Database Schema Mismatch - Diagnosis and Fix

Date: October 13, 2025
Status: Critical Issue - Site Down
Root Cause: Database migrations not applied to Railway PostgreSQL database

---

Problem Summary

After pushing authentication fixes to GitHub (commit `c15a460`), Railway auto-deployed the updated code. However, the site is now down with repeated database query failures.

Error Details

Error Message:

```
Failed query: select "id", "username", "password", "email", "first_name",
"last_name",
"profile_image_url", "subscription_status", "subscription_tier", "subscription_expires_at",
"downloads_used", "downloads_reset_at", "is_admin", "created_at", "updated_at"
from "users" where "users"."id" = $1 limit $2
```

Symptoms:
- ✅ Database connection successful
- ❌ All queries to users table failing
- ❌ Login shows “401: Unauthorized” error
- ❌ Application completely non-functional

Root Cause Analysis

1. Migration Files Added But Not Applied

In commit `c15a460`, the following migration files were created:
- `migrations/0000_luxuriant_black_panther.sql`
- `migrations/meta/0000_snapshot.json`
- `migrations/meta/_journal.json`

However, **these migrations were never executed on the Railway database**.

2. Schema Definition vs. Database State

Expected Schema (from code):

```
CREATE TABLE "users" (
  "id" varchar PRIMARY KEY DEFAULT gen_random_uuid() NOT NULL,
  "username" varchar NOT NULL,
  "password" varchar NOT NULL,
  "email" varchar,
  "first_name" varchar,
  "last_name" varchar,
  "profile_image_url" varchar,
  "subscription_status" varchar DEFAULT 'free' NOT NULL,
  "subscription_tier" varchar,
  "subscription_expires_at" timestamp,
  "downloads_used" integer DEFAULT 0 NOT NULL,
  "downloads_reset_at" timestamp,
  "is_admin" boolean DEFAULT false NOT NULL,
  "created_at" timestamp DEFAULT now(),
  "updated_at" timestamp DEFAULT now(),
  CONSTRAINT "users_username_unique" UNIQUE("username"),
  CONSTRAINT "users_email_unique" UNIQUE("email")
);
```

Current Database State:
The Railway PostgreSQL database likely has:
- An old/incomplete users table schema, OR
- No users table at all

This causes all queries to fail because the columns don't exist or the table structure doesn't match.

3. No Automatic Migration on Deploy

The application doesn't have automatic migration execution configured:
- ❌ No migration script in `package.json`
- ❌ No migration logic in `server/index.ts`
- ❌ No Railway build hook to run migrations

Current build process:

```
"scripts": {
  "prebuild": "node scripts/prebuild.js",  // Only installs dependencies
  "build": "vite build && esbuild ...",    // Only builds code
  "start": "cross-env NODE_ENV=production node dist/index.js"
}
```

Solution Options

Option 1: Run Migrations Manually (Fastest - Recommended)

Step 1: Connect to Railway PostgreSQL Database

1. Go to Railway dashboard → PostgreSQL service
2. Click on "Connect" tab
3. Copy the connection string (format: `postgresql://user:pass@host:port/dbname`)

Step 2: Run Migration SQL Directly

Using a database client (like Beekeeper Studio shown in screenshots) or psql:

```
-- Run the entire migration file content:

CREATE TABLE IF NOT EXISTS "field_mappings" (
    "id" varchar PRIMARY KEY DEFAULT gen_random_uuid() NOT NULL,
    "project_id" varchar NOT NULL,
    "source_field" text NOT NULL,
    "target_field" text,
    "mapping_type" text NOT NULL,
    "confidence" integer,
    "transformation" jsonb,
    "is_validated" boolean DEFAULT false,
    "created_at" timestamp DEFAULT now() NOT NULL
);

CREATE TABLE IF NOT EXISTS "integration_projects" (
    "id" varchar PRIMARY KEY DEFAULT gen_random_uuid() NOT NULL,
    "user_id" varchar,
    "name" text NOT NULL,
    "description" text,
    "status" text DEFAULT 'draft' NOT NULL,
    "source_schema" jsonb,
    "target_schema" jsonb,
    "field_mappings" jsonb,
    "transformation_logic" jsonb,
    "integration_code" jsonb,
    "xslt_validation" jsonb,
    "created_at" timestamp DEFAULT now() NOT NULL,
    "updated_at" timestamp DEFAULT now() NOT NULL
);

CREATE TABLE IF NOT EXISTS "sessions" (
    "sid" varchar PRIMARY KEY NOT NULL,
    "sess" jsonb NOT NULL,
    "expire" timestamp NOT NULL
);

CREATE TABLE IF NOT EXISTS "uploaded_files" (
    "id" varchar PRIMARY KEY DEFAULT gen_random_uuid() NOT NULL,
    "project_id" varchar NOT NULL,
    "file_name" text NOT NULL,
    "file_type" text NOT NULL,
    "file_size" integer NOT NULL,
    "system_type" text NOT NULL,
    "detected_schema" jsonb,
    "schema_confidence" integer,
    "uploaded_at" timestamp DEFAULT now() NOT NULL
);

-- Drop and recreate users table to match schema
DROP TABLE IF EXISTS "users" CASCADE;

CREATE TABLE "users" (
    "id" varchar PRIMARY KEY DEFAULT gen_random_uuid() NOT NULL,
    "username" varchar NOT NULL,
    "password" varchar NOT NULL,
    "email" varchar,
    "first_name" varchar,
    "last_name" varchar,
    "profile_image_url" varchar,
    "subscription_status" varchar DEFAULT 'free' NOT NULL,
    "subscription_tier" varchar,
    "subscription_expires_at" timestamp,
    "downloads_used" integer DEFAULT 0 NOT NULL,
    "downloads_reset_at" timestamp,
    "is_admin" boolean DEFAULT false NOT NULL,
    "created_at" timestamp DEFAULT now(),
    "updated_at" timestamp DEFAULT now(),
    CONSTRAINT "users_username_unique" UNIQUE("username"),
    CONSTRAINT "users_email_unique" UNIQUE("email")
);

-- Add foreign key constraints
ALTER TABLE "field_mappings" ADD CONSTRAINT "field_mappings_project_id_integration_projects_id_fk"
  FOREIGN KEY ("project_id") REFERENCES "public"."integration_projects"("id")
  ON DELETE no action ON UPDATE no action;

ALTER TABLE "integration_projects" ADD CONSTRAINT "integration_projects_user_id_users_id_fk"
  FOREIGN KEY ("user_id") REFERENCES "public"."users"("id")
  ON DELETE no action ON UPDATE no action;

ALTER TABLE "uploaded_files" ADD CONSTRAINT "uploaded_files_project_id_integration_projects_id_fk"
  FOREIGN KEY ("project_id") REFERENCES "public"."integration_projects"("id")
  ON DELETE no action ON UPDATE no action;

-- Create index
CREATE INDEX IF NOT EXISTS "IDX_session_expire" ON "sessions" USING btree ("expire");
```

Step 3: Verify Tables Created

```
-- Check if users table exists with correct schema
SELECT column_name, data_type, is_nullable, column_default
FROM information_schema.columns
WHERE table_name = 'users'
ORDER BY ordinal_position;

-- Check all tables
SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'public';
```

Step 4: Restart Railway Service

After running the migrations, restart the `integration-hub` service in Railway dashboard.

Option 2: Use Drizzle Kit to Push Schema (Alternative)

Step 1: Set DATABASE_URL Locally

```
export DATABASE_URL="postgresql://user:pass@host:port/dbname"
```

Step 2: Push Schema to Database

```
cd /home/ubuntu/integration-hub
npx drizzle-kit push
```

This will:
1. Connect to the Railway database
2. Compare schema.ts with actual database
3. Generate and apply necessary changes
4. Create all missing tables and columns

Step 3: Restart Railway Service

Option 3: Add Automatic Migration to Railway Build (Long-term Fix)

Step 1: Add Migration Scripts to package.json

```
"scripts": {
  "prebuild": "node scripts/prebuild.js",
  "db:migrate": "drizzle-kit push",
  "build": "npm run db:migrate && vite build && esbuild server/index.ts --platform=node --bundle --format=esm --outdir=dist --packages=external --external:vite --external:vite.config.ts",
  "start": "cross-env NODE_ENV=production node dist/index.js",
  "dev": "vite"
}
```

Step 2: Update Railway Build Configuration

In Railway project settings, set:
- **Build Command:** `npm run build`
- **Start Command:** `npm start`

This ensures migrations run automatically on every deploy.

Recommended Action Plan

Immediate Fix (Now):

1. ✅ Use **Option 1** (Manual SQL execution) - fastest way to restore service
2. ✅ Verify using a database client (Beekeeper Studio as shown in your screenshots)
3. ✅ Restart Railway service
4. ✅ Test login functionality

Long-term Fix (Next):

1. ✅ Implement **Option 3** (automatic migrations)
2. ✅ Add migration script to package.json
3. ✅ Push changes to GitHub
4. ✅ Verify auto-deploy runs migrations
5. ✅ Add migration status logging to startup

Verification Steps

After applying the fix:

1. **Check Database Connection:**
```
sql
   SELECT NOW();
```

2. **Verify Users Table Schema:**
```
sql
   \d users
```

3. **Test Creating a User:**
```
sql
   INSERT INTO users (username, password, email)
   VALUES ('testuser', 'hashedpass', 'test@example.com')
   RETURNING *;
```

4. **Test Query from Application:**
```
sql
   SELECT id, username, password, email, first_name, last_name,
          profile_image_url, subscription_status, subscription_tier,
          subscription_expires_at, downloads_used, downloads_reset_at,
          is_admin, created_at, updated_at
   FROM users
   WHERE username = 'testuser'
   LIMIT 1;
```

5. **Check Application Logs:**
   - Look for "✅ Database connection verified successfully"
   - Should NOT see query failure errors
   - Login should work without 401 errors

Files Involved

- **Schema Definition:** `/shared/schema.ts`
- **Migration File:** `/migrations/0000_luxuriant_black_panther.sql`
- **Database Config:** `/drizzle.config.ts`
- **Database Connection:** `/server/db.ts`
- **Build Script:** `/package.json`

Additional Notes

Why This Happened

1. Migrations were generated locally with `drizzle-kit generate`
2. Migration files were committed to git
3. Code was pushed to GitHub

4. Railway auto-deployed the new code
5. **BUT** migrations were never executed on Railway database
6. Application code expects new schema, database has old/missing schema
7. All queries fail due to schema mismatch

Prevention

- Add automatic migration execution to build process
- Add database schema version tracking
- Add startup validation to check schema matches expectations
- Consider using Drizzle’s migrate() function in application startup

---

Contact Information

If you need help executing these steps or encounter any issues, the SQL script above is ready to run directly in your database client (Beekeeper Studio).

Critical: Backup your database before running DROP TABLE commands if you have existing data.