Drizzle ORM Migration Guide

Overview

This guide documents the migration from Prisma to Drizzle ORM in the Sports Bar TV Controller application.

Migration Status

✅ Completed

- Core Drizzle configuration (`src/db/index.ts`)
- Schema definition (`src/db/schema.ts`)
- Helper functions (`src/lib/db-helpers.ts`)
- Wolfpack API routes:
- `src/app/api/wolfpack/inputs/route.ts`
- `src/app/api/wolfpack/route-to-matrix/route.ts`
- Example routes:
- `src/app/api/schedules/route.ts`
- `src/app/api/home-teams/route.ts`

🔄 In Progress

- Audio processor routes
- Test routes
- Additional API endpoints

❌ To Be Migrated

- Remaining API routes (~100 files)
- Service files using Prisma
- Remove Prisma compatibility adapter
- Remove prisma directory

Migration Pattern

1. Import Changes

Before (Prisma):

```
import { prisma } from '@/lib/db'
```

After (Drizzle):

```
import { db, schema } from '@/db'
import { eq, and, or, desc, asc, inArray } from 'drizzle-orm'
import { logger } from '@/lib/logger'
import { findFirst, findMany, create, update, upsert } from '@/lib/db-helpers'
```

2. Query Conversions

Find First

Before (Prisma):

```
const config = await prisma.matrixConfiguration.findFirst({
  where: { isActive: true }
})
```

After (Drizzle with helpers):

```
const config = await findFirst('matrixConfigurations', {
  where: eq(schema.matrixConfigurations.isActive, true)
})
```

After (Drizzle direct):

```
const config = await db
  .select()
  .from(schema.matrixConfigurations)
  .where(eq(schema.matrixConfigurations.isActive, true))
  .limit(1)
  .get()
```

Find Many with Relations

Before (Prisma):

```
const config = await prisma.matrixConfiguration.findFirst({
  where: { isActive: true },
  include: {
    inputs: {
      where: { isActive: true },
      orderBy: { channelNumber: 'asc' }
    }
  }
})
```

After (Drizzle - Two Queries):

```
// Get config
const config = await db
  .select()
  .from(schema.matrixConfigurations)
  .where(eq(schema.matrixConfigurations.isActive, true))
  .limit(1)
  .get()

// Get related inputs
const inputs = await db
  .select()
  .from(schema.matrixInputs)
  .where(
    and(
      eq(schema.matrixInputs.configId, config.id),
      eq(schema.matrixInputs.isActive, true)
    )
  )
  .orderBy(asc(schema.matrixInputs.channelNumber))
  .all()
```

Create

Before (Prisma):

```
await prisma.testLog.create({
  data: {
    testType: 'wolfpack',
    testName: 'Connection Test',
    status: 'success'
  }
})
```

After (Drizzle with helpers):

```
await create('testLogs', {
  testType: 'wolfpack',
  testName: 'Connection Test',
  status: 'success'
})
```

Upsert

Before (Prisma):

```
await prisma.wolfpackMatrixRouting.upsert({
  where: { matrixOutputNumber },
  update: {
    wolfpackInputNumber,
    wolfpackInputLabel: input.label
  },
  create: {
    matrixOutputNumber,
    wolfpackInputNumber,
    wolfpackInputLabel: input.label
  }
})
```

After (Drizzle with helpers):

```
await upsert(
  'wolfpackMatrixRoutings',
  eq(schema.wolfpackMatrixRoutings.matrixOutputNumber, matrixOutputNumber),
  {
    // create data
    matrixOutputNumber,
    wolfpackInputNumber,
    wolfpackInputLabel: input.label
  },
  {
    // update data
    wolfpackInputNumber,
    wolfpackInputLabel: input.label
  }
)
```

3. Logging

Add comprehensive logging to all database operations:

```
logger.api.request('GET', '/api/wolfpack/inputs')
logger.api.response('GET', '/api/wolfpack/inputs', 200, { count: inputs.length })
logger.api.error('GET', '/api/wolfpack/inputs', error)
```

4. Error Handling

Replace `console.error` with `logger.error`:

```
// Before
console.error('Error fetching data:', error)

// After
logger.api.error('GET', '/api/endpoint', error)
```

Database Helper Functions

Available from `@/lib/db-helpers`:

- **findFirst** - Find single record
- **findMany** - Find multiple records
- **findUnique** - Find by unique field
- **create** - Create single record
- **createMany** - Create multiple records
- **update** - Update single record
- **updateMany** - Update multiple records
- **upsert** - Create or update
- **deleteRecord** - Delete single record
- **deleteMany** - Delete multiple records
- **count** - Count records
- **transaction** - Execute in transaction

Drizzle Operators

Available from `drizzle-orm`:

- **eq** - Equal to
- **ne** - Not equal to
- **gt** - Greater than
- **gte** - Greater than or equal
- **lt** - Less than
- **lte** - Less than or equal
- **and** - Logical AND
- **or** - Logical OR
- **inArray** - IN clause
- **like** - LIKE pattern matching
- **desc** - Descending order
- **asc** - Ascending order

Table Name Mapping

Prisma models map to Drizzle tables with different casing:

Prisma Model	Drizzle Table
`prisma.matrixConfiguration`	`schema.matrixConfigurations`
`prisma.matrixInput`	`schema.matrixInputs`
`prisma.matrixOutput`	`schema.matrixOutputs`
`prisma.wolfpackMatrixRouting`	`schema.wolfpackMatrixRoutings`
`prisma.wolfpackMatrixState`	`schema.wolfpackMatrixStates`
`prisma.testLog`	`schema.testLogs`
`prisma.audioProcessor`	`schema.audioProcessors`
`prisma.audioZone`	`schema.audioZones`

See `src/db/schema.ts` for complete table definitions.

Testing

After migration, test:

1. **API Endpoints** - All CRUD operations
2. **Relations** - Data fetching with related tables
3. **Logging** - Verify logger output
4. **Error Handling** - Test error scenarios

5. **Performance** - Check query performance

Cleanup Checklist

Once all files are migrated:

1. ✅ Remove `src/db/prisma-adapter.ts`
2. ✅ Remove `src/lib/prisma.ts`
3. ✅ Remove `prisma/` directory
4. ✅ Update all imports to use Drizzle
5. ✅ Test all features end-to-end
6. ✅ Remove Prisma from package.json (if exists)
7. ✅ Update documentation

Reference Examples

See these files for complete migration examples:
- `src/app/api/schedules/route.ts` - Full CRUD with helpers
- `src/app/api/wolfpack/inputs/route.ts` - Simple query with relations
- `src/app/api/wolfpack/route-to-matrix/route.ts` - Complex upsert operations
- `src/lib/db-helpers.ts` - All helper function implementations

Common Issues

Issue: Boolean values not working

Solution: Use integers (0/1) for boolean fields in SQLite

Issue: Date fields showing as strings

Solution: Convert dates with `new Date().toISOString()`

Issue: Relations not working

Solution: Use separate queries and manually join data

Issue: Type errors

Solution: Use `schema.tableName.fieldName` for type safety

Support

For questions or issues:
1. Check the Drizzle ORM documentation: https://orm.drizzle.team
2. Review example files in the codebase
3. Test queries in Drizzle Studio: `npm run db:studio`