Database Documentation

PSScript PowerShell Analysis Platform

Last Updated: January 15, 2026 **Version:** 1.0.0

Table of Contents

Overview

PSScript uses a PostgreSQL database with Redis caching layer, managed through Sequelize ORM. The system supports: - PowerShell script storage and version control - AI-driven script analysis with security scoring - Vector embeddings for semantic search (pgvector) - User authentication with account lockout protection - Execution logging and analytics

Technology Stack

Component	Technology	Version
Database	PostgreSQL	15.x
ORM	Sequelize	^6.32.1
Cache	Redis	7.0
Vector Search	pgvector	0.8.0
Runtime	Node.js	18.x+

Port Assignments (Fixed)

Service	Port	Notes
PostgreSQL	5432	Primary database
Redis	6379	Cache layer
Backend API	4000	Express server
Frontend	3000	React app
AI Service	8000	FastAPI

Database Architecture

```
┌──────────────────────────────────────────────────────────────────
│                        APPLICATION LAYER
├──────────────────────────────────────────────────────────────────
│  Frontend (React)  │  Backend (Express)  │  AI Service (FastAPI
└──────────────────────────────────────────────────────────────────
          │                   │                   │
          │                   │                   │
          ▼                   ▼                   ▼
┌──────────────────────────────────────────────────────────────────
│                      API LAYER
│
│  ┌──────────┐  ┌──────────┐  ┌──────────┐  ┌──────────┐
│  │ Scripts  │  │   Auth   │  │ Analysis │  │Analytics │
│  │  Routes  │  │  Routes  │  │  Routes  │  │  Routes  │
│  └────┬─────┘  └────┬─────┘  └────┬─────┘  └────┬─────┘
│       │             │             │             │
└───────┼─────────────┼────────────────┼───────────────┼──────────
        │             │                │               │
        │             │                │               │
        ▼             ▼                ▼               ▼
┌──────────────────────────────────────────────────────────────────
│                   SEQUELIZE ORM LAYER
│
│  ┌────────┐  ┌────────┐  ┌──────────┐  ┌─────────────┐  ┌────────┐
│  │ User   │  │ Script │  │ Analysis │  │ Embedding   │  │ ...    │
│  │ Model  │  │ Model  │  │ Model    │  │ Model       │  │        │
│  └───┬────┘  └───┬────┘  └────┬─────┘  └──────┬──────┘  └───┬────┘
│      │           │            │               │             │
└──────┼───────────┼────────────┼───────────────┼─────────────┼────
       │           │            │               │             │
       │           │            │               │             │
       ▼           ▼            ▼               ▼             ▼
┌──────────────────────────────────────────────────────────────────
│                       DATA LAYER
│
│  ┌─────────────────────────────┐  ┌──────────────────────────┐
│  │      PostgreSQL 15          │  │         Redis 7.0        │
│  │                             │  │                          │
│  │  ┌───────────────────────┐  │  │  ┌────────────────────┐  │
│  │  │ pgvector extension    │  │  │  │ Session Cache      │  │
│  │  │ (vector similarity)   │  │  │  │ Query Cache        │  │
│   │   └────────────────────────┘   │   │  Analysis Cache      │   │
│   └──────────────────────────────┘   │   └─────────────────────┘   │
└────────────────────────────────────────────────────────────────────
```

Connection Configuration

Environment Variables

```
 # PostgreSQL
DB_HOST=localhost           # Database host (use 'postgres' in Doc
DB_PORT=5432               # Database port (DO NOT CHANGE)
DB_NAME=psscript           # Database name
DB_USER=postgres           # Database user
DB_PASSWORD=postgres       # Database password
DB_SSL=false               # Enable SSL (true in production)
DB_SSL_CA_PATH=            # Path to SSL CA certificate

# Alternative: Single connection URL
DATABASE_URL=postgresql://user:password@host:port/database

# Redis
REDIS_HOST=localhost       # Redis host (use 'redis' in Docker)
REDIS_PORT=6379           # Redis port (DO NOT CHANGE)
REDIS_URL=redis://localhost:6379
```

Connection Pool Settings

Setting	Value	Purpose
Pool Max	10	Maximum concurrent connections
Pool Min	0	Minimum idle connections
Acquire Timeout	30,000ms	Max wait for connection
Idle Timeout	10,000ms	Close idle connections after
Connection Timeout	15,000ms	Initial connection timeout

Retry Configuration

Setting	Value	Purpose
Max Retries	5	Connection retry attempts
Initial Delay	2,000ms	First retry delay
Backoff	Exponential	Delay doubles each retry
Deadlock Retries	3	Automatic deadlock recovery

Data Models

1. User

Table: `users`

Column	Type	Constraints	Description
id	SERIAL	PK	Auto-increment ID
username	VARCHAR(255)	UNIQUE, NOT NULL	User's display name
email	VARCHAR(255)	UNIQUE, NOT NULL	Email address
password_hash	VARCHAR(255)	NOT NULL	Bcrypt hashed password
role	VARCHAR(50)	DEFAULT 'user'	User role (user, admin)
last_login_at	TIMESTAMP	NULL	Last successful login
login_attempts	INTEGER	DEFAULT 0	Failed login counter
locked_until	TIMESTAMP	NULL	Account lockout expiry
created_at	TIMESTAMP	DEFAULT NOW()	Account creation time
updated_at	TIMESTAMP	DEFAULT NOW()	Last update time

Security Features: - Password hashing: bcrypt with 12 rounds (OWASP recommended) - Account lockout after 5 failed attempts - 30-minute lockout duration - Constant-time password comparison

---

2. Script

Table: `scripts`

Column	Type	Constraints	Description
id	SERIAL	PK	Auto-increment ID
title	VARCHAR(255)	NOT NULL	Script title
description	TEXT	NULL	Script description
content	TEXT	NOT NULL	PowerShell script content
user_id	INTEGER	FK → users.id	Script owner
category_id	INTEGER	FK → categories.id	Script category
version	INTEGER	DEFAULT 1	Current version number
is_public	BOOLEAN	DEFAULT false	Public visibility
execution_count	INTEGER	DEFAULT 0	Times executed
file_hash	VARCHAR(255)	NULL	MD5 hash for deduplication
average_execution_time	FLOAT	NULL	Average runtime
last_executed_at	TIMESTAMP	NULL	Last execution time
created_at	TIMESTAMP	DEFAULT NOW()	Creation time
updated_at	TIMESTAMP	DEFAULT NOW()	Last update time

Deduplication: - File hash (MD5) calculated on upload - Prevents duplicate script storage - Returns existing script if hash matches

---

3. ScriptAnalysis

Table: `script_analysis`

Column	Type	Constraints	Description
id	SERIAL	PK	Auto-increment ID
script_id	INTEGER	FK → scripts.id	Associated script
purpose	TEXT	NULL	Script purpose description
security_score	FLOAT	NULL	Security rating (0-100)
quality_score	FLOAT	NULL	Code quality rating
risk_score	FLOAT	NULL	Risk assessment
parameter_docs	JSONB	DEFAULT {}	Parameter documentation
suggestions	JSONB	DEFAULT []	Improvement suggestions
command_details	JSONB	DEFAULT []	PowerShell command analysis
ms_docs_references	JSONB	DEFAULT []	Microsoft docs links
security_issues	JSONB	DEFAULT []	Security vulnerabilities
best_practice_violations	JSONB	DEFAULT []	PSScriptAnalyzer issues
performance_insights	JSONB	DEFAULT []	Performance recommendations
potential_risks	JSONB	DEFAULT []	Execution risks
code_complexity_metrics	JSONB	DEFAULT {}	Complexity analysis
compatibility_notes	JSONB	DEFAULT []	Version compatibility
execution_summary	JSONB	DEFAULT {}	Resource usage summary
analysis_version	VARCHAR	DEFAULT '1.0'	Analysis engine version
created_at	TIMESTAMP	DEFAULT NOW()	Analysis time
updated_at	TIMESTAMP	DEFAULT NOW()	Last update time

---

4. ScriptEmbedding

Table: `script_embeddings`

Column	Type	Constraints	Description
id	SERIAL	PK	Auto-increment ID
script_id	INTEGER	FK → scripts.id, UNIQUE	Associated script
embedding	VECTOR(1536)	NOT NULL	OpenAI embedding vector
embedding_type	VARCHAR(50)	DEFAULT 'openai'	Embedding provider
model_version	VARCHAR(50)	DEFAULT 'text-embedding-ada-002'	Model used
created_at	TIMESTAMP	DEFAULT NOW()	Creation time
updated_at	TIMESTAMP	DEFAULT NOW()	Last update time

Vector Search: - Uses pgvector extension - IVFFlat index with cosine distance - 100 lists for approximate nearest neighbor search

---

5. Category

Table: `categories`

Column	Type	Constraints	Description
id	SERIAL	PK	Auto-increment ID
name	VARCHAR(255)	UNIQUE, NOT NULL	Category name
description	TEXT	NULL	Category description
created_at	TIMESTAMP	DEFAULT NOW()	Creation time
updated_at	TIMESTAMP	DEFAULT NOW()	Last update time

---

6. Tag

Table: `tags`

Column	Type	Constraints	Description
id	SERIAL	PK	Auto-increment ID
name	VARCHAR(255)	UNIQUE, NOT NULL	Tag name
created_at	TIMESTAMP	DEFAULT NOW()	Creation time

---

7. ScriptTag (Junction)

Table: `script_tags`

Column	Type	Constraints	Description
script_id	INTEGER	PK, FK → scripts.id	Script reference
tag_id	INTEGER	PK, FK → tags.id	Tag reference
created_at	TIMESTAMP	DEFAULT NOW()	Association time

---

8. ScriptVersion

Table: `script_versions`

Column	Type	Constraints	Description
id	SERIAL	PK	Auto-increment ID
script_id	INTEGER	FK → scripts.id	Parent script
content	TEXT	NOT NULL	Version content
version	INTEGER	NOT NULL	Version number
user_id	INTEGER	FK → users.id	Version creator
commit_message	TEXT	NULL	Change description
created_at	TIMESTAMP	DEFAULT NOW()	Version creation time

Unique Constraint: `(script_id, version)`

---

9. ExecutionLog

Table: `execution_logs`

Column	Type	Constraints	Description
id	SERIAL	PK	Auto-increment ID
script_id	INTEGER	FK → scripts.id	Executed script
user_id	INTEGER	FK → users.id, SET NULL	Executor
status	VARCHAR(50)	NOT NULL	Execution status
execution_time	FLOAT	NULL	Duration in ms
parameters	JSONB	DEFAULT {}	Execution parameters
error_message	TEXT	NULL	Error details
ip_address	VARCHAR(45)	NULL	Executor IP (IPv6 safe)
created_at	TIMESTAMP	DEFAULT NOW()	Execution time

Status Values: `success`, `failure`, `timeout`, `cancelled`

---

10. ChatHistory

Table: `chat_history`

Column	Type	Constraints	Description
id	SERIAL	PK	Auto-increment ID
user_id	INTEGER	FK → users.id, CASCADE	Chat owner
messages	JSONB	NOT NULL	Conversation messages
response	TEXT	NOT NULL	AI response
embedding	VECTOR(1536)	NULL	Conversation embedding
created_at	TIMESTAMP	DEFAULT NOW()	Conversation time
updated_at	TIMESTAMP	DEFAULT NOW()	Last update time

---

11. Documentation

Table: `documentation`

Column	Type	Constraints	Description
id	SERIAL	PK	Auto-increment ID
title	VARCHAR(500)	NOT NULL	Document title
url	VARCHAR(2048)	UNIQUE, NOT NULL	Source URL
content	TEXT	NULL	Full content
summary	TEXT	NULL	AI summary
source	VARCHAR(100)	DEFAULT 'Microsoft Learn'	Content source
content_type	VARCHAR(50)	DEFAULT 'article'	Document type
category	VARCHAR(100)	DEFAULT 'General'	Category
tags	JSONB	DEFAULT []	Classification tags
extracted_commands	JSONB	DEFAULT []	Found cmdlets
extracted_functions	JSONB	DEFAULT []	Found functions
extracted_modules	JSONB	DEFAULT []	Found modules
metadata	JSONB	DEFAULT {}	Additional metadata
crawled_at	TIMESTAMP	DEFAULT NOW()	Crawl time
last_updated	TIMESTAMP	NULL	Source update time
created_at	TIMESTAMP	DEFAULT NOW()	Creation time
updated_at	TIMESTAMP	DEFAULT NOW()	Last update time

Relationships

Entity Relationship Diagram

```
┌──────────┐      ┌──────────┐      ┌───────────┐
│   User   │──1:N──│  Script  │──1:1──│ Analysis  │
└──────────┘      └──────────┘      └───────────┘
     │                 │
     │                 │
     │                 ├──1:N──  ┌───────────────┐
     │                 │         │ ScriptVersion │
     │                 │         └───────────────┘
     │                 │
     │                 ├──1:1──  ┌───────────────┐
     │                 │         │ Embedding     │
     │                 │         └───────────────┘
     │                 │
     │                 ├──M:N──  ┌──────────┐
     │                 │         │   Tag    │
     │                 │         └──────────┘
     │                 │
     │                 ├──N:1──  ┌───────────┐
     │                 │         │ Category  │
     │                 │         └───────────┘
     │                 │
     │                 └──1:N──  ┌───────────────┐
     │                           │ExecutionLog   │
     │                           └───────────────┘
     │
     └──1:N──  ┌──────────────┐
               │ ChatHistory  │
               └──────────────┘
```

Cascade Behaviors

Parent	Child	On Delete
User	Script	CASCADE
User	ChatHistory	CASCADE
User	ExecutionLog	SET NULL
Script	ScriptAnalysis	CASCADE
Script	ScriptVersion	CASCADE
Script	ScriptEmbedding	CASCADE
Script	ScriptTag	CASCADE
Script	ExecutionLog	CASCADE
Category	Script	SET NULL
Tag	ScriptTag	CASCADE

Indexes

Primary Indexes

Table	Index Name	Columns	Type
scripts	idx_scripts_category	category_id	B-tree
scripts	idx_scripts_user	user_id	B-tree
scripts	idx_scripts_file_hash	file_hash	B-tree
script_versions	idx_script_versions_script	script_id	B-tree
script_analysis	idx_script_analysis_script	script_id	B-tree
execution_logs	idx_execution_logs_script	script_id	B-tree
execution_logs	idx_execution_logs_user	user_id	B-tree
script_embeddings	script_embeddings_idx	embedding	IVFFlat

User Security Indexes

Index Name	Columns	Purpose
idx_users_email	email	Login lookup
idx_users_username	username	Username search
idx_users_role	role	Role filtering
idx_users_email_password	email, password_hash	Composite login
idx_users_locked_until	locked_until	Lockout queries
idx_users_login_attempts	login_attempts	Security monitoring
idx_users_last_login	last_login_at	Activity tracking
idx_users_created_at	created_at	Registration analytics

Chat History Indexes

Index Name	Columns	Purpose
chat_history_user_id_idx	user_id	User's conversations
chat_history_created_at_idx	created_at	Time-based queries

Migrations

Migration Files (Chronological)

File	Purpose	Status
schema.sql	Initial schema creation	Base
add_file_hash_to_scripts.sql	File deduplication support	Applied
add_command_details_to_script_analysis.sql	Enhanced analysis fields	Applied
add_agent_tables.sql	Agentic AI system tables	Applied
add_account_lockout.sql	User security features	Applied
add_performance_indexes.sql	Query optimization	Applied
04_create_chat_history_table.sql	Chat support	Applied
add_updated_at_to_categories.sql	Category timestamps	Applied
update_categories.sql	Category updates	Applied
seed_default_user.sql	Default user creation	Applied
20260107-pgvector-upgrade.sql	pgvector 0.8.0	Applied

Running Migrations

```
# Check migration status
cd src/backend && node run-migration.js status

# Apply all pending migrations
cd src/backend && node run-migration.js up

# Rollback last migration
cd src/backend && node run-migration.js down
```

Caching Strategy

Redis Cache Implementation

Cache Key Patterns:

Pattern	TTL	Purpose
`script:{id}`	600s	Single script cache
`scripts:{page}:{limit}:{filters}`	300s	Paginated list cache
`categories:all`	Indefinite	Category list
`analysis:{scriptId}`	600s	Analysis results
`analytics:*`	3600s	Analytics data

Cache Operations

```
// Get cached value
const cached = await cache.get('script:123');

// Set with TTL (seconds)
await cache.set('script:123', data, 600);

// Delete single key
await cache.del('script:123');

// Clear by pattern
await cache.clearPattern('scripts:*');
```

Fallback Behavior

1. **Redis Available:** Use Redis for all caching
2. **Redis Unavailable:** Automatic in-memory cache fallback
3. **Health Checks:** Every 30 seconds
4. **Reconnection:** Automatic with exponential backoff

Security Features

Password Security

```
 // Password hashing (12 rounds - OWASP 2025 recommended)
const hash = await bcrypt.hash(password, 12);

// Password verification (constant-time)
const valid = await bcrypt.compare(password, hash);
```

Account Lockout

Setting	Value
Max Failed Attempts	5
Lockout Duration	30 minutes
Reset on Success	Yes

SQL Injection Prevention

- All queries use Sequelize ORM
- Parameterized queries for raw SQL
- Input validation on all endpoints

SSL/TLS

Environment	Configuration
Production	`rejectUnauthorized: true` (required)
Development	`rejectUnauthorized: false` (optional)

API Integration

Data Flow: Upload → Analysis → Results

```
 1. Frontend Upload
    └─→ POST /api/scripts/upload/async
        └─→ AsyncUploadController.uploadFiles()
            └─→ Multer validates file
                └─→ Queue upload for processing

2. Background Processing
    └─→ processNextFile()
        └─→ Calculate MD5 hash
            └─→ Check deduplication
                └─→ INSERT Script (if new)
                    └─→ CREATE ScriptVersion v1

3. Analysis (if requested)
    └─→ POST /api/scripts/:id/analyze
        └─→ Fetch script content
            └─→ Call AI service (port 8000)
                └─→ UPSERT ScriptAnalysis
                    └─→ Cache results (Redis)

4. Retrieval
    └─→ GET /api/scripts/:id
        └─→ Check cache
            └─→ Include: User, Category, Tags, Analysis
                └─→ Return with associations
```

Endpoint Summary

Category	**Total**	**Public**	**Auth Required**	**Admin Only**
Scripts	22	8	14	0
Auth	5	3	2	0
Users	6	0	6	3
Analytics	3	0	3	0
Chat	5	1	4	0
Categories	6	3	0	3
Total	**74+**	**21**	**50**	**6**

Performance Considerations

Query Optimization Best Practices (2025-2026)

1. **Use Selective Queries**
2. Avoid `SELECT *`
3. Specify only needed columns

4. Use Sequelize `attributes` option

5. **Pagination**

6. Prefer cursor-based over OFFSET
7. Use keyset pagination for large datasets

8. Limit default page size to 20

9. **Indexing Strategy**

10. Index frequently queried columns
11. Avoid over-indexing (impacts writes)

12. Use partial indexes for filtered queries

13. **Connection Pooling**

14. Current: max 10, min 0
15. Adjust based on load testing

16. Monitor pool utilization

17. **Caching**

18. Cache list queries (5 min TTL)
19. Cache single records (10 min TTL)
20. Invalidate on mutations

Monitoring Recommendations

Metric	Target	Action if Exceeded
Query Time	< 100ms	Add index or optimize
Connection Pool Usage	< 80%	Increase max pool
Cache Hit Rate	> 90%	Adjust TTL or keys
Deadlocks	0/hour	Review transaction scope

Appendix: Quick Reference

Start Database Services

```
# Docker Compose (recommended)
docker-compose up postgres redis

# Check service health
docker-compose ps
```

Verify Connection

```
# PostgreSQL
psql -h localhost -p 5432 -U postgres -d psscript -c "SELECT 1"

# Redis
redis-cli -h localhost -p 6379 ping
```

Common Queries

```
-- Script count by category
SELECT c.name, COUNT(s.id)
FROM categories c
LEFT JOIN scripts s ON s.category_id = c.id
GROUP BY c.id;

-- User activity (last 30 days)
SELECT COUNT(*) FROM scripts
WHERE created_at >= NOW() - INTERVAL '30 days';

-- Security score distribution
SELECT
  CASE
    WHEN security_score >= 8 THEN 'High'
    WHEN security_score >= 5 THEN 'Medium'
    ELSE 'Low'
  END as level,
  COUNT(*)
FROM script_analysis
GROUP BY level;
```

Document generated: January 15, 2026 Next review date: April 15, 2026