Database Schema Analysis - Writgo Architectuur Vereenvoudiging

Huidige Database Structuur

Clients Tabel (Bestaand)

Gebaseerd op de API routes en type definitions:

```typescript
interface Client {
  id: string;
  name: string;
  email: string;
  password: string; // hashed
  phone: string | null;
  companyName: string | null;
  website: string | null;

  // Buffer/Social Media (Legacy?)
  bufferEmail: string | null;
  bufferConnected: boolean;
  bufferConnectedAt: Date | null;

  // Credits System
  subscriptionCredits: number;
  topUpCredits: number;
  isUnlimited: boolean;

  // Subscription
  subscriptionPlan: string | null; // INSTAPPER, STARTER, GROEI, DOMINANT
  subscriptionStatus: string | null;

  // Status
  isActive: boolean;
  automationActive: boolean;
  onboardingCompleted: boolean;

  // Timestamps
  createdAt: Date;
  updatedAt: Date;

  // Relations
  ClientSubscription?: ClientSubscription | null;
  _count?: {
    contentPieces: number;
  };
}
```

Client Subscriptions Tabel (Nieuw - uit database.types.ts)

```
interface ClientSubscription {
  id: string;
  client_id: string;
  package_type: PackageType; // 'INSTAPPER' | 'STARTER' | 'GROEI' | 'DOMINANT'
  price: number;
  pillar_articles: number;
  cluster_articles: number;
  social_posts: number;
  videos: number;
  start_date: Date;
  end_date?: Date | null;
  active: boolean;
  created_at: Date;
  updated_at: Date;
}
```

Connected Platforms Tabel (Nieuw - uit database.types.ts)

```
interface ConnectedPlatform {
  id: string;
  client_id: string;
  platform_type: PlatformType; // linkedin, instagram, facebook, twitter, etc.
  platform_name: string;
  access_token?: string | null;
  refresh_token?: string | null;
  token_expiry?: Date | null;
  platform_user_id?: string | null;
  platform_username?: string | null;
  connected_at: Date;
  active: boolean;
  metadata: Record<string, any>; // Voor WordPress URL, extra settings, etc.
  created_at: Date;
  updated_at: Date;
}
```

Content Delivery Tabel (Nieuw - uit database.types.ts)

```
interface ContentDelivery {
  id: string;
  client_id: string;
  subscription_id?: string | null;
  content_type: ContentType; // 'pillar' | 'cluster' | 'social' | 'video'
  title: string;
  content?: string | null;
  status: ContentStatus; // 'draft' | 'scheduled' | 'published' | 'failed'
  scheduled_date?: Date | null;
  published_date?: Date | null;
  platform_ids: string[]; // Array van connected_platform IDs
  impressions: number;
  engagements: number;
  clicks: number;
  metadata: Record<string, any>;
  created_at: Date;
  updated_at: Date;
}
```

Probleem: Dubbele Structuur?

Uit de analyse blijkt:
1. **Clients tabel heeft al veel velden** voor social media en subscriptions
2. **Er zijn ook aparte tabellen** (client_subscriptions, connected_platforms, content_delivery)
3. Deze lijken niet allemaal gebruikt te worden in de huidige code

Vereenvoudigde Architectuur - Wat Moet Er Gebeuren?

✅ GOED: Connected Platforms direct aan Client

De `connected_platforms` tabel is al goed ontworpen:
- Direct gekoppeld aan `client_id` (NIET via projects!)
- Ondersteunt meerdere platforms per client
- Heeft metadata voor WordPress URL en andere settings
- Perfect voor ons businessmodel

⚠️ PROBLEEM: Content Creation Flow

Huidige situatie onduidelijk:
- Waar wordt content momenteel opgeslagen? In `contentPieces` of `content_delivery`?
- Hoe worden platforms nu gekoppeld aan content?
- Gebruikt de app de nieuwe `content_delivery` tabel of een oude structuur?

Wat we nodig hebben:
1. Content wordt aangemaakt door admin
2. Content wordt gekoppeld aan 1 klant
3. Content wordt gedistribueerd naar WordPress + alle connected platforms van die klant

⚠️ PROBLEEM: Project Layer

Vraag: Bestaat er nog een `projects` tabel?
- Geen Project interface gevonden in lib/types.ts
- Geen project API routes gevonden
- Maar: `contentPieces` relation suggereert dat er nog een oude structuur is

Implementatie Strategie

Fase 1: Database Audit ✅

We moeten eerst vaststellen:
1. Welke tabellen bestaan er ECHT in Supabase?
2. Welke worden ACTIEF gebruikt in de API routes?
3. Zijn `client_subscriptions`, `connected_platforms`, `content_delivery` al aangemaakt?

Fase 2: Migratiestrategie Bepalen

Afhankelijk van audit:

Scenario A: Nieuwe tabellen bestaan al
- Gebruik `connected_platforms` voor alle platform integraties
- Migreer oude `bufferEmail`, `bufferConnected` velden (legacy cleanup)
- Update API routes om nieuwe tabellen te gebruiken

Scenario B: Nieuwe tabellen bestaan niet
- Maak `connected_platforms` tabel aan
- Maak `content_delivery` tabel aan (of gebruik bestaande `contentPieces` als basis)
- Migreer data

Fase 3: Admin Interface Update

- Klanten pagina: toon connected platforms uit `connected_platforms` tabel
- Klanten aanmaken: gebruik subscriptionPlan field voor pakket selectie
- Platforms management: direct gekoppeld aan client

Fase 4: Client Dashboard Update

- Platforms pagina: lijst van connected platforms
- Onderscheid WordPress (1x, required) vs Social Media (multiple, based on package)
- Content kalender: toon content uit `content_delivery` tabel

Fase 5: Content Creation Flow

- Admin selecteert klant
- Content wordt aangemaakt in `content_delivery` tabel
- `platform_ids` field bevat IDs van alle platforms waar het naartoe moet
- GetLate.dev integratie gebruikt deze platform_ids voor distributie

Database Audit Resultaten ✅

Bestaande Tabellen in Supabase:

- ✅ `client` - client accounts
- ✅ `project` - projects per client (1:N relatie!)
- ✅ `socialMediaAccount` - social media verbindingen
- ✅ `contentPiece` - content opslag
- ✅ `scheduledPost` - geplande posts
- ✅ `socialMediaPost` - social media posts
- ✅ `clientAISettings` - AI settings
- ... en 80+ andere tabellen

Project Tabel Velden:

```
{
  id: string;
  clientId: string;
  name: string;
  websiteUrl: string;
  description: string | null;
  targetAudience: string | null;
  brandVoice: string | null;
  niche: string | null;
  keywords: string[];
  contentPillars: string[];
  writingStyle: string | null;
  customInstructions: string | null;
  isPrimary: boolean;
  isActive: boolean;
  sitemap: any;
  sitemapScannedAt: Date | null;
}
```

🚨 KRITISCHE BEVINDING

120+ API routes gebruiken `prisma.project`!
- Content generatie
- WordPress integratie
- Social media posting
- Keyword research
- Site analysis
- En veel meer...

Herziening: Implementatie Strategie

❌ NIET HAALBAAR: Project Layer Verwijderen

Te veel afhankelijkheden, zou 120+ API routes breken.

✅ NIEUWE STRATEGIE: Auto-Project + UI Simplificatie

Concept: "Invisible Project Layer"
- Project layer blijft technisch bestaan (backend)
- Maar wordt **volledig verborgen** in UI (frontend)
- Client ziet alleen hun bedrijfsinfo, geen "projecten"
- Admin ziet één "bedrijf" per klant, geen project-lijst

Implementatie:

1. **Auto-Create Default Project**
   - Bij client creation: automatisch 1 project aanmaken
   - Project name = company name
   - Project website = client website
   - isPrimary = true

2. **Admin Interface Vereenvoudigen**
   - "Projecten" navigatie item VERWIJDEREN
   - "Klanten" pagina toont bedrijfsinfo + WordPress URL

- Bij “Klanten” bewerken: update zowel client als default project
- Client creation formulier: velden gaan naar zowel client als project

3. **Client Dashboard Vereenvoudigen**
   - Geen “Projecten” keuze/selector
   - “Instellingen” pagina toont bedrijfsinfo (komt uit project)
   - WordPress URL, brand voice, keywords = uit default project
   - Client kan deze bewerken zonder te weten dat het een “project” is

4. **API Routes Aanpassen**
   - Waar nu `req.body.projectId` gevraagd wordt:
     - Automatisch default project van client ophalen
     - `const project = await prisma.project.findFirst({ where: { clientId, isPrimary: true } })`
     - Geen breaking changes aan 120+ routes nodig

5. **Platform Koppeling**
   - Social media accounts blijven gekoppeld aan project
   - Maar in UI: “jouw verbonden platforms” (geen project referentie)
   - GetLate.dev integratie: gebruikt default project van client

Voordelen van Deze Aanpak

✅ **Geen Breaking Changes** - alle bestaande API routes blijven werken
✅ **UI Simplificatie** - client ziet geen projecten, alleen hun bedrijf
✅ **Backend Flexibiliteit** - project layer blijft voor toekomstige uitbreidingen
✅ **Snelle Implementatie** - vooral frontend wijzigingen
✅ **Backward Compatible** - bestaande clients met meerdere projecten blijven werken

Nadelen/Risico’s

⚠️ **Multi-Project Legacy** - bestaande clients met meerdere projecten
- Oplossing: toon waarschuwing in admin als client >1 project heeft
- Voor Writgo nieuwe klanten: altijd 1 project

⚠️ **Project Selector in UI** - moet overal verwijderd worden
- Vervangen door automatische default project lookup

Volgende Stappen (Herzien)

1. **PR Mergen** - huidige bug fixes branch
2. **Auto-Project Implementeren** - wijzig client creation API
3. **Admin UI Update** - verwijder Projecten navigatie, update Klanten pagina
4. **Client Dashboard Update** - verwijder project selectors, toon bedrijfsinfo
5. **API Middleware** - helper functie: getClientDefaultProject()
6. **Testing** - volledige flow met nieuwe klant
7. **Nieuwe PR** - met volledige documentatie