Database Migration Instructions - UPDATED

⚠️ BELANGRIJKE UPDATE (12 December 2024)

De oude migraties hadden een fundamenteel probleem: ze verwezen naar de `BlogPost` tabel die alleen in `schema.sql` bestaat, niet in de migraties zelf. Dit veroorzaakte de error: **"relation BlogPost does not exist"**.

OPLOSSING: Gebruik nu het nieuwe **COMPLETE_MIGRATION_PACKAGE.sql** bestand dat ALLES bevat, inclusief de BlogPost tabel.

Nieuwe Migratie Workflow (AANBEVOLEN)

Optie 1: Verse Database (Nog geen migraties uitgevoerd)

1. **Ga naar Supabase SQL Editor**
   - Navigeer naar je project op `supabase.com`
   - Klik op "SQL Editor" in de linker sidebar

2. **Voer het complete migratie pakket uit**
   - Open `/supabase/migrations/COMPLETE_MIGRATION_PACKAGE.sql`
   - Kopieer de VOLLEDIGE inhoud
   - Plak in Supabase SQL Editor
   - Klik "Run" (rechtsonder)

3. **Verificatie**
   - Je zou moeten zien: "✅ Migration completed successfully!"
   - Ga naar stap 4 voor uitgebreide verificatie

4. **Uitgebreide verificatie**
   - Open `/supabase/migrations/VERIFY_TABLES.sql`
   - Kopieer en run in Supabase SQL Editor
   - Controleer alle checkmarks (✅)

Verwachte output:
- ✅ 6 tabellen aangemaakt (BlogPost, ContentPlan, ContentPlanItem, TopicalAuthorityMap, TopicalMapArticle, BatchJob)
- ✅ Alle foreign keys correct
- ✅ Alle indexes aanwezig
- ✅ RLS policies actief
- ✅ Triggers werkend

Optie 2: Database Cleanup + Complete Migratie

Als je eerder de oude migraties probeerde en errors kreeg:

1. **Cleanup oude tabellen**
   ```sql
   – Run dit EERST in Supabase SQL Editor
   DROP TABLE IF EXISTS “BatchJob” CASCADE;
   DROP TABLE IF EXISTS “TopicalMapArticle” CASCADE;
   DROP TABLE IF EXISTS “TopicalAuthorityMap” CASCADE;
   DROP TABLE IF EXISTS “ContentPlanItem” CASCADE;
   DROP TABLE IF EXISTS “ContentPlan” CASCADE;

– Verificatie (should return 0 rows)
SELECT table_name
FROM information_schema.tables
WHERE table_name IN (
‘ContentPlan’, ‘ContentPlanItem’,
‘TopicalAuthorityMap’, ‘TopicalMapArticle’, ‘BatchJob’
);
   ```

1. **Voer complete migratie uit**
   - Volg Optie 1, stappen 2-4

Waarom Deze Oplossing Werkt

Het Probleem

De oude migraties (`20251212_content_plans_tables_FIXED.sql` en `20251212_topical_authority_map_tables.sql`) hadden foreign keys naar de `BlogPost` tabel:

```
CONSTRAINT "ContentPlanItem_blogPostId_fkey" FOREIGN KEY ("blogPostId")
  REFERENCES "BlogPost"("id") ON DELETE SET NULL
```

Maar de `BlogPost` tabel werd **NERGENS** aangemaakt in de migraties. Hij bestaat alleen in het `schema.sql` bestand dat de meeste gebruikers NIET uitvoeren.

De Oplossing

Het nieuwe `COMPLETE_MIGRATION_PACKAGE.sql` :

1. ✅ **Maakt EERST de BlogPost tabel aan** (als deze nog niet bestaat)
2. ✅ **Dan pas de Content Plan tabellen** (met foreign keys naar BlogPost)
3. ✅ **Dan de Topical Authority Map tabellen** (ook met foreign keys naar BlogPost)
4. ✅ **Gebruikt IF NOT EXISTS** checks overal (kan veilig meerdere keren worden uitgevoerd)
5. ✅ **Bevat ALLE indexes, triggers, en RLS policies**

Verificatie Queries (Snelle Checks)

Check 1: Alle Tabellen Bestaan

```sql
SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'public'
  AND table_name IN (
    'BlogPost',
    'ContentPlan',
    'ContentPlanItem',
    'TopicalAuthorityMap',
    'TopicalMapArticle',
    'BatchJob'
  )
ORDER BY table_name;
```

Verwacht: 6 rijen

Check 2: Foreign Keys Zijn Correct

```sql
SELECT
  tc.table_name,
  kcu.column_name,
  ccu.table_name AS foreign_table_name
FROM information_schema.table_constraints AS tc
JOIN information_schema.key_column_usage AS kcu
  ON tc.constraint_name = kcu.constraint_name
JOIN information_schema.constraint_column_usage AS ccu
  ON ccu.constraint_name = tc.constraint_name
WHERE tc.constraint_type = 'FOREIGN KEY'
  AND tc.table_name IN (
    'ContentPlanItem',
    'TopicalMapArticle',
    'BatchJob'
  )
ORDER BY tc.table_name;
```

Verwacht: 8 foreign keys (waaronder 2x naar BlogPost)

Check 3: Test Insert in BlogPost

```
-- Test of BlogPost tabel werkt
INSERT INTO "BlogPost" (
  title, slug, excerpt, content
) VALUES (
  'Test Post',
  'test-post-' || gen_random_uuid()::text,
  'Test excerpt',
  'Test content'
) RETURNING id, title;

-- Clean up
DELETE FROM "BlogPost" WHERE title = 'Test Post';
```

Verwacht: INSERT succesvol, geen errors

Troubleshooting

❌ Error: “relation ‘Client’ does not exist”

Oorzaak: Je database heeft nog geen basis tabellen
Oplossing: Run EERST het basis schema:

```
-- In Supabase SQL Editor, run dit VOOR de migratie:
-- Kopieer en run /supabase/schema.sql
```

❌ Error: “duplicate key value violates unique constraint”

Oorzaak: Tabellen bestaan al uit eerdere poging
Oplossing: Gebruik Optie 2 (cleanup + retry)

❌ Error: “permission denied for table”

Oorzaak: RLS policies blokkeren toegang
Oplossing: Run als database eigenaar (service_role) in Supabase

❌ Error: “syntax error near ‘USING’“

Oorzaak: Oude PostgreSQL versie
Oplossing: Supabase gebruikt altijd PostgreSQL 14+, dit zou niet moeten gebeuren

---

Volledige Test Procedure

Run deze stappen om 100% zeker te zijn:

```sql
-- 1. Check tables
SELECT count(*) as table_count
FROM information_schema.tables
WHERE table_name IN (
  'BlogPost', 'ContentPlan', 'ContentPlanItem',
  'TopicalAuthorityMap', 'TopicalMapArticle', 'BatchJob'
);
-- Expected: 6

-- 2. Test BlogPost insert
INSERT INTO "BlogPost" (title, slug, excerpt, content)
VALUES ('Test', 'test-' || gen_random_uuid()::text, 'Test', 'Test')
RETURNING id;

-- 3. Test ContentPlan insert (replace CLIENT_ID)
INSERT INTO "ContentPlan" (
  "clientId", name, niche, "targetAudience", "totalPosts", period
)
SELECT id, 'Test Plan', 'AI', 'Developers', 5, '1 week'
FROM "Client" LIMIT 1
RETURNING id;

-- 4. Test foreign key (replace PLAN_ID and BLOGPOST_ID)
INSERT INTO "ContentPlanItem" (
  "planId", "blogPostId", title, description
) VALUES (
  'PLAN_ID',
  'BLOGPOST_ID',
  'Test Item',
  'Test Description'
);

-- 5. Clean up
DELETE FROM "ContentPlanItem" WHERE title = 'Test Item';
DELETE FROM "ContentPlan" WHERE name = 'Test Plan';
DELETE FROM "BlogPost" WHERE title = 'Test';

SELECT '✅ All tests passed!' as result;
```

Git Commit

Na succesvolle migratie:

```
cd /home/ubuntu/writgoai_app
git add .
git commit -m "fix: Remove BlogPost dependency from migrations

- Created COMPLETE_MIGRATION_PACKAGE.sql with BlogPost table included
- Fixed 'relation BlogPost does not exist' error
- Added comprehensive VERIFY_TABLES.sql script
- Updated migration instructions with new workflow
- All foreign keys now work correctly"

git push origin main
```

Bestanden in Deze Update

- ✅ **NEW:** `supabase/migrations/COMPLETE_MIGRATION_PACKAGE.sql` - Alles-in-één migratie
- ✅ **NEW:** `supabase/migrations/VERIFY_TABLES.sql` - Uitgebreide verificatie
- ✅ **UPDATED:** `DATABASE_MIGRATION_INSTRUCTIONS.md` - Deze instructies
- ⚠️ **DEPRECATED:** `20251212_content_plans_tables_FIXED.sql` - Gebruik niet meer (blijft voor reference)
- ⚠️ **DEPRECATED:** `20251212_topical_authority_map_tables.sql` - Gebruik niet meer (blijft voor reference)

Volgende Stappen

1. ✅ Run COMPLETE_MIGRATION_PACKAGE.sql in Supabase
2. ✅ Run VERIFY_TABLES.sql voor validatie
3. ✅ Test content plan API endpoints
4. ✅ Commit naar Git
5. ✅ Test in productie

⚠️ Foreign Key Issues?

Als je na de migratie foreign key problemen tegenkomt (zoals “only 6 foreign keys instead of 8” of “Key (planId)=(PLAN_ID) is not present”), gebruik dan de **Foreign Key Fix Guide**:

📖 **Zie:** FOREIGN_KEY_FIX_GUIDE.md (./FOREIGN_KEY_FIX_GUIDE.md)

Quick Fix: Run `/supabase/migrations/COMPLETE_FIX_PACKAGE.sql` in Supabase SQL Editor.

Dit script:
- ✅ Cleanup orphaned data
- ✅ Fix invalid references
- ✅ Voegt ontbrekende foreign keys toe (ContentPlanItem.blogPostId en TopicalMapArticle.blogPostId)
- ✅ Verifieert dat alle 8 foreign keys aanwezig zijn

Status: Ready for production 🚀
Last Updated: 12 December 2024
Migration Version: 2.0 (Complete Package)