

# KRITISCH: Supabase Schema Wiederherstellen





---

## PROBLEM:



---

Wir sind im Code zu einem früheren Zustand zurückgegangen (vor Clerk v3 Migration), aber die Supabase Datenbank könnte die `users` und `orgs` Tabellen bereits **gelöscht** haben!

### Der alte Code ERWARTET diese Tabellen:

-  `orgs` - Organisationen
-  `users` - User-Daten (mit `supabase_user_id` = Clerk User ID)
-  `audit_logs` - Audit-Protokolle
-  `system_events` - System-Events

### Wenn die Tabellen fehlen:

-  ERROR: relation "`users`" does not exist
-  App crasht beim Login
-  Dashboard lädt nicht

---

## LÖSUNG: Schema überprüfen und wiederherstellen





---

### Schritt 1: Supabase Datenbank öffnen

1. Gehen Sie zu [Supabase Dashboard](https://supabase.com/dashboard) (<https://supabase.com/dashboard>)
2. Wählen Sie Ihr Projekt
3. Klicken Sie auf **"Table Editor"** (im linken Menü)

### Schritt 2: Existierende Tabellen prüfen

#### Überprüfen Sie, welche Tabellen existieren:

-  `orgs` ← Sollte existieren
-  `users` ← Sollte existieren
-  `audit_logs` ← Sollte existieren
-  `system_events` ← Sollte existieren

#### Wenn `users` oder `orgs` FEHLEN:

→ Gehen Sie zu **Schritt 3**

#### Wenn ALLE Tabellen existieren:

→  **PERFEKT! Nichts zu tun!** Sie können deployen.

---

## Schritt 3: Schema wiederherstellen (falls Tabellen fehlen)

### Option A: SQL Editor verwenden (EMPFOHLEN)

#### 1. Öffnen Sie den SQL Editor:

- Im Supabase Dashboard: Klicken Sie auf **"SQL Editor"** (im linken Menü)
- Oder gehen Sie direkt: [https://supabase.com/dashboard/project/YOUR\\_PROJECT\\_ID/sql](https://supabase.com/dashboard/project/YOUR_PROJECT_ID/sql)

#### 2. Kopieren Sie das komplette Schema:

```
-- JNX-OS v1 Database Schema
-- PostgreSQL / Supabase

-- Organizations table
CREATE TABLE IF NOT EXISTS orgs (
  org_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name TEXT NOT NULL,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Users table
CREATE TABLE IF NOT EXISTS users (
  user_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  supabase_user_id UUID UNIQUE NOT NULL,
  email TEXT NOT NULL,
  org_id UUID REFERENCES orgs(org_id),
  role TEXT NOT NULL DEFAULT 'member',
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Audit logs table
CREATE TABLE IF NOT EXISTS audit_logs (
  log_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  org_id UUID REFERENCES orgs(org_id),
  actor_user_id UUID REFERENCES users(user_id),
  action TEXT NOT NULL,
  target TEXT,
  metadata JSONB,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- System events table
CREATE TABLE IF NOT EXISTS system_events (
  event_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  event_type TEXT NOT NULL,
  severity TEXT,
  message TEXT,
  metadata JSONB,
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Create indexes for better performance
CREATE INDEX IF NOT EXISTS idx_users_supabase_user_id ON users(supabase_user_id);
CREATE INDEX IF NOT EXISTS idx_users_org_id ON users(org_id);
CREATE INDEX IF NOT EXISTS idx_audit_logs_org_id ON audit_logs(org_id);
CREATE INDEX IF NOT EXISTS idx_audit_logs_created_at ON audit_logs(created_at DESC);
CREATE INDEX IF NOT EXISTS idx_system_events_created_at ON system_events(created_at DESC);
```

#### 1. Fügen Sie das SQL in den Editor ein

2. **Klicken Sie auf “Run”** (oder drücken Sie `Cmd+Enter` / `Ctrl+Enter` )

3. **Warten Sie auf Erfolg:**

✓ Success. No rows returned

4. **Verifizieren:**

- Gehen Sie zurück zu **“Table Editor”**
- Überprüfen Sie, ob alle Tabellen existieren:

- ✓ orgs
- ✓ users
- ✓ audit\_logs
- ✓ system\_events

## Option B: Schema-Datei hochladen

1. **Finden Sie die Datei:**

```
bash
nextjs_space/lib/db/schema.sql
```

2. **Öffnen Sie Supabase SQL Editor**

3. **Kopieren Sie den Inhalt** der Datei

4. **Führen Sie das SQL aus** (wie in Option A)

## Schritt 4: Clerk Webhook konfigurieren (WICHTIG!)

**WARUM?** Der alte Code nutzt Clerk Webhooks, um User-Daten in die `users` Tabelle zu syncen!

**Webhook-URL:**

```
https://www.jnslabs.ai/api/webhooks/clerk
```

### Konfiguration in Clerk:

1. Gehen Sie zu [Clerk Dashboard](https://dashboard.clerk.com) (<https://dashboard.clerk.com>)
2. Wählen Sie Ihre Application
3. Navigieren Sie zu **“Webhooks”** (im linken Menü)
4. Klicken Sie auf **“Add Endpoint”**
5. **Endpoint URL:** `https://www.jnslabs.ai/api/webhooks/clerk`
6. **Events auswählen:**
  - ✓ user.created
  - ✓ user.updated
  - ✓ user.deleted
  - ✓ organization.created
  - ✓ organization.updated
  - ✓ organization.deleted
7. Klicken Sie auf **“Create”**
8. **Kopieren Sie den Signing Secret** (z.B. `whsec_...` )

## Signing Secret in Vercel setzen:

1. Gehen Sie zu [Vercel Dashboard](https://vercel.com/dashboard) (<https://vercel.com/dashboard>)
  2. Wählen Sie Ihr Projekt
  3. **Settings → Environment Variables**
  4. Fügen Sie hinzu:
 

Key: `CLERK_WEBHOOK_SECRET`  
 Value: `whsec_...` (Ihr Signing Secret)  
 Environment: `Production`
  5. Klicken Sie auf **“Save”**
- 

## Schritt 5: Testen Sie die Webhook-Integration

### Lokal testen (Optional):

1. **Installieren Sie Clerk CLI:**

```
bash
npm install -g @clerk/clerk-sdk-node
```
2. **Starten Sie Webhook-Forwarding:**

```
bash
clerk listen --forward http://localhost:3000/api/webhooks/clerk
```
3. **Erstellen Sie einen Test-User** im Clerk Dashboard
4. **Überprüfen Sie die Logs:**
  - Clerk CLI sollte Webhook-Events anzeigen
  - Ihr Dev Server sollte die Events empfangen

### Production testen:

1. **Deployen Sie die App** (Deploy-Button)
  2. **Loggen Sie sich ein** auf [www.jnxxlabs.ai](http://www.jnxxlabs.ai)
  3. **Überprüfen Sie Supabase:**
    - Gehen Sie zu **“Table Editor”** → **“users”**
    - Sollte einen neuen Eintrag mit Ihrer Email zeigen
    - `supabase_user_id` sollte Ihre Clerk User ID sein
- 

## Verification Checklist

---

### Vor dem Deployment:

- ☐ Supabase Tabellen existieren:
- ☐ `orgs`
- ☐ `users`
- ☐ `audit_logs`
- ☐ `system_events`
- ☐ Clerk Webhook konfiguriert:

- [ ] Endpoint URL: `https://www.jnxxlabs.ai/api/webhooks/clerk`
- [ ] Events ausgewählt: `user.*`, `organization.*`
- [ ] Signing Secret kopiert
- [ ] Vercel Environment Variables:
- [ ] `CLERK_WEBHOOK_SECRET` gesetzt (Production)
- [ ] Alle anderen Clerk Keys vorhanden

## Nach dem Deployment:

- [ ] Login funktioniert auf `www.jnxxlabs.ai`
- [ ] "Setting up..." Seite erscheint (mit Auto-Refresh)
- [ ] Nach 5-10 Sekunden: Dashboard lädt
- [ ] User-Daten sind in Supabase `users` Tabelle sichtbar
- [ ] Keine Fehler in Vercel Function Logs

## Troubleshooting

### Problem: "relation 'users' does not exist"

**Ursache:** Die `users` Tabelle fehlt in Supabase

#### Lösung:

1. Führen Sie `schema.sql` in Supabase SQL Editor aus (Schritt 3)
2. Überprüfen Sie Table Editor
3. Redeploy auf Vercel



### Problem: Dashboard zeigt endlos "Setting up..."

#### Mögliche Ursachen:



##### A) Webhook nicht konfiguriert:

-  Lösung: Konfigurieren Sie Clerk Webhook (Schritt 4)




##### B) Webhook Secret fehlt:

-  Lösung: Setzen Sie `CLERK_WEBHOOK_SECRET` in Vercel (Schritt 4)
-  Redeploy nach dem Setzen!

##### C) Webhook failed:

-  Überprüfen Sie Clerk Dashboard → Webhooks → Logs
-  Überprüfen Sie Vercel Function Logs
- Häufiger Fehler: `401 Unauthorized` → Secret ist falsch

##### D) `users` Tabelle nicht beschreibbar:

-  Überprüfen Sie Supabase RLS (Row Level Security)
-  **WICHTIG:** Service Role Key sollte RLS umgehen!
-  Vercel `SUPABASE_SERVICE_ROLE_KEY` sollte korrekt sein

## Problem: “Invalid signature” in Webhook

**Ursache:** `CLERK_WEBHOOK_SECRET` ist falsch oder fehlt

### Lösung:

1. Gehen Sie zu Clerk Dashboard → Webhooks
  2. Finden Sie Ihren Endpoint
  3. Kopieren Sie den **Signing Secret** erneut
  4. Aktualisieren Sie `CLERK_WEBHOOK_SECRET` in Vercel
  5. Redeploy
- 

## Problem: User wird nicht in `users` Tabelle erstellt

### Debug-Schritte:

#### 1. Überprüfen Sie Webhook-Status:

- Clerk Dashboard → Webhooks → Ihr Endpoint
- Schauen Sie auf “Recent attempts”
- Status sollte `200 OK` sein

#### 2. Überprüfen Sie Function Logs:

- Vercel Dashboard → Ihr Projekt → Functions
- Suchen Sie nach `/api/webhooks/clerk`
- Schauen Sie auf Fehler

#### 3. Manuell testen:

```
bash
```

```
# In SQL Editor:
```

```
SELECT * FROM users;
```

- Sollte User mit Ihrer Email zeigen

#### 4. Fallback: Manuell erstellen:

```
sql
```

```
-- In Supabase SQL Editor:
```

```
INSERT INTO users (supabase_user_id, email, role)
```

```
VALUES (
```

```
  'user_YOUR_CLERK_USER_ID', -- Ihre Clerk User ID
```

```
  'jonathanjung@live.de',
```

```
  'admin'
```

```
);
```

---



## WICHTIGE HINWEISE

---

### Über die Architektur:

#### Dieser Code-Stand verwendet:

- ☒ **Clerk für Authentication** (Login/Signup)
- ☒ **Supabase für Datenspeicherung** (users, orgs, audit\_logs)
- ☒ **Clerk Webhooks für Sync** (Clerk → Supabase)

**Das bedeutet:**

- User wird in Clerk erstellt (bei Signup/Login)
- Webhook sendet Event an `/api/webhooks/clerk`
- App erstellt User in Supabase `users` Tabelle
- Dashboard lädt, wenn `users` Eintrag existiert

**KRITISCH:**

- Ohne Webhook → Kein User in Supabase → Dashboard lädt nie
- Ohne Schema → Webhook crashed → Kein User in Supabase

## ZUSAMMENFASSUNG

Was	Status	Aktion
Supabase Schema	⚠️ Überprüfen	Führen Sie <code>schema.sql</code> aus falls Tabellen fehlen
Clerk Webhook	⚠️ Konfigurieren	Endpoint + Events + Secret
Vercel Env Vars	⚠️ Setzen	<code>CLERK_WEBHOOK_SECRET</code>
Deployment	⏸️ Warten	Erst nach obigen Schritten!

## ✅ Nach Befolgung dieser Anleitung:

1. ✅ Alle Tabellen existieren in Supabase
2. ✅ Clerk Webhook ist konfiguriert
3. ✅ Environment Variables sind gesetzt
4. ✅ Sie können deployen
5. ✅ Login funktioniert
6. ✅ Dashboard lädt (nach Webhook-Sync)
7. ✅ Alles läuft! 🎉

**Status:** ⚠️ **ACTION REQUIRED BEFORE DEPLOY**

**Nächster Schritt:** Überprüfen Sie Supabase Tabellen und konfigurieren Sie den Webhook!