# Data Architecture

## Database Schema Overview

**Philosophy:** Hybrid approach combining normalized relational tables with JSONB flexibility.

### 15 Core Tables

1. **municipalities** - Multi-city support (single city for MVP)
2. **departments** - Organizational units (Water & Field Services, Parks & Rec, IT, etc.)
3. **fiscal\_years** - Reference table for fiscal year periods
4. **users** - User profiles and roles (extends Supabase auth.users)
5. **strategic\_plans** - The 3-year strategic planning document
6. **strategic\_goals** - 3-5 major goals per plan
7. **initiatives** - Individual strategic initiatives (heart of the plan)
8. **initiative\_budgets** - Normalized budget tracking for aggregation
9. **initiative\_kpis** - Performance metrics / Key Performance Indicators
10. **quarterly\_milestones** - Implementation timeline tracking
11. **initiative\_dependencies** - Dependencies between initiatives (many-to-many)
12. **initiative\_collaborators** - Cross-departmental initiatives (many-to-many)
13. **comments** - Collaborative feedback and discussion
14. **audit\_logs** - Track all changes for accountability
15. **document\_embeddings** - Vector embeddings for RAG/AI (Phase 2)

### Entity Relationship Diagram

municipalities  
 ↓ (1:many)  
departments ──→ users (many:1)  
 ↓ (1:many)  
strategic\_plans ──→ fiscal\_years (3-year span)  
 ↓ (1:many)  
strategic\_goals  
 ↓ (1:many)  
initiatives ←──→ initiatives (dependencies, many:many)  
 ↓ ↓ (many:many via junction)  
 ↓ departments (collaborative initiatives)  
 ↓  
 ├─→ initiative\_budgets (normalized financial tracking)  
 ├─→ initiative\_kpis (performance metrics)  
 ├─→ quarterly\_milestones (implementation timeline)  
 ├─→ comments (feedback and discussion)  
 └─→ audit\_logs (change tracking)

## TypeScript Data Models

**Core Type Definitions** (types/database.ts):

// Auto-generated from Supabase CLI  
export type Database = {  
 public: {  
 Tables: {  
 strategic\_plans: {  
 Row: {  
 id: string  
 department\_id: string  
 start\_fiscal\_year\_id: string  
 end\_fiscal\_year\_id: string  
 title: string  
 status: 'draft' | 'under\_review' | 'approved' | 'active' | 'archived'  
 version: string  
 executive\_summary: string | null  
 department\_vision: string | null  
 swot\_analysis: Json | null  
 environmental\_scan: Json | null  
 benchmarking\_data: Json | null  
 total\_investment\_amount: number  
 approved\_by: string | null  
 approved\_at: string | null  
 published\_at: string | null  
 created\_by: string  
 created\_at: string  
 updated\_at: string  
 }  
 Insert: Omit<Row, 'id' | 'created\_at' | 'updated\_at'>  
 Update: Partial<Insert>  
 }  
 initiatives: {  
 Row: {  
 id: string  
 strategic\_goal\_id: string  
 lead\_department\_id: string  
 fiscal\_year\_id: string  
 initiative\_number: string  
 name: string  
 priority\_level: 'NEED' | 'WANT' | 'NICE\_TO\_HAVE'  
 rank\_within\_priority: number  
 description: string  
 rationale: string | null  
 expected\_outcomes: Json | null  
 status: 'not\_started' | 'in\_progress' | 'at\_risk' | 'completed' | 'deferred'  
 financial\_analysis: Json | null  
 roi\_analysis: Json | null  
 cost\_benefit\_analysis: Json | null  
 total\_year\_1\_cost: number  
 total\_year\_2\_cost: number  
 total\_year\_3\_cost: number  
 responsible\_party: string | null  
 created\_at: string  
 updated\_at: string  
 }  
 Insert: Omit<Row, 'id' | 'created\_at' | 'updated\_at'>  
 Update: Partial<Insert>  
 }  
 // ... other tables  
 }  
 Views: {  
 // Materialized views for dashboards  
 }  
 Functions: {  
 // RLS helper functions  
 }  
 }  
}

**Application-Level Types** (types/models.ts):

// Strategic Plan with relationships  
export interface StrategicPlanWithRelations {  
 id: string  
 department: Department  
 start\_fiscal\_year: FiscalYear  
 end\_fiscal\_year: FiscalYear  
 title: string  
 status: PlanStatus  
 goals: StrategicGoalWithInitiatives[]  
 total\_investment\_amount: number  
 created\_by: User  
 created\_at: string  
 updated\_at: string  
}  
  
// Initiative with full context  
export interface InitiativeWithContext {  
 id: string  
 strategic\_goal: StrategicGoal  
 lead\_department: Department  
 fiscal\_year: FiscalYear  
 initiative\_number: string  
 name: string  
 priority\_level: PriorityLevel  
 rank\_within\_priority: number  
 description: string  
 status: InitiativeStatus  
 financial\_analysis: FinancialAnalysis  
 roi\_analysis: ROIAnalysis  
 budgets: InitiativeBudget[]  
 kpis: InitiativeKPI[]  
 milestones: QuarterlyMilestone[]  
 collaborators: InitiativeCollaborator[]  
 comments: Comment[]  
}  
  
// JSONB Structures  
export interface FinancialAnalysis {  
 year\_1?: YearBudget  
 year\_2?: YearBudget  
 year\_3?: YearBudget  
 funding\_sources: FundingSource[]  
}  
  
export interface YearBudget {  
 personnel\_costs: number  
 equipment\_technology: number  
 professional\_services: number  
 training\_development: number  
 materials\_supplies: number  
 other\_costs: number  
 total: number  
}  
  
export interface FundingSource {  
 source: string // "General Fund", "EPA Grant", etc.  
 amount: number  
 status: 'secured' | 'requested' | 'pending' | 'projected'  
}  
  
export interface ROIAnalysis {  
 financial: {  
 projected\_annual\_savings: number  
 projected\_revenue\_generation: number  
 payback\_period\_months: number  
 three\_year\_net\_impact: number  
 }  
 non\_financial: {  
 service\_quality\_improvement?: string  
 efficiency\_gains?: string  
 risk\_reduction?: string  
 citizen\_satisfaction?: string  
 employee\_impact?: string  
 }  
}

## Data Access Patterns

**Server Component Data Fetching**:

// app/dashboard/page.tsx (Server Component)  
import { createServerSupabaseClient } from '@/lib/supabase/server'  
  
export default async function DashboardPage() {  
 const supabase = createServerSupabaseClient()  
  
 // RLS automatically filters by user's department  
 const { data: plans } = await supabase  
 .from('strategic\_plans')  
 .select(`  
 \*,  
 department:departments(\*),  
 start\_fiscal\_year:fiscal\_years!start\_fiscal\_year\_id(\*),  
 goals:strategic\_goals(  
 \*,  
 initiatives(\*)  
 )  
 `)  
 .eq('status', 'active')  
 .order('created\_at', { ascending: false })  
  
 return <DashboardView plans={plans} />  
}

**Client Component Data Fetching** (rare, prefer Server Components):

// components/InitiativeForm.tsx (Client Component)  
'use client'  
  
import { createBrowserSupabaseClient } from '@/lib/supabase/client'  
import { useState, useEffect } from 'react'  
  
export function InitiativeForm() {  
 const supabase = createBrowserSupabaseClient()  
 const [goals, setGoals] = useState([])  
  
 useEffect(() => {  
 supabase  
 .from('strategic\_goals')  
 .select('\*')  
 .then(({ data }) => setGoals(data || []))  
 }, [])  
  
 return <form>...</form>  
}

**Server Actions for Mutations**:

// app/actions/initiatives.ts  
'use server'  
  
import { createServerSupabaseClient } from '@/lib/supabase/server'  
import { revalidatePath } from 'next/cache'  
import { z } from 'zod'  
  
const initiativeSchema = z.object({  
 strategic\_goal\_id: z.string().uuid(),  
 name: z.string().min(5),  
 priority\_level: z.enum(['NEED', 'WANT', 'NICE\_TO\_HAVE']),  
 description: z.string().min(20),  
 // ... other fields  
})  
  
export async function createInitiative(formData: FormData) {  
 const supabase = createServerSupabaseClient()  
  
 // Parse and validate  
 const parsed = initiativeSchema.safeParse(Object.fromEntries(formData))  
 if (!parsed.success) {  
 return { error: parsed.error.flatten() }  
 }  
  
 // Insert (RLS enforces department access)  
 const { data, error } = await supabase  
 .from('initiatives')  
 .insert(parsed.data)  
 .select()  
 .single()  
  
 if (error) {  
 return { error: error.message }  
 }  
  
 // Revalidate cache  
 revalidatePath('/dashboard')  
 revalidatePath(`/initiatives/${data.id}`)  
  
 return { data }  
}