### 5. Composite Index cho Unique Slug

```
CONSTRAINT unique_author_slug UNIQUE (author_id, slug)
```

→ Slug unique trong scope của user (BR-03)

## 4. TRIGGERS - TỰ ĐỘNG HÓA

Database có 15+ triggers để tự động hóa các tác vụ: • Auto-update timestamps • Increment/decrement counters • Generate slugs và excerpts • Validate business rules

Trigger	Loại	Mục đích	Áp dụng cho
update_updated_at	BEFORE UPDATE	Auto-update updated_at	4 tables
increment_*_count	AFTER INSERT	Tăng counters	likes, comments, tags
decrement_*_count	AFTER DELETE	Giảm counters	likes, comments, tags
generate_post_slug	BEFORE INSERT	Auto slug từ title	posts
generate_excerpt	BEFORE INSERT	Auto excerpt từ content	posts
validate_comment_depth	BEFORE INSERT	Chặn reply-to-reply	comments

### 4.1 Ví dụ Trigger: Auto-generate Slug

```
CREATE OR REPLACE FUNCTION generate_post_slug()
RETURNS TRIGGER AS $$
DECLARE
    base_slug TEXT;
    final_slug TEXT;
    counter INTEGER := 1;
BEGIN
    -- Convert title to slug
    base_slug := lower(regexp_replace(NEW.title, '^[a-zA-Z0-9\s-]', '', 'g'));
    base_slug := regexp_replace(base_slug, '\s+', '-', 'g');
    base_slug := trim(both '-' from base_slug);

    final_slug := base_slug;

    -- Check uniqueness
    WHILE EXISTS (
        SELECT 1 FROM posts
        WHERE author_id = NEW.author_id
        AND slug = final_slug
        AND deleted_at IS NULL
    ) LOOP
        final_slug := base_slug || '-' || counter;
        counter := counter + 1;
    END LOOP;

    NEW.slug := final_slug;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

### 4.2 Lợi ích của Triggers

- ✓ Consistency: Counter luôn chính xác, không cần manual update
- ✓ Performance: Denormalized data sẵn sàng, không cần JOIN COUNT
- ✓ Business Rules: Enforce tự động (slug unique, 1-level comment)
- ✓ Developer Experience: Logic tập trung ở DB, không scatter trong code

⚠ Trade-offs:

- Phức tạp hơn khi debug
- Tốn overhead cho write operations
- Cần test kỹ để tránh infinite loops

## 5. MIGRATION SCRIPT

File migration SQL đầy đủ: 001\_initial\_schema.sql

Tổng cộng: ~600 dòng SQL bao gồm:

- CREATE TABLE statements (16 tables)
- CREATE INDEX statements (60+ indexes)
- CREATE TRIGGER statements (15+ triggers)
- Seed data (categories, tags)

Cách chạy migration:

```
# Step 1: Create database
createdb blog_platform

# Step 2: Enable extensions
psql blog_platform -c "CREATE EXTENSION IF NOT EXISTS \"uuid-osspl\""
psql blog_platform -c "CREATE EXTENSION IF NOT EXISTS \"pgvector\""
psql blog_platform -c "CREATE EXTENSION IF NOT EXISTS \"pg_trgm\""

# Step 3: Run migration
psql blog_platform -f 001_initial_schema.sql

# Step 4: Verify
psql blog_platform -c "SELECT tablename FROM pg_tables WHERE schemaname = 'public'"
psql blog_platform -c "SELECT count(*) FROM pg_indexes WHERE schemaname = 'public'"
```

### 5.1 Migration Checklist

- ☐ Pre-migration:
  - Backup existing database (nếu có)
  - Check PostgreSQL version ( $\geq 14$  recommended)
  - Install pgvector extension
- ☐ During migration:
  - Run in transaction (BEGIN; ... COMMIT;)
  - Monitor for errors
  - Verify row counts sau khi seed data
- ☐ Post-migration:
  - Test all triggers
  - Verify indexes created
  - Run ANALYZE để update statistics
  - Test sample queries
  - Setup regular VACUUM ANALYZE cron job

## 6. DATA DICTIONARY

Data dictionary chi tiết cho tất cả 16 bảng với mô tả từng column. Đây là tài liệu tham khảo cho developers khi code.

### 6.1 Bảng USERS

Column	Type	Null	Default	Description
id	UUID	NO	gen_random_uuid()	Primary key
email	VARCHAR(255)	NO		Email login (unique)
username	VARCHAR(50)	NO		Username (a-z0-9_-)
password_hash	VARCHAR(255)	NO		Bcrypt hash
full_name	VARCHAR(100)	YES	NULL	Display name
bio	TEXT	YES	NULL	User bio
avatar_url	VARCHAR(500)	YES	NULL	Profile pic URL
social_links	JSONB	NO	{}	Social media links
email_verified	BOOLEAN	NO	FALSE	Email verified?
is_active	BOOLEAN	NO	TRUE	Account active?
is_admin	BOOLEAN	NO	FALSE	Admin privileges?
spam_score	INTEGER	NO	0	Spam count (0-100)
created_at	TIMESTAMP	NO	NOW()	Created time
updated_at	TIMESTAMP	NO	NOW()	Updated time
deleted_at	TIMESTAMP	YES	NULL	Soft delete

Lưu ý: Data dictionary đầy đủ cho tất cả 16 bảng có trong file 001\_initial\_schema.sql với COMMENT ON statements. Truy vấn dictionary:

```
SELECT table_name, column_name, data_type, is_nullable
FROM information_schema.columns
WHERE table_schema = 'public' ORDER BY table_name, ordinal_position;
```

# TÓM TẮT & NEXT STEPS

Đã hoàn thành:

- ✓ ERD chi tiết với 16 bảng
- ✓ SQL schemas với constraints đầy đủ
- ✓ 60+ indexes cho performance
- ✓ 15+ triggers cho automation
- ✓ Migration script sẵn sàng deploy
- ✓ Data dictionary hoàn chỉnh

Metrics:

- Tổng số bảng: 16 tables
- Tổng số indexes: 60+ indexes
- Tổng số triggers: 15+ triggers
- Dòng SQL: ~600 lines
- Extensions: uuid-oss, pgvector, pg\_trgm

Next Steps (Tuần 1-3):

1. Setup Docker Compose với PostgreSQL 16
2. Run migration script
3. Setup Prisma ORM
4. Generate TypeScript types từ schema
5. Implement Repository pattern
6. Write integration tests

Files quan trọng:

- 001\_initial\_schema.sql - Migration script
- database\_design.md - Markdown documentation
- Ngay\_4-7\_Thiet\_Ke\_Database.pdf - PDF này