Contents

SECTION 38: NEURAL-FIRST ORCHESTRATION & THINK TANK WORKFLOW REGISTRY (v4.4.0)

This section transforms RADIANT from template-based to neural-first orchestration Includes: Think Tank Workflow Registry, Visual Editor, Per-User Neural Models, Real-Time Steering

---

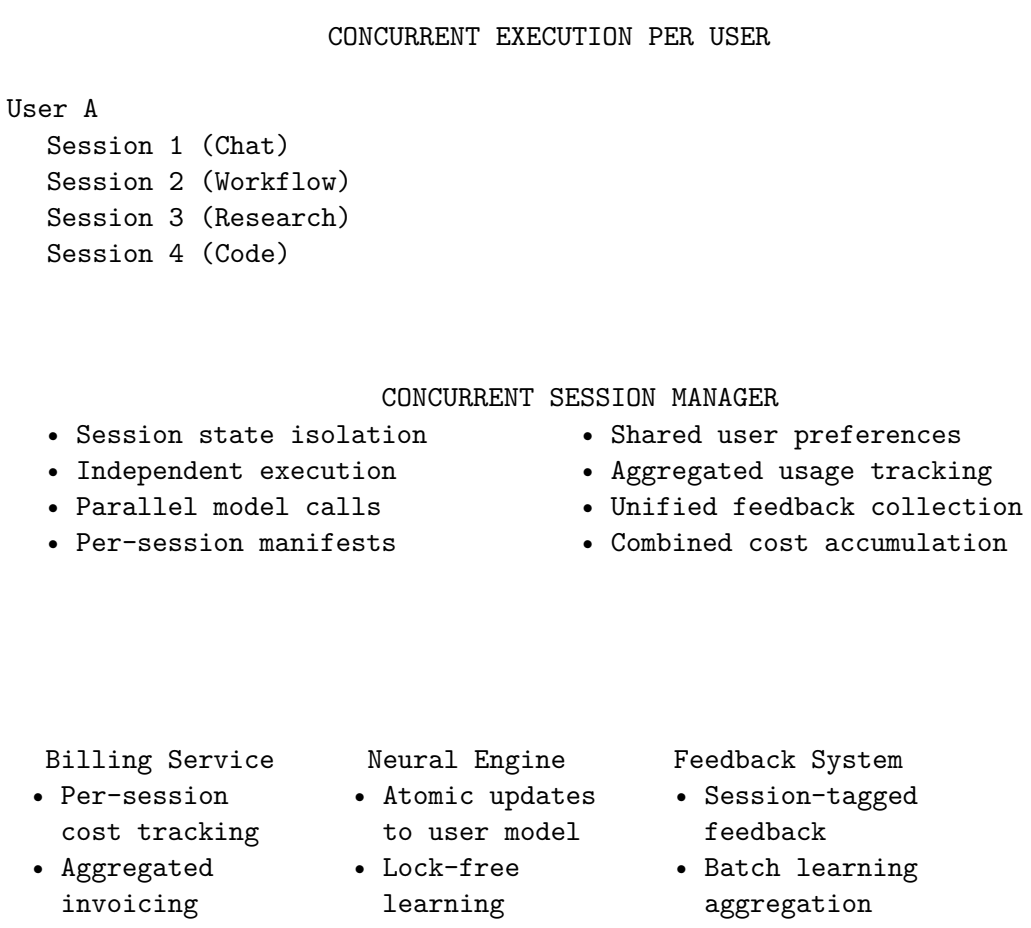
38.1 Concurrent Execution Architecture

CRITICAL: RADIANT supports concurrent execution per user across all systems.

Concurrent Execution Principles

1. **Per-User Parallelism**: Each user can run multiple AI conversations/workflows simultaneously
2. **Billing Awareness**: Usage tracking and cost accumulation work across parallel sessions
3. **Feedback Aggregation**: Feedback from concurrent sessions properly attributed and aggregated
4. **Neural Learning**: Learning signals from parallel sessions contribute to user model without race conditions
5. **Session Isolation**: Each concurrent session has independent state while sharing user preferences

Concurrent Session Architecture



38.2 Section Overview

What This Section Creates

1. **Think Tank Workflow Registry** - 127 orchestration patterns, 127 production workflows, 834 domains in database
2. **Neural-First Architecture** - Neural Engine as fabric, Brain as governor, tight integration loop
3. **Per-User Neural Models** - Personalized embeddings for preferences, domains, behavior
4. **Orchestration Constructor** - Dynamic workflow generation from primitives (not just template selection)
5. **Real-Time Steering** - Neural Engine monitors and adjusts during execution
6. **Visual Workflow Editor** - Comprehensive admin interface for building orchestrations
7. **Client Decision Transparency** - Think Tank receives reasoning, confidence, alternatives
8. **Swift Deployer v2** - Enhanced deployment app with workflow and Neural configuration
9. **Concurrent Execution Support** - Full awareness in billing, feedback, and learning systems

Design Philosophy

Principle	Current (v4.3)	New (v4.4)
Neural Role	35% weight advisor	60% weight fabric with Brain veto
Orchestration	Template selection	Dynamic construction from primitives
User Model	Global defaults	Per-user embeddings + preferences
Feedback	Post-hoc batch learning	Real-time steering + continuous learning
Admin Control	Model selection only	Full parameter visibility and editing
Client Transparency	Result only	Reasoning + confidence + alternatives
Concurrent Support	Implicit	Explicit per-session tracking

38.3 Database Schema: Think Tank Workflow Registry

migrations/038_neural_orchestration_registry.sql

```
-- ============================================================================
-- RADIANT v4.4.0 - Neural-First Orchestration & Think Tank Workflow Registry
-- ============================================================================

-- Enable pgvector if not already enabled
CREATE EXTENSION IF NOT EXISTS vector;

-- ============================================================================
-- PART 1: THINK TANK ORCHESTRATION PATTERNS (127 patterns)
```

```sql
-- ============================================================================

-- Orchestration pattern categories
CREATE TABLE IF NOT EXISTS orchestration_pattern_categories (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    category_id VARCHAR(50) UNIQUE NOT NULL,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    display_order INTEGER DEFAULT 0,
    pattern_count INTEGER DEFAULT 0,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Core orchestration patterns (127 total from Think Tank Compendium)
CREATE TABLE IF NOT EXISTS orchestration_patterns (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pattern_id VARCHAR(100) UNIQUE NOT NULL,
    category_id VARCHAR(50) NOT NULL REFERENCES orchestration_pattern_categories(category_id),

    -- Pattern metadata
    name VARCHAR(200) NOT NULL,
    description TEXT NOT NULL,
    research_basis TEXT,                          -- Academic source if applicable

    -- Implementation details
    complexity VARCHAR(20) NOT NULL CHECK (complexity IN ('low', 'medium', 'high', 'very_high'
    impact VARCHAR(20) NOT NULL CHECK (impact IN ('low', 'medium', 'high', 'transformative')),
    implemented_by TEXT[],                        -- Competitors who implement this
    implementation_status VARCHAR(30) DEFAULT 'planned',

    -- Semantic embedding for Neural Engine matching
    semantic_embedding VECTOR(768),

    -- Execution characteristics
    execution_type VARCHAR(20) DEFAULT 'serial' CHECK (execution_type IN ('serial', 'parallel'
    parallelizable BOOLEAN DEFAULT FALSE,
    typical_model_count INTEGER DEFAULT 1,
    typical_latency_ms INTEGER,
    typical_cost_multiplier DECIMAL(4,2) DEFAULT 1.0,

    -- Pattern definition (DAG structure)
    pattern_definition JSONB NOT NULL,

    -- Trigger conditions
    trigger_keywords TEXT[],
    trigger_intents TEXT[],
    trigger_complexity_range NUMRANGE,
```

```
    -- Requirements
    required_capabilities TEXT[],
    min_tier INTEGER DEFAULT 1,

    -- Stats
    usage_count INTEGER DEFAULT 0,
    avg_satisfaction_score DECIMAL(3,2),

    enabled BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Insert orchestration pattern categories (10 categories)
INSERT INTO orchestration_pattern_categories (category_id, name, description, display_order, pa
('multi_model', 'Multi-Model Coordination', 'Patterns for coordinating multiple AI models', 1,
('sequential', 'Sequential & Pipeline', 'Step-by-step processing patterns', 2, 15),
('verification', 'Verification & Fact-Checking', 'Patterns for validating AI outputs', 3, 12),
('debate', 'Debate & Adversarial Review', 'Patterns for adversarial evaluation', 4, 12),
('reasoning', 'Reasoning Enhancement', 'Patterns for improving reasoning quality', 5, 15),
('agent', 'Agent Architectures', 'Multi-agent coordination patterns', 6, 18),
('tool', 'Tool Use & Integration', 'Patterns for external tool integration', 7, 10),
('memory', 'Memory & Personalization', 'Patterns for context and personalization', 8, 8),
('user_facing', 'User-Facing Workflow Features', 'Patterns for user interaction', 9, 12),
('emerging', 'Emerging & Cutting-Edge', 'Experimental and research-stage patterns', 10, 15)
ON CONFLICT (category_id) DO UPDATE SET pattern_count = EXCLUDED.pattern_count;

-- ============================================================================
-- PART 2: THINK TANK PRODUCTION WORKFLOWS (127 workflows)
-- ============================================================================

CREATE TABLE IF NOT EXISTS production_workflow_categories (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    category_id VARCHAR(50) UNIQUE NOT NULL,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    display_order INTEGER DEFAULT 0,
    workflow_count INTEGER DEFAULT 0,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE IF NOT EXISTS production_workflows (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    workflow_id VARCHAR(100) UNIQUE NOT NULL,
    category_id VARCHAR(50) NOT NULL REFERENCES production_workflow_categories(category_id),

    name VARCHAR(200) NOT NULL,
```

```
    description TEXT NOT NULL,
    primary_deliverable VARCHAR(200),
    semantic_embedding VECTOR(768),
    workflow_definition JSONB NOT NULL,
    input_schema JSONB,
    output_schema JSONB,
    complexity VARCHAR(20) DEFAULT 'medium',
    typical_duration_minutes INTEGER,

    trigger_keywords TEXT[],
    trigger_domains TEXT[],
    required_patterns TEXT[],
    required_capabilities TEXT[],
    min_tier INTEGER DEFAULT 1,

    usage_count INTEGER DEFAULT 0,
    avg_satisfaction_score DECIMAL(3,2),
    enabled BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Insert production workflow categories (12 categories)
INSERT INTO production_workflow_categories (category_id, name, description, display_order, work
('technical', 'Technical Documentation', 'Engineering and technical writing', 1, 14),
('research', 'Research & Analysis', 'Research papers and analysis reports', 2, 12),
('business', 'Business Documents', 'Business plans, proposals, reports', 3, 10),
('legal', 'Legal Documents', 'Contracts, policies, compliance', 4, 10),
('medical', 'Medical & Healthcare', 'Medical documentation and protocols', 5, 10),
('education', 'Education & Training', 'Learning materials and curricula', 6, 10),
('marketing', 'Marketing & Communications', 'Marketing content and campaigns', 7, 10),
('financial', 'Financial Documents', 'Financial reports and analysis', 8, 10),
('creative', 'Creative Production', 'Creative works and design assets', 9, 10),
('trades', 'Skilled Trades', 'Trade-specific documentation', 10, 10),
('scientific', 'Scientific Workflows', 'Scientific analysis and research', 11, 8),
('software', 'Software Development', 'Software documentation and specs', 12, 13)
ON CONFLICT (category_id) DO UPDATE SET workflow_count = EXCLUDED.workflow_count;

-- ============================================================================
-- PART 3: THINK TANK DOMAIN REGISTRY (834 domains)
-- ============================================================================

CREATE TABLE IF NOT EXISTS domain_categories (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    category_id VARCHAR(50) UNIQUE NOT NULL,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    display_order INTEGER DEFAULT 0,
```

```
    domain_count INTEGER DEFAULT 0,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE IF NOT EXISTS specialized_domains (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    domain_id VARCHAR(100) UNIQUE NOT NULL,
    category_id VARCHAR(50) NOT NULL REFERENCES domain_categories(category_id),
    parent_domain_id VARCHAR(100) REFERENCES specialized_domains(domain_id),

    name VARCHAR(200) NOT NULL,
    description TEXT,
    semantic_embedding VECTOR(768),

    expert_context TEXT,
    terminology JSONB,
    standards TEXT[],
    best_practices TEXT[],

    keywords TEXT[],
    related_domains TEXT[],
    preferred_models TEXT[],
    preferred_patterns TEXT[],
    preferred_workflows TEXT[],

    usage_count INTEGER DEFAULT 0,
    enabled BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Insert domain categories (17 categories, 834 domains total)
INSERT INTO domain_categories (category_id, name, description, display_order, domain_count) VA
('sciences', 'Sciences', 'Physics, Chemistry, Biology, Earth Sciences, Materials', 1, 56),
('mathematics', 'Mathematics & Logic', 'Algebra, Calculus, Statistics, Logic', 2, 32),
('computer_science', 'Computer Science & Programming', 'Languages, Development, Theory', 3, 67)
('ai', 'Artificial Intelligence', 'ML, NLP, Computer Vision, AI Systems', 4, 48),
('engineering', 'Engineering', 'Mechanical, Electrical, Civil, Chemical', 5, 72),
('medicine', 'Medicine & Healthcare', 'Clinical, Surgery, Allied Health', 6, 89),
('mental_health', 'Mental Health & Psychology', 'Clinical, Counseling, Therapy', 7, 34),
('fitness', 'Fitness, Therapy & Nutrition', 'Training, Nutrition, Wellness', 8, 42),
('humanities', 'Humanities', 'Philosophy, History, Literature, Arts', 9, 68),
('social_sciences', 'Social Sciences', 'Sociology, Economics, Political Science', 10, 41),
('business', 'Business & Management', 'Strategy, Finance, Marketing, Operations', 11, 58),
('legal', 'Law & Legal', 'Corporate, IP, Employment, Litigation', 12, 34),
('education', 'Education', 'Curriculum, Assessment, Special Ed', 13, 29),
('trades', 'Skilled Trades', 'Construction, Mechanical, Electrical, Automotive', 14, 44),
('arts_design', 'Arts, Design & Creativity', 'Visual, Performing, Digital', 15, 52),
('communication', 'Communication & Media', 'Journalism, PR, Social Media', 16, 26),
```

```
('emerging', 'Interdisciplinary & Emerging', 'Sustainability, AI Ethics, Futures', 17, 31)
ON CONFLICT (category_id) DO UPDATE SET domain_count = EXCLUDED.domain_count;

-- ============================================================================
-- PART 4: PER-USER NEURAL MODELS
-- ============================================================================

CREATE TABLE IF NOT EXISTS user_neural_models (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,

    -- Profile embeddings (768-dim vectors)
    profile_embedding VECTOR(768),
    query_style_embedding VECTOR(768),
    feedback_pattern_embedding VECTOR(768),

    -- Structured preferences
    model_preferences JSONB DEFAULT '{}',
    workflow_preferences JSONB DEFAULT '{
        "prefersVerification": 0.5,
        "toleratesLatency": 0.5,
        "costSensitivity": 0.5,
        "prefersExplanation": 0.5,
        "prefersDetailedResponses": 0.5,
        "prefersStructuredOutput": 0.5
    }',
    quality_thresholds JSONB DEFAULT '{
        "minimumAcceptableQuality": 0.6,
        "qualityVsSpeedTradeoff": 0.5,
        "qualityVsCostTradeoff": 0.5
    }',

    -- Behavioral patterns
    behavioral_patterns JSONB DEFAULT '{
        "typicalQueryLength": "medium",
        "typicalComplexity": "moderate",
        "frequentDomains": [],
        "frequentWorkflows": [],
        "peakUsageHours": [],
        "feedbackFrequency": 0.5
    }',

    -- Confidence and training stats
    overall_confidence DECIMAL(5,4) DEFAULT 0,
    total_interactions INTEGER DEFAULT 0,
    total_feedback_events INTEGER DEFAULT 0,
    training_sample_count INTEGER DEFAULT 0,
```

```
    -- Admin overrides
    admin_overrides JSONB DEFAULT '{
        "forceWorkflow": null,
        "forceModel": null,
        "disablePersonalization": false,
        "customParameters": {}
    }',

    last_training_at TIMESTAMPTZ,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),

    UNIQUE(user_id, tenant_id)
);

-- Domain-specific embeddings per user
CREATE TABLE IF NOT EXISTS user_domain_embeddings (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_model_id UUID NOT NULL REFERENCES user_neural_models(id) ON DELETE CASCADE,
    domain_id VARCHAR(100) NOT NULL,
    embedding VECTOR(768),
    confidence DECIMAL(5,4) DEFAULT 0,
    sample_count INTEGER DEFAULT 0,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),
    UNIQUE(user_model_id, domain_id)
);

-- ============================================================================
-- PART 5: NEURAL ENGINE CONFIGURATION (Admin Editable)
-- ============================================================================

CREATE TABLE IF NOT EXISTS neural_engine_config (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID REFERENCES tenants(id) ON DELETE CASCADE,

    config JSONB NOT NULL DEFAULT '{
        "learning": {
            "learningRate": 0.01,
            "userModelUpdateFrequency": "realtime",
            "minSamplesForPersonalization": 20,
            "confidenceDecayRate": 0.001
        },
        "userModel": {
            "embeddingDimension": 768,
            "minConfidenceThreshold": 0.6,
            "coldStartStrategy": "global_defaults"
```

```
        },
        "construction": {
            "maxNodesPerWorkflow": 20,
            "preferTemplates": true,
            "templateMatchThreshold": 0.8
        },
        "monitoring": {
            "realTimeSteeringEnabled": true,
            "anomalyDetectionSensitivity": 0.7,
            "maxSteeringActionsPerExecution": 3
        },
        "modelSelection": {
            "neuralWeightInScoring": 0.6,
            "fallbackToDefaults": true
        },
        "scope": {
            "enableGlobalLearning": true,
            "enableTenantLearning": true,
            "enableUserLearning": true,
            "globalLearningWeight": 0.2,
            "tenantLearningWeight": 0.3,
            "userLearningWeight": 0.5
        }
    }',

    created_by UUID REFERENCES administrators(id),
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),

    UNIQUE(tenant_id)
);

-- ============================================================================
-- PART 6: BRAIN GOVERNOR CONFIGURATION (Admin Editable)
-- ============================================================================

CREATE TABLE IF NOT EXISTS brain_config (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID REFERENCES tenants(id) ON DELETE CASCADE,

    config JSONB NOT NULL DEFAULT '{
        "costs": {
            "maxCostPerRequest": 0.50,
            "maxCostPerHour": 10.00,
            "maxCostPerDay": 100.00,
            "costAlertThreshold": 0.8
        },
        "latency": {
```

```
            "maxLatencyMs": 30000,
            "latencyWarningThreshold": 0.7
        },
        "quality": {
            "minimumConfidenceThreshold": 0.5,
            "requireVerificationAboveComplexity": 0.8,
            "maxRetriesPerNode": 3
        },
        "neuralTrust": {
            "trustThreshold": 0.7,
            "autoApproveKnownWorkflows": true,
            "requireHumanApprovalBelow": 0.3
        },
        "steering": {
            "allowModelSwitching": true,
            "allowNodeSkipping": false,
            "allowWorkflowModification": true
        },
        "compliance": {
            "requireHIPAAForMedical": true,
            "requireAuditLogging": true,
            "piiDetectionEnabled": true
        }
    }',

    created_by UUID REFERENCES administrators(id),
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),

    UNIQUE(tenant_id)
);

-- Brain policies and decision audit
CREATE TABLE IF NOT EXISTS brain_policies (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    name VARCHAR(200) NOT NULL,
    description TEXT,
    policy_type VARCHAR(50) NOT NULL,
    conditions JSONB NOT NULL DEFAULT '[]',
    actions JSONB NOT NULL DEFAULT '[]',
    enabled BOOLEAN DEFAULT TRUE,
    priority INTEGER DEFAULT 0,
    created_by UUID REFERENCES administrators(id),
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE IF NOT EXISTS brain_decisions (
```

```
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    execution_id UUID NOT NULL,
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    session_id UUID REFERENCES concurrent_sessions(id),
    decision_type VARCHAR(50) NOT NULL,
    proposal JSONB NOT NULL,
    decision JSONB NOT NULL,
    approved BOOLEAN NOT NULL,
    modifications JSONB,
    guardrails_applied JSONB,
    rejection_reason TEXT,
    decided_by VARCHAR(50) NOT NULL,
    admin_id UUID REFERENCES administrators(id),
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- ============================================================================
-- PART 7: CONSTRUCTED WORKFLOWS & NEURAL EVENTS
-- ============================================================================

CREATE TABLE IF NOT EXISTS constructed_workflows (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    name VARCHAR(200),
    description TEXT,
    nodes JSONB NOT NULL,
    edges JSONB NOT NULL,
    generation_method VARCHAR(50) NOT NULL,
    source_template_id UUID REFERENCES workflow_definitions(id),
    source_pattern_ids UUID[],
    auto_metadata JSONB DEFAULT '{}',
    admin_metadata_overrides JSONB DEFAULT '{}',
    reasoning JSONB DEFAULT '{}',
    alternatives JSONB DEFAULT '[]',
    brain_approved BOOLEAN DEFAULT TRUE,
    brain_modifications JSONB,
    usage_count INTEGER DEFAULT 0,
    avg_satisfaction_score DECIMAL(3,2),
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE IF NOT EXISTS neural_execution_events (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    execution_id UUID NOT NULL,
    node_id VARCHAR(100),
    session_id UUID REFERENCES concurrent_sessions(id),
    event_type VARCHAR(50) NOT NULL,
```

```
    event_data JSONB NOT NULL,
    brain_notified BOOLEAN DEFAULT FALSE,
    brain_response JSONB,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE IF NOT EXISTS neural_learning_signals (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    signal_type VARCHAR(50) NOT NULL,
    user_id UUID REFERENCES users(id),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    workflow_id UUID,
    model_id VARCHAR(100),
    node_id VARCHAR(100),
    session_id UUID REFERENCES concurrent_sessions(id),
    signal_value DECIMAL(5,4) NOT NULL,
    confidence DECIMAL(5,4) DEFAULT 1.0,
    weight DECIMAL(5,4) DEFAULT 1.0,
    source VARCHAR(50) NOT NULL,
    processed BOOLEAN DEFAULT FALSE,
    processed_at TIMESTAMPTZ,
    created_at TIMESTAMPTZ DEFAULT NOW()
);

-- ================================================================================
-- PART 8: INDEXES
-- ================================================================================

CREATE INDEX IF NOT EXISTS idx_orchestration_patterns_category ON orchestration_patterns(categ
CREATE INDEX IF NOT EXISTS idx_orchestration_patterns_enabled ON orchestration_patterns(enable
CREATE INDEX IF NOT EXISTS idx_production_workflows_category ON production_workflows(category_
CREATE INDEX IF NOT EXISTS idx_specialized_domains_category ON specialized_domains(category_id
CREATE INDEX IF NOT EXISTS idx_user_neural_models_user ON user_neural_models(user_id);
CREATE INDEX IF NOT EXISTS idx_user_neural_models_tenant ON user_neural_models(tenant_id);

-- Vector similarity indexes
CREATE INDEX IF NOT EXISTS idx_orchestration_patterns_embedding ON orchestration_patterns
    USING ivfflat (semantic_embedding vector_cosine_ops) WITH (lists = 50);
CREATE INDEX IF NOT EXISTS idx_production_workflows_embedding ON production_workflows
    USING ivfflat (semantic_embedding vector_cosine_ops) WITH (lists = 50);
CREATE INDEX IF NOT EXISTS idx_specialized_domains_embedding ON specialized_domains
    USING ivfflat (semantic_embedding vector_cosine_ops) WITH (lists = 100);

-- Neural event indexes for concurrent execution
CREATE INDEX IF NOT EXISTS idx_neural_execution_events_session ON neural_execution_events(sessi
CREATE INDEX IF NOT EXISTS idx_neural_learning_signals_session ON neural_learning_signals(sessi
CREATE INDEX IF NOT EXISTS idx_brain_decisions_session ON brain_decisions(session_id);
```

```sql
-- ================================================================================
-- PART 9: ROW LEVEL SECURITY
-- ================================================================================

ALTER TABLE user_neural_models ENABLE ROW LEVEL SECURITY;
ALTER TABLE brain_policies ENABLE ROW LEVEL SECURITY;
ALTER TABLE brain_decisions ENABLE ROW LEVEL SECURITY;
ALTER TABLE constructed_workflows ENABLE ROW LEVEL SECURITY;

CREATE POLICY user_neural_models_isolation ON user_neural_models
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY brain_policies_isolation ON brain_policies
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY brain_decisions_isolation ON brain_decisions
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);
CREATE POLICY constructed_workflows_isolation ON constructed_workflows
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

-- ================================================================================
-- PART 10: HELPER FUNCTIONS
-- ================================================================================

-- Function: Get user's neural model with fallbacks
CREATE OR REPLACE FUNCTION get_user_neural_model(p_user_id UUID, p_tenant_id UUID)
RETURNS JSONB AS $$
DECLARE
    user_model JSONB;
BEGIN
    SELECT jsonb_build_object(
        'userId', user_id,
        'modelPreferences', model_preferences,
        'workflowPreferences', workflow_preferences,
        'qualityThresholds', quality_thresholds,
        'behavioralPatterns', behavioral_patterns,
        'overallConfidence', overall_confidence,
        'adminOverrides', admin_overrides
    ) INTO user_model
    FROM user_neural_models
    WHERE user_id = p_user_id AND tenant_id = p_tenant_id;

    RETURN COALESCE(user_model, '{}');
END;
$$ LANGUAGE plpgsql;

-- Function: Record learning signal (atomic for concurrent execution)
CREATE OR REPLACE FUNCTION record_learning_signal(
    p_signal_type VARCHAR(50),
    p_user_id UUID,
```

```
    p_tenant_id UUID,
    p_workflow_id UUID,
    p_model_id VARCHAR(100),
    p_session_id UUID,
    p_signal_value DECIMAL,
    p_source VARCHAR(50)
)
RETURNS UUID AS $$
DECLARE
    signal_id UUID;
BEGIN
    INSERT INTO neural_learning_signals (
        signal_type, user_id, tenant_id, workflow_id, model_id,
        session_id, signal_value, source
    ) VALUES (
        p_signal_type, p_user_id, p_tenant_id, p_workflow_id, p_model_id,
        p_session_id, p_signal_value, p_source
    ) RETURNING id INTO signal_id;

    RETURN signal_id;
END;
$$ LANGUAGE plpgsql;

-- Add session_id to execution_manifests for concurrent tracking
DO $$
BEGIN
    IF NOT EXISTS (
        SELECT 1 FROM information_schema.columns
        WHERE table_name = 'execution_manifests' AND column_name = 'session_id'
    ) THEN
        ALTER TABLE execution_manifests ADD COLUMN session_id UUID REFERENCES concurrent_sessio
        CREATE INDEX idx_execution_manifests_session ON execution_manifests(session_id);
    END IF;
END $$;

-- Statistics view
CREATE OR REPLACE VIEW neural_orchestration_stats AS
SELECT
    (SELECT COUNT(*) FROM orchestration_patterns WHERE enabled = TRUE) as total_patterns,
    (SELECT COUNT(*) FROM orchestration_patterns WHERE implementation_status = 'implemented') a
    (SELECT COUNT(*) FROM production_workflows WHERE enabled = TRUE) as total_workflows,
    (SELECT COUNT(*) FROM specialized_domains WHERE enabled = TRUE) as total_domains,
    (SELECT COUNT(*) FROM user_neural_models) as total_user_models,
    (SELECT AVG(overall_confidence) FROM user_neural_models) as avg_user_confidence,
    (SELECT COUNT(*) FROM constructed_workflows) as total_constructed_workflows,
    (SELECT COUNT(*) FROM neural_execution_events WHERE created_at > NOW() - INTERVAL '24 hours
```

38.4 Seed Data: Sample Orchestration Patterns

migrations/038b_seed_orchestration_patterns.sql

```
-- Sample of 127 Orchestration Patterns from Think Tank Compendium
-- Full seed data includes all 127 patterns

-- Multi-Model Coordination Patterns (10)
INSERT INTO orchestration_patterns (pattern_id, category_id, name, description, complexity, imp
VALUES
('side_by_side_comparison', 'multi_model', 'Side-by-Side Comparison', 'Send identical prompt to
('consensus_voting', 'multi_model', 'Consensus Voting', 'Query multiple models, aggregate via m
('ensemble_synthesis', 'multi_model', 'Ensemble Response Synthesis', 'Combine responses from mu
('intelligent_routing', 'multi_model', 'Intelligent Model Routing', 'Auto-select optimal model
ON CONFLICT (pattern_id) DO UPDATE SET updated_at = NOW();

-- Sequential Patterns (sample)
INSERT INTO orchestration_patterns (pattern_id, category_id, name, description, complexity, imp
VALUES
('draft_critique_revise', 'sequential', 'Draft → Critique → Revise', 'Generate, evaluate, impro
('research_analyze_report', 'sequential', 'Research → Analyze → Report', 'Sequential research w
('plan_execute_verify', 'sequential', 'Plan → Execute → Verify', 'Agentic task completion patte
ON CONFLICT (pattern_id) DO UPDATE SET updated_at = NOW();

-- Verification Patterns (sample)
INSERT INTO orchestration_patterns (pattern_id, category_id, name, description, complexity, imp
VALUES
('red_team_attack', 'verification', 'Red Team Attack', 'Adversarial model challenges primary re
('chain_of_verification', 'verification', 'Chain of Verification (CoVe)', 'Generate claims → ge
('hallucination_detection', 'verification', 'Hallucination Detection', 'Identify unsupported or
ON CONFLICT (pattern_id) DO UPDATE SET updated_at = NOW();

-- Debate Patterns (sample)
INSERT INTO orchestration_patterns (pattern_id, category_id, name, description, complexity, imp
VALUES
('ai_debate', 'debate', 'AI Debate', 'Two models argue opposing positions', 'medium', 'high', A
('round_table_consensus', 'debate', 'Round Table Consensus', 'Multiple agents discuss to reach
('ai_judge', 'debate', 'AI Judge', 'Third model evaluates debate outcome', 'medium', 'high', AI
ON CONFLICT (pattern_id) DO UPDATE SET updated_at = NOW();

-- Reasoning Enhancement Patterns (sample)
INSERT INTO orchestration_patterns (pattern_id, category_id, name, description, complexity, imp
VALUES
('chain_of_thought', 'reasoning', 'Chain of Thought (CoT)', 'Step-by-step reasoning', 'low', 'I
('tree_of_thoughts', 'reasoning', 'Tree of Thoughts (ToT)', 'Explore multiple reasoning branche
('self_consistency', 'reasoning', 'Self-Consistency', 'Sample multiple CoT paths, vote on answe
('self_refine', 'reasoning', 'Self-Refine Loop', 'Iterative self-improvement', 'medium', 'high
ON CONFLICT (pattern_id) DO UPDATE SET updated_at = NOW();
```

```
-- Note: Full implementation includes all 127 patterns from Think Tank Compendium
```

38.5 API Endpoints Summary

Neural Orchestration Admin Endpoints

Method	Endpoint	Description
GET	`/api/v2/admin/neural/config`	Get Neural Engine config
PUT	`/api/v2/admin/neural/config`	Update Neural Engine config
GET	`/api/v2/admin/brain/config`	Get Brain Governor config
PUT	`/api/v2/admin/brain/config`	Update Brain Governor config
GET	`/api/v2/admin/patterns`	List orchestration patterns
PUT	`/api/v2/admin/patterns/{patternId}`	Update pattern status
GET	`/api/v2/admin/workflows/production`	List production workflows
GET	`/api/v2/admin/domains`	List specialized domains
GET	`/api/v2/admin/user-models`	List user neural models
PUT	`/api/v2/admin/user-models/{userId}/overrides`	Set user overrides
DELETE	`/api/v2/admin/user-models/{userId}`	Reset user model
GET	`/api/v2/admin/registry/stats`	Get registry statistics

Workflow Editor Endpoints

Method	Endpoint	Description
GET	`/api/v2/admin/workflows/templates`	List workflow templates
POST	`/api/v2/admin/workflows/templates`	Create template
PUT	`/api/v2/admin/workflows/templates/{id}`	Update template
DELETE	`/api/v2/admin/workflows/templates/{id}`	Delete template
POST	`/api/v2/admin/workflows/generate-metadata`	Auto-generate metadata
POST	`/api/v2/admin/workflows/test`	Test workflow execution

Client Decision Transparency Endpoints

Method	Endpoint	Description
GET	`/api/v2/decision/{executionId}`	Get decision transparency
POST	`/api/v2/decision/{executionId}/override`	Submit override request

38.6 Swift Deployer v2 Updates

The Swift Deployment App now includes:

1. **Neural Engine Configuration View**

- Learning parameters (rate, frequency, decay)
- User model settings (cold start, confidence thresholds)
- Construction parameters (max nodes, template preferences)
- Real-time monitoring toggles

2. **Brain Governor Configuration View**
   - Cost controls (per-request, per-hour, per-day limits)
   - Latency controls (max latency, warning thresholds)
   - Quality controls (confidence, verification requirements)
   - Neural trust settings (approval thresholds)
   - Steering controls (model switching, node skipping)
   - Compliance settings (HIPAA, audit logging)
3. **Workflow Registry Browser**
   - View 127 orchestration patterns by category
   - View 127 production workflows by category
   - View 834 specialized domains by category
   - Search and filter capabilities
   - Usage statistics dashboard

38.7 Verification & Deployment

Pre-Deployment Checklist

```
# 1. Apply database migrations
psql $DATABASE_URL -f migrations/038_neural_orchestration_registry.sql
psql $DATABASE_URL -f migrations/038b_seed_orchestration_patterns.sql

# 2. Verify tables created
psql $DATABASE_URL -c "SELECT * FROM neural_orchestration_stats"

# 3. Verify registry counts
psql $DATABASE_URL -c "SELECT COUNT(*) as patterns FROM orchestration_patterns"
psql $DATABASE_URL -c "SELECT COUNT(*) as categories FROM domain_categories"

# 4. Test Neural Engine config
curl https://api.YOUR_DOMAIN.com/api/v2/admin/neural/config \
  -H "Authorization: Bearer $ADMIN_TOKEN"

# 5. Test decision transparency
curl https://api.YOUR_DOMAIN.com/api/v2/decision/{executionId} \
  -H "Authorization: Bearer $TOKEN"
```

Summary v4.4.0

RADIANT v4.4.0 (PROMPT-18) adds **Neural-First Orchestration & Think Tank Workflow Registry**:

Section 38: Neural-First Orchestration (v4.4.0)

1. **Think Tank Workflow Registry** (Database Schema)
   - 127 orchestration patterns across 10 categories
   - 127 production workflows across 12 categories
   - 834 specialized domains across 17 categories
   - Full semantic embeddings for Neural Engine matching
2. **Neural Engine Enhancements**
   - Per-user neural models with 768-dim embeddings
   - Orchestration constructor (dynamic workflow generation)
   - Real-time steering during execution
   - Learning from concurrent sessions (atomic updates)
3. **Brain Governor Enhancements**
   - Full admin-editable configuration
   - Policy engine with conditions and actions
   - Decision audit logging
   - Concurrent session awareness
4. **Visual Workflow Editor**
   - Drag-and-drop node placement
   - 12 node type categories
   - Neural I/O connector visualization
   - Auto-generated metadata
5. **Client Decision Transparency**
   - Reasoning exposed to Think Tank
   - Confidence scores with explanations
   - Alternatives with tradeoffs
   - Override options with impact estimation
6. **Swift Deployer v2**
   - Neural Engine configuration UI
   - Brain Governor configuration UI
   - Workflow registry browser
7. **Concurrent Execution Support**
   - Session-aware billing aggregation
   - Session-tagged feedback
   - Atomic neural model updates

Design Philosophy (v4.4.0)

- **Neural-First** - Neural Engine is the fabric (60% weight), Brain is the governor
- **Dynamic Construction** - Workflows built from primitives, not just template selection
- **Per-User Learning** - Personalized embeddings and preferences
- **Real-Time Steering** - Adjustments during execution, not just post-hoc
- **Full Transparency** - Clients see reasoning, confidence, alternatives
- **Admin Control** - All parameters editable through dashboard
- **Concurrent Aware** - Proper handling of parallel user sessions

Also includes all v4.3.0 features:

- Feedback Learning System
- Neural Engine Loop
- Multi-Language Voice Feedback
- Implicit Signals
- A/B Testing Framework

Total Sections: 40 (0-39) Total Lines: ~53,000 Total Size: ~2.3MB
