mySafePlay - Vercel Postgres Migration Guide

Overview

This guide provides step-by-step instructions to migrate the mySafePlay application from Supabase to Vercel Postgres. The migration addresses the network connectivity issues with Supabase on Vercel's production environment.

Background

Vercel's production environment blocks direct database connections on port 5432, preventing the mySafePlay application from connecting to the Supabase database. Vercel Postgres (powered by Neon) provides native integration without these network restrictions.

Prerequisites

- Access to Vercel dashboard for safeplay-staging project
- Access to current Supabase project
- Local development environment with Node.js and npm
- PostgreSQL client tools (`pg_dump` , `pg_restore`) installed

Migration Steps

1. Create Vercel Postgres Database

1. **Navigate to Vercel Dashboard**
   - Go to https://vercel.com/dashboard
   - Select your `safeplay-staging` project

2. **Create Database**
   - Click on the "Storage" tab
   - Click "Create Database" or "Connect Database"
   - Select "Postgres" (Neon-powered)
   - Choose database name: `safeplay-production`
   - Select region closest to your users
   - Click "Create"

3. **Note Environment Variables**
   Vercel automatically creates these environment variables:
   - `POSTGRES_URL` - Pooled connection for runtime
   - `POSTGRES_PRISMA_URL` - Optimized for Prisma
   - `POSTGRES_URL_NON_POOLING` - Direct connection for migrations

2. Update Environment Configuration

1. **Vercel Dashboard Settings**
   - Go to Project Settings → Environment Variables

- Set `DATABASE_URL` to use `POSTGRES_PRISMA_URL`
- Apply to Production, Preview, and Development environments

2. **Local Development**
   `bash`
   `npx vercel env pull .env.local`

3. Export Data from Supabase

1. **Get Supabase Connection String**
   - Supabase Dashboard → Project Settings → Database
   - Copy the **non-pooled** connection string (port 5432)
   - Replace `[YOUR-PASSWORD]` with actual password

2. **Create Database Backup**
   ```bash
   # Set environment variable
   export SUPABASE_DB_URL="postgresql://postgres.xxxxxxxxxxxxxxxxxxxx:YOUR_PASSWORD@aws-0-us-west-1.pooler.supabase.com:5432/postgres"

# Create backup
pg_dump "$SUPABASE_DB_URL" \
--clean \
--if-exists \
--quote-all-identifiers \
--no-owner \
--no-privileges \
-Fc \
-v \
-f safeplay_backup.dump
```

4. Deploy Schema to Vercel Postgres

```
# Generate Prisma client
npx prisma generate

# Deploy migrations
npx prisma migrate deploy
```

5. Import Data to Vercel Postgres

```
# Get Vercel Postgres connection string from .env.local
export VERCEL_POSTGRES_URL="your_postgres_url_from_env_file"

# Import data (critical: use -O flag for Vercel Postgres)
pg_restore -v \
  -O \
  --no-owner \
  -d "$VERCEL_POSTGRES_URL" \
  safeplay_backup.dump
```

6. Verify Migration

```
# Run verification script
node scripts/verify-migration.js

# Test API endpoint
curl https://safeplay-staging.vercel.app/api/test-db
```

7. Deploy Application

```
# Make deployment script executable
chmod +x scripts/deploy-with-new-db.sh

# Run deployment
./scripts/deploy-with-new-db.sh
```

Important Notes

Vercel Postgres Specifics

- **Powered by Neon**: Vercel Postgres uses Neon.tech as the underlying provider
- **Connection Pooling**: Uses PgBouncer for connection pooling
- **No Superuser**: Limited privileges compared to full PostgreSQL superuser
- **SSL Required**: All connections must use SSL

Migration Considerations

- **Use `-0` flag**: Essential for `pg_restore` due to ownership limitations
- **Non-pooled connections**: Use `POSTGRES_URL_NON_POOLING` for migrations
- **Pooled connections**: Use `POSTGRES_PRISMA_URL` for application runtime

Schema Configuration

The Prisma schema has been updated to support both pooled and direct connections:

```
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
  directUrl = env("POSTGRES_URL_NON_POOLING")
}
```

Troubleshooting

Common Issues

1. **Connection Errors**
   - Verify environment variables are correctly set
   - Ensure using non-pooled connection for migrations

2. **Permission Errors**
   - Always use `-O` and `--no-owner` flags with `pg_restore`
   - Vercel Postgres doesn't support `ALTER OWNER` statements

3. **Migration Failures**
   - Check Prisma schema syntax
   - Verify all required environment variables are present

Verification Steps

1. **Database Connection**: Use `/api/test-db` endpoint
2. **Data Integrity**: Run `scripts/verify-migration.js`
3. **Application Functionality**: Test core features
4. **Performance**: Monitor response times and query performance

Rollback Plan

If migration fails:

1. **Revert Environment Variables**
   - Change `DATABASE_URL` back to Supabase connection
   - Redeploy application

2. **Data Recovery**
   - Supabase data remains unchanged
   - Can re-import from backup if needed

Post-Migration Tasks

1. **Monitor Application**
   - Check logs for database-related errors
   - Monitor performance metrics

2. **Update Documentation**
   - Update deployment procedures
   - Document new database configuration

3. **Team Communication**
   - Inform team of new database setup
   - Update development environment setup

Support

- **Vercel Support**: For platform-specific issues
- **Neon Documentation**: For database-specific questions
- **Prisma Documentation**: For ORM-related issues

Files Created/Modified

- `pages/api/test-db.js` - Database connection test endpoint
- `scripts/verify-migration.js` - Migration verification script
- `scripts/deploy-with-new-db.sh` - Automated deployment script
- `prisma/schema.prisma` - Updated with directUrl configuration
- `MIGRATION_GUIDE.md` - This documentation

Migration Date: $(date)
Application: mySafePlay
Database: Vercel Postgres (Neon)
Environment: Production (safeplay-staging.vercel.app)