# Contents



# RADIANT v4.18.0 - Database Migrations Export

**Component**: PostgreSQL Database Schema **Database**: Aurora PostgreSQL 15 **Files**: 160+ SQL migration files **Features**: Row-Level Security (RLS), pgvector, Multi-tenancy

---

## Architecture Narrative

RADIANT uses **Aurora PostgreSQL Serverless v2** with a comprehensive migration system. All tables implement **Row-Level Security (RLS)** for multi-tenant isolation, using the `app.current_tenant_id` session variable set at the start of each request.

### Key Database Features

1. **Multi-Tenant Isolation** - RLS policies on all tables
2. **Vector Search** - pgvector extension for embeddings
3. **Audit Logging** - Append-only audit tables
4. **Versioned Migrations** - Sequential numbered migrations
5. **Seed Data** - Default configurations and models

### Migration Naming Convention

`XXX_feature_name.sql`

```
Where XXX is a sequential number:
001-009: Core platform schema
010-019: AI and Brain features
020-029: Think Tank and collaboration
030-039: Configuration and billing
040-049: Advanced AI features
050-099: Platform extensions
100+: Latest additions
```

---

## Migration Inventory

### Core Platform (001-009)

| Migration | Purpose |
|---|---|
| `001_initial_schema.sql` | Base tables: tenants, users, administrators |
| `002_tenant_isolation.sql` | RLS policies for multi-tenancy |
| `003_ai_models.sql` | AI model registry tables |
| `004_usage_billing.sql` | Usage tracking and billing |
| `005_admin_approval.sql` | Two-person approval workflow |
| `006_self_hosted_models.sql` | Self-hosted model management |
| `007_external_providers.sql` | External provider configuration |

### AI & Brain Features (010-019)

| Migration | Purpose |
|---|---|
| `010_visual_ai_pipeline.sql` | Visual AI processing |
| `011_brain_router.sql` | AGI Brain router tables |
| `012_metrics_analytics.sql` | Metrics and analytics |
| `013_neural_engine.sql` | Neural engine configuration |
| `014_error_logging.sql` | Error logging tables |
| `015_credentials_registry.sql` | Credential management |
| `016_think_tank.sql` | Think Tank sessions and steps |
| `017_concurrent_chat.sql` | Concurrent chat support |
| `018_realtime_collaboration.sql` | Real-time collaboration |
| `019_persistent_memory.sql` | Persistent memory storage |

### Think Tank & Collaboration (020-029)

| Migration | Purpose |
|---|---|
| `020_focus_personas.sql` | Focus persona management |
| `021_team_plans.sql` | Team planning features |
| `022_provider_registry.sql` | Provider registry |
| `023_time_machine.sql` | Time Machine versioning |

| Migration | Purpose |
|---|---|
| 024_orchestration_engine.sql | Orchestration engine |
| 025_license_management.sql | License management |
| 026_unified_model_registry.sql | Unified model registry |
| 027_feedback_learning.sql | Feedback and learning |
| 028_neural_orchestration.sql | Neural orchestration |
| 029_workflow_proposals.sql | Workflow proposals |

**Configuration & Billing (030-039)**

| Migration | Purpose |
|---|---|
| 030_app_isolation.sql | App-level isolation |
| 031_internationalization.sql | i18n support |
| 032_dynamic_configuration.sql | Dynamic config |
| 032b_artifact_genui_engine.sql | Artifact generation UI |
| 032c_artifact_genui_seed.sql | Artifact UI seed data |
| 032d_artifact_extend_base.sql | Artifact extensions |
| 033_billing_credits.sql | Credit-based billing |
| 034_storage_billing.sql | Storage billing |
| 035_versioned_subscriptions.sql | Subscription versioning |
| 036_dual_admin_approval.sql | Dual admin approval |
| 037_canvas_artifacts.sql | Canvas artifacts |
| 038_scheduled_prompts.sql | Scheduled prompts |

**Advanced AI Features (040-099)**

| Migration | Purpose |
|---|---|
| 040_agi_brain_plans.sql | AGI brain planning |
| 041_consciousness_engine.sql | Consciousness engine |
| 042_formal_reasoning.sql | Formal reasoning |
| 043_ego_state.sql | Ego state management |
| 044_dreaming.sql | Dream cycle storage |
| 045_ghost_vectors.sql | Ghost vector storage |
| 050_grimoire_governor.sql | Grimoire & Governor |
| ... | Additional features |

**Platform Extensions (100+)**

| Migration | Purpose |
|---|---|
| 100_cato_safety.sql | Cato safety system |
| 101_ecd_verification.sql | ECD truth verification |
| 102_ego_state_injection.sql | Zero-cost ego system |
| 103_metrics_learning.sql | Metrics-based learning |

| Migration | Purpose |
|---|---|
| ... | Additional extensions |

---

## Key Migrations

### 001_initial_schema.sql

**Purpose**: Creates base tables for the multi-tenant platform.

```sql
-- RADIANT v4.17.0 - Migration 001: Initial Schema
-- Creates base tables for multi-tenant platform

-- Enable required extensions
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
CREATE EXTENSION IF NOT EXISTS "pgcrypto";

-- ============================================================================
-- TENANTS TABLE
-- ============================================================================

CREATE TABLE tenants (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    name VARCHAR(100) NOT NULL UNIQUE,
    display_name VARCHAR(200) NOT NULL,
    domain VARCHAR(255),
    settings JSONB NOT NULL DEFAULT '{}',
    status VARCHAR(20) NOT NULL DEFAULT 'active'
        CHECK (status IN ('active', 'suspended', 'pending')),
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_tenants_name ON tenants(name);
CREATE INDEX idx_tenants_domain ON tenants(domain) WHERE domain IS NOT NULL;
CREATE INDEX idx_tenants_status ON tenants(status);

-- ============================================================================
-- USERS TABLE (End Users)
-- ============================================================================

CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    cognito_user_id VARCHAR(128) NOT NULL,
    email VARCHAR(255) NOT NULL,
    display_name VARCHAR(200),
```

```
    role VARCHAR(50) NOT NULL DEFAULT 'user'
        CHECK (role IN ('user', 'power_user', 'admin')),
    status VARCHAR(20) NOT NULL DEFAULT 'active'
        CHECK (status IN ('active', 'suspended', 'pending')),
    settings JSONB NOT NULL DEFAULT '{}',
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    UNIQUE (tenant_id, email),
    UNIQUE (cognito_user_id)
);

CREATE INDEX idx_users_tenant_id ON users(tenant_id);
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_cognito_user_id ON users(cognito_user_id);

-- ============================================================================
-- ADMINISTRATORS TABLE (Platform Admins)
-- ============================================================================

CREATE TABLE administrators (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    cognito_user_id VARCHAR(128) NOT NULL UNIQUE,
    email VARCHAR(255) NOT NULL UNIQUE,
    display_name VARCHAR(200) NOT NULL,
    role VARCHAR(50) NOT NULL DEFAULT 'admin'
        CHECK (role IN ('super_admin', 'admin', 'operator', 'auditor')),
    permissions TEXT[] NOT NULL DEFAULT '{}',
    mfa_enabled BOOLEAN NOT NULL DEFAULT false,
    last_login_at TIMESTAMPTZ,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    invited_by UUID REFERENCES administrators(id)
);

-- ============================================================================
-- APPROVAL REQUESTS TABLE (Two-Person Approval)
-- ============================================================================

CREATE TABLE approval_requests (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    requester_id UUID NOT NULL REFERENCES administrators(id),
    action_type VARCHAR(100) NOT NULL,
    resource_type VARCHAR(100) NOT NULL,
    resource_id VARCHAR(255),
    payload JSONB NOT NULL DEFAULT '{}',
    status VARCHAR(20) NOT NULL DEFAULT 'pending'
        CHECK (status IN ('pending', 'approved', 'rejected', 'expired')),
    approved_by UUID REFERENCES administrators(id),
```

```
    approved_at TIMESTAMPTZ,
    rejection_reason TEXT,
    expires_at TIMESTAMPTZ NOT NULL,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);
```

---

**016_think_tank.sql**

**Purpose**: Think Tank platform for complex problem decomposition.

```
-- RADIANT v4.17.0 - Migration 016: Think Tank Platform
-- Complex problem decomposition and multi-step reasoning

CREATE TABLE thinktank_sessions (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    problem_summary TEXT,
    domain VARCHAR(50),
    complexity VARCHAR(20) CHECK (complexity IN ('low', 'medium', 'high', 'extreme')),
    total_steps INTEGER DEFAULT 0,
    avg_confidence DECIMAL(3, 2),
    solution_found BOOLEAN DEFAULT false,
    total_tokens INTEGER DEFAULT 0,
    total_cost DECIMAL(10, 6) DEFAULT 0,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    completed_at TIMESTAMPTZ
);

CREATE TABLE thinktank_steps (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    session_id UUID NOT NULL REFERENCES thinktank_sessions(id) ON DELETE CASCADE,
    step_number INTEGER NOT NULL,
    step_type VARCHAR(50) NOT NULL
        CHECK (step_type IN ('decompose', 'reason', 'execute', 'verify', 'synthesize')),
    description TEXT,
    reasoning TEXT,
    result TEXT,
    confidence DECIMAL(3, 2) CHECK (confidence >= 0 AND confidence <= 1),
    model_used VARCHAR(100),
    tokens_used INTEGER,
    duration_ms INTEGER,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE TABLE thinktank_tools (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
```

```
    tool_name VARCHAR(100) NOT NULL UNIQUE,
    tool_type VARCHAR(50) NOT NULL,
    description TEXT,
    parameters_schema JSONB NOT NULL DEFAULT '{}',
    implementation TEXT,
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

-- Indexes
CREATE INDEX idx_thinktank_sessions_tenant ON thinktank_sessions(tenant_id, created_at DESC);
CREATE INDEX idx_thinktank_sessions_user ON thinktank_sessions(user_id);
CREATE INDEX idx_thinktank_steps_session ON thinktank_steps(session_id, step_number);

-- Row Level Security
ALTER TABLE thinktank_sessions ENABLE ROW LEVEL SECURITY;
ALTER TABLE thinktank_steps ENABLE ROW LEVEL SECURITY;

CREATE POLICY thinktank_sessions_isolation ON thinktank_sessions
    FOR ALL USING (tenant_id = current_setting('app.current_tenant_id', true)::uuid);

CREATE POLICY thinktank_steps_isolation ON thinktank_steps
    FOR ALL USING (
        session_id IN (
            SELECT id FROM thinktank_sessions
            WHERE tenant_id = current_setting('app.current_tenant_id', true)::uuid
        )
    );

-- Default tools
INSERT INTO thinktank_tools (tool_name, tool_type, description, parameters_schema) VALUES
    ('web_search', 'search', 'Search the web for information', '{"query": "string"}'),
    ('calculator', 'compute', 'Perform mathematical calculations', '{"expression": "string"}')
    ('code_executor', 'compute', 'Execute code snippets', '{"language": "string", "code": "str
    ('file_reader', 'io', 'Read file contents', '{"path": "string"}'),
    ('api_caller', 'network', 'Make API requests', '{"url": "string", "method": "string", "bod
```

---

**050_grimoire_governor.sql**

**Purpose**: Grimoire procedural memory and Economic Governor.

```
-- RADIANT v5.0.0 - Migration 050: Grimoire & Governor
-- Procedural memory and cost optimization

-- =============================================================================
-- KNOWLEDGE HEURISTICS TABLE (The Grimoire)
-- =============================================================================
```

```sql
CREATE TABLE knowledge_heuristics (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    domain VARCHAR(100) NOT NULL,
    trigger_pattern TEXT NOT NULL,
    action_template TEXT NOT NULL,
    success_count INTEGER DEFAULT 0,
    failure_count INTEGER DEFAULT 0,
    confidence DECIMAL(3, 2) DEFAULT 0.50,
    embedding VECTOR(1536),
    source_execution_id UUID,
    last_used_at TIMESTAMPTZ,
    expires_at TIMESTAMPTZ,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

CREATE INDEX idx_heuristics_domain ON knowledge_heuristics(tenant_id, domain);
CREATE INDEX idx_heuristics_embedding ON knowledge_heuristics USING ivfflat (embedding vector_

-- RLS
ALTER TABLE knowledge_heuristics ENABLE ROW LEVEL SECURITY;

CREATE POLICY heuristics_tenant_isolation ON knowledge_heuristics
    FOR ALL USING (tenant_id = current_setting('app.current_tenant_id', true)::uuid);

-- ============================================================================
-- GOVERNOR SAVINGS LOG
-- ============================================================================

CREATE TABLE governor_savings_log (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    tenant_id UUID NOT NULL REFERENCES tenants(id) ON DELETE CASCADE,
    task_id UUID NOT NULL,
    original_model VARCHAR(100) NOT NULL,
    selected_model VARCHAR(100) NOT NULL,
    complexity_score DECIMAL(3, 2) NOT NULL,
    estimated_original_cost DECIMAL(10, 6) NOT NULL,
    actual_cost DECIMAL(10, 6) NOT NULL,
    savings DECIMAL(10, 6) NOT NULL,
    decision_reason TEXT,
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

ALTER TABLE governor_savings_log ENABLE ROW LEVEL SECURITY;

CREATE POLICY governor_log_isolation ON governor_savings_log
```

```
    FOR ALL USING (tenant_id = current_setting('app.current_tenant_id', true)::uuid);
```

---

### RLS Pattern

All tenant-scoped tables use this pattern:

```
-- Enable RLS
ALTER TABLE table_name ENABLE ROW LEVEL SECURITY;

-- Create isolation policy
CREATE POLICY table_name_isolation ON table_name
    FOR ALL USING (tenant_id = current_setting('app.current_tenant_id', true)::uuid);

-- For related tables (via foreign key)
CREATE POLICY child_table_isolation ON child_table
    FOR ALL USING (
        parent_id IN (
            SELECT id FROM parent_table
            WHERE tenant_id = current_setting('app.current_tenant_id', true)::uuid
        )
    );
```

#### Setting Tenant Context

```
// In Lambda handlers
await executeStatement(
  `SELECT set_config('app.current_tenant_id', $1, true)`,
  [{ name: 'tenantId', value: { stringValue: tenantId } }]
);
```

---

### Database Extensions

```
-- Required extensions
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";     -- UUID generation
CREATE EXTENSION IF NOT EXISTS "pgcrypto";      -- Cryptographic functions
CREATE EXTENSION IF NOT EXISTS "vector";        -- pgvector for embeddings
CREATE EXTENSION IF NOT EXISTS "pg_trgm";       -- Trigram similarity
```

---

### Migration Execution

#### Fresh Install

```
# Run all migrations in order
for f in migrations/*.sql; do
  psql -h $DB_HOST -U $DB_USER -d $DB_NAME -f "$f"
done
```

### Incremental Update

```
# Run only new migrations (from version X)
for f in migrations/$(($CURRENT_VERSION + 1))_*.sql; do
  psql -h $DB_HOST -U $DB_USER -d $DB_NAME -f "$f"
done
```

---

*This concludes the Database Migrations export. See other export files for additional components.*