Deep Reflections Database Migration Guide

🎯 Overview

This guide provides step-by-step instructions for fixing the Deep Reflections database migration that was causing the constraint error. The corrected migration ensures proper SQL ordering and handles existing schema gracefully.

🐛 Issue Identified

Error: `ERROR: 42703: column "reflection_type" does not exist`

Root Cause: The original migration tried to add a constraint on the `reflection_type` column before ensuring the column existed, causing a dependency error.

✅ Solution

The corrected migration (`20250107_enhance_reflections_for_audio_CORRECTED.sql`) fixes this by:

1. **Adding all required columns first** (including `reflection_type` if missing)
2. **Then adding constraints** after columns exist
3. **Safe handling** of existing data with proper IF NOT EXISTS checks
4. **Transaction safety** with BEGIN/COMMIT for rollback capability

🚀 Migration Steps

Step 1: Backup Your Data (RECOMMENDED)

```
-- Run this in Supabase SQL Editor to backup existing reflections
CREATE TABLE reflections_backup_20250107 AS
SELECT * FROM public.reflections;
```

Step 2: Apply the Enhanced Schema Migration

1. Open **Supabase Dashboard → SQL Editor**
2. Copy the entire contents of `20250107_enhance_reflections_for_audio_CORRECTED.sql`
3. Paste into SQL Editor
4. Click **Run** to execute the migration

Step 3: Apply the RLS Policies and Security Fix

1. In the same **Supabase SQL Editor**
2. Copy the entire contents of `20250107_fix_reflections_rls_policies.sql`
3. Paste into SQL Editor
4. Click **Run** to execute the security migration

Step 4: Verify Migration Success

Run these verification queries in Supabase SQL Editor:

```sql
-- Check that all columns were added
SELECT column_name, data_type, is_nullable, column_default
FROM information_schema.columns
WHERE table_name = 'reflections'
ORDER BY ordinal_position;

-- Check that constraints were created
SELECT constraint_name, constraint_type
FROM information_schema.table_constraints
WHERE table_name = 'reflections';

-- 🔐 SECURITY VERIFICATION: Check RLS policies exist
SELECT
    schemaname,
    tablename,
    policyname,
    permissive,
    roles,
    cmd,
    qual,
    with_check
FROM pg_policies
WHERE schemaname = 'public' AND tablename = 'reflections'
ORDER BY policyname;

-- 🔍 DEBUG: Test authentication context (very important!)
SELECT public.debug_user_auth_context();

-- Test the deep_reflections_view
SELECT * FROM public.deep_reflections_view LIMIT 1;

-- Test the security-hardened functions
SELECT public.user_has_reflections();
SELECT public.get_section_reflections('test-section', 5);
```

Step 4: Test Deep Reflections Feature

1. Navigate to your app's Sacred Reflections page (`/reflections`)
2. The page should now display "Loading your reflections..." instead of showing mock data
3. Try creating a new Deep Reflection through the audio player modal
4. Verify it saves to the database and appears in the Sacred Reflections page

📋 What the Migration Adds

New Columns:

- `reflection_type` - Type of reflection (includes 'deep_reflection')
- `tags` - Array of tags for categorization
- `section_id` - Reference to book section
- `audio_timestamp` - Precise timing within audio tracks
- `section_title` - Display title for sections
- `audio_title` - Title of specific audio track

New Database Objects:

- `deep_reflections_view` - Specialized view for deep reflections
- `get_section_reflections()` - Function to get reflections by section

- `user_has_reflections()` - Function to check if user has reflections
- Multiple indexes for performance optimization

🔧 Troubleshooting

If Migration Fails:

1. **Check error message** in Supabase SQL Editor
2. **Rollback if needed** (migration uses transaction safety)
3. **Restore from backup** if necessary:
   ```
   sql
   DELETE FROM public.reflections;
   INSERT INTO public.reflections SELECT * FROM reflections_backup_20250107;
   ```

🔐 If You Get “RLS Policy Violation” Errors:

Error: `new row violates row-level security policy for table 'reflections'`

Solutions:
1. **Verify RLS policies exist:**
```
sql
SELECT COUNT(*) FROM pg_policies
WHERE schemaname = 'public' AND tablename = 'reflections';
-- Should return 4 (one for each CRUD operation)
```

1. **Check authentication context:**
   ```
   sql
   SELECT public.debug_user_auth_context();
   -- Should show authenticated = true and your user_id
   ```

2. **Verify user_id is being set correctly** in your app:
   - Check that your app passes `user_id: auth.uid()` when inserting reflections
   - Ensure you’re using the authenticated Supabase client, not anon client
   - Verify auth headers are included in API requests

3. **Test RLS manually:**
   ```sql
   – This should work (returns your reflections)
   SELECT * FROM public.reflections;

– This should fail with RLS error if user_id doesn’t match
INSERT INTO public.reflections (user_id, question_text, answer_text, reflection_type)
VALUES (‘00000000-0000-0000-0000-000000000000’, ‘Test’, ‘Test’, ‘deep_reflection’);
```

If Supabase Security Advisor Still Shows Warnings:

1. **Check function security warnings are resolved:**
   - Functions should have `SET search_path = ''`
   - Views should not use `SECURITY DEFINER`
2. **Run the security verification again:**
   ```
   sql
   SELECT routine_name, security_type, security_invoker
   ```

```
  FROM information_schema.routines
  WHERE routine_schema = 'public';
```

If App Still Shows Mock Data:

1. **Clear browser cache** and refresh
2. **Check Supabase connection** in app (should show green checkmarks)
3. **Verify authentication state** in browser dev tools
4. **Check that useDeepReflection hook** is using real Supabase client

If Deep Reflections Don't Save:

1. **Check browser console** for JavaScript errors
2. **Verify authentication** (user must be logged in)
3. **Test authentication context** with debug function
4. **Check network tab** for failed API requests
5. **Verify user_id is being passed** in reflection data

🎉 Expected Results

After successful migration:
- ✅ Sacred Reflections page shows "Loading your reflections..."
- ✅ Deep Reflection modal saves to real Supabase database
- ✅ Reflections appear in Sacred Reflections page
- ✅ Delete functionality works with optimistic updates
- ✅ No more mock data fallbacks

📞 Support

If you encounter any issues:
1. Check the migration log for specific error messages
2. Verify your Supabase project has all necessary permissions
3. Ensure your database has auth.users table (required for foreign key)
4. Make sure RLS is properly configured

🔄 Rollback Plan

If you need to rollback the migration:

```
-- Remove added columns (be careful - this will lose data!)
ALTER TABLE public.reflections
DROP COLUMN IF EXISTS section_id,
DROP COLUMN IF EXISTS audio_timestamp,
DROP COLUMN IF EXISTS section_title,
DROP COLUMN IF EXISTS audio_title;

-- Remove constraint
ALTER TABLE public.reflections DROP CONSTRAINT IF EXISTS reflec-
tions_reflection_type_check;

-- Remove functions and views
DROP VIEW IF EXISTS public.deep_reflections_view;
DROP FUNCTION IF EXISTS public.get_section_reflections;
DROP FUNCTION IF EXISTS public.user_has_reflections;

-- Restore from backup if needed
DELETE FROM public.reflections;
INSERT INTO public.reflections SELECT * FROM reflections_backup_20250107;
```

🎯 Final Notes

- This migration is **safe for production** use
- **Existing data** will be preserved
- **New features** will be immediately available
- The app will **automatically connect** to live Supabase data after migration