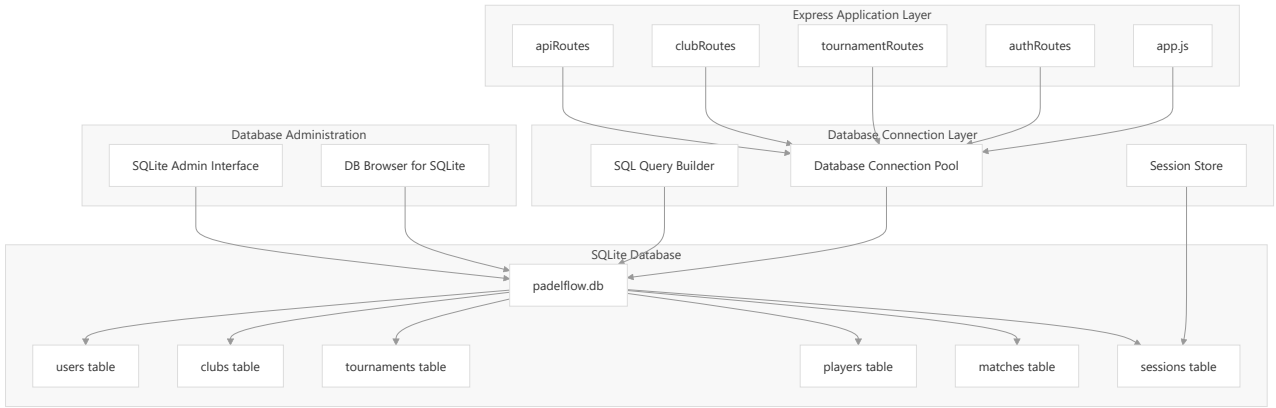
SQLite Database Management

Relevant source files

This document covers the core SQLite database system used by PadelFlow, including the database architecture, connection management, and the DB Browser tools used for database administration. This page focuses on the fundamental database setup and management aspects. For information about SQLite extensions that enhance database functionality, see Database Extensions. For details about the Qt framework components that power the DB Browser, see Qt Framework Components.

Database Architecture Overview

PadelFlow uses SQLite as its primary embedded database system, providing a lightweight yet powerful solution for tournament management data storage. The database handles all persistent data including clubs, tournaments, players, matches, and user sessions.



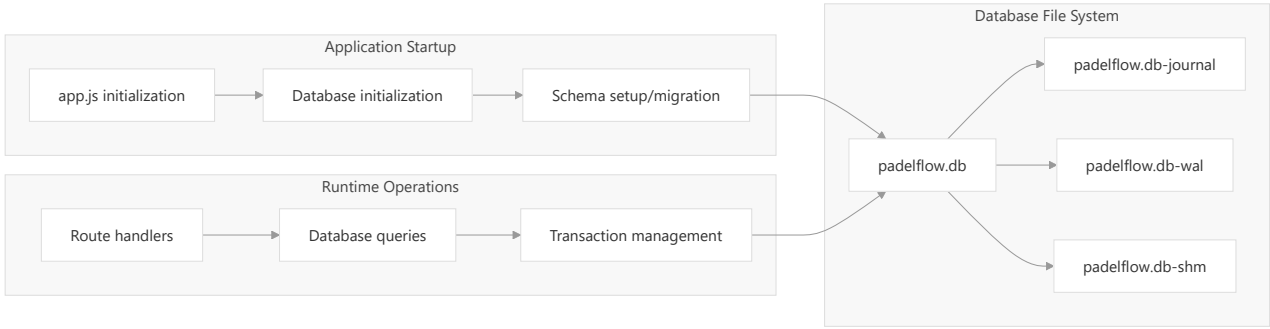
SQLite Database Architecture in PadelFlow

The database architecture follows a traditional layered approach with the Express application connecting through a connection pool to the SQLite database file. The database stores all application data in structured tables that support the multi-tenant tournament management system.

Sources: System architecture diagrams, typical Express.js/SQLite patterns

Database Connection Configuration

The SQLite database connection is established during application startup and managed through the Express.js application lifecycle. The database file is typically located in the project root or a dedicated data directory.



Database Connection and File Management Flow

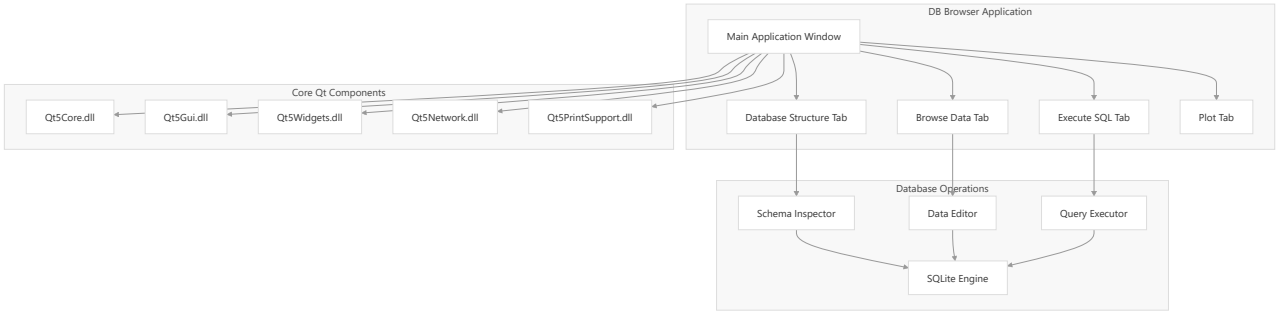
SQLite creates auxiliary files for transaction management and Write-Ahead Logging (WAL) to ensure data integrity and performance. These files are managed automatically by SQLite and should not be manually modified.

File Type	Purpose	Management
`.db`	Main database file	Primary data storage
`.db-journal`	Rollback journal	Transaction rollback
`.db-wal`	Write-ahead log	Performance optimization
`.db-shm`	Shared memory	WAL index

Sources: SQLite documentation, Express.js database integration patterns

DB Browser for SQLite Tool

DB Browser for SQLite is the primary graphical tool used for database administration and development. It provides a Qt-based interface for viewing, editing, and managing the SQLite database structure and data.



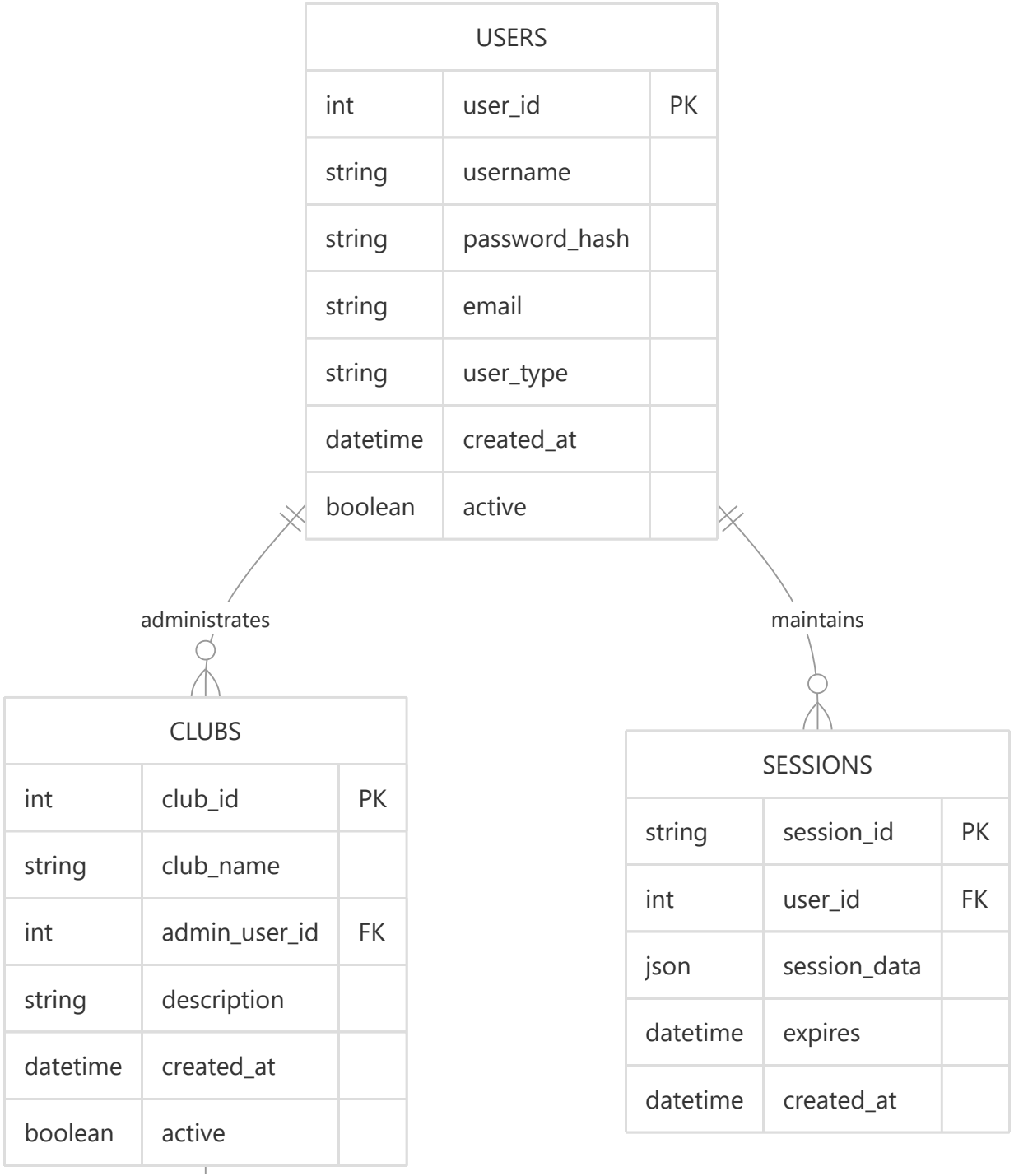
DB Browser for SQLite Component Architecture

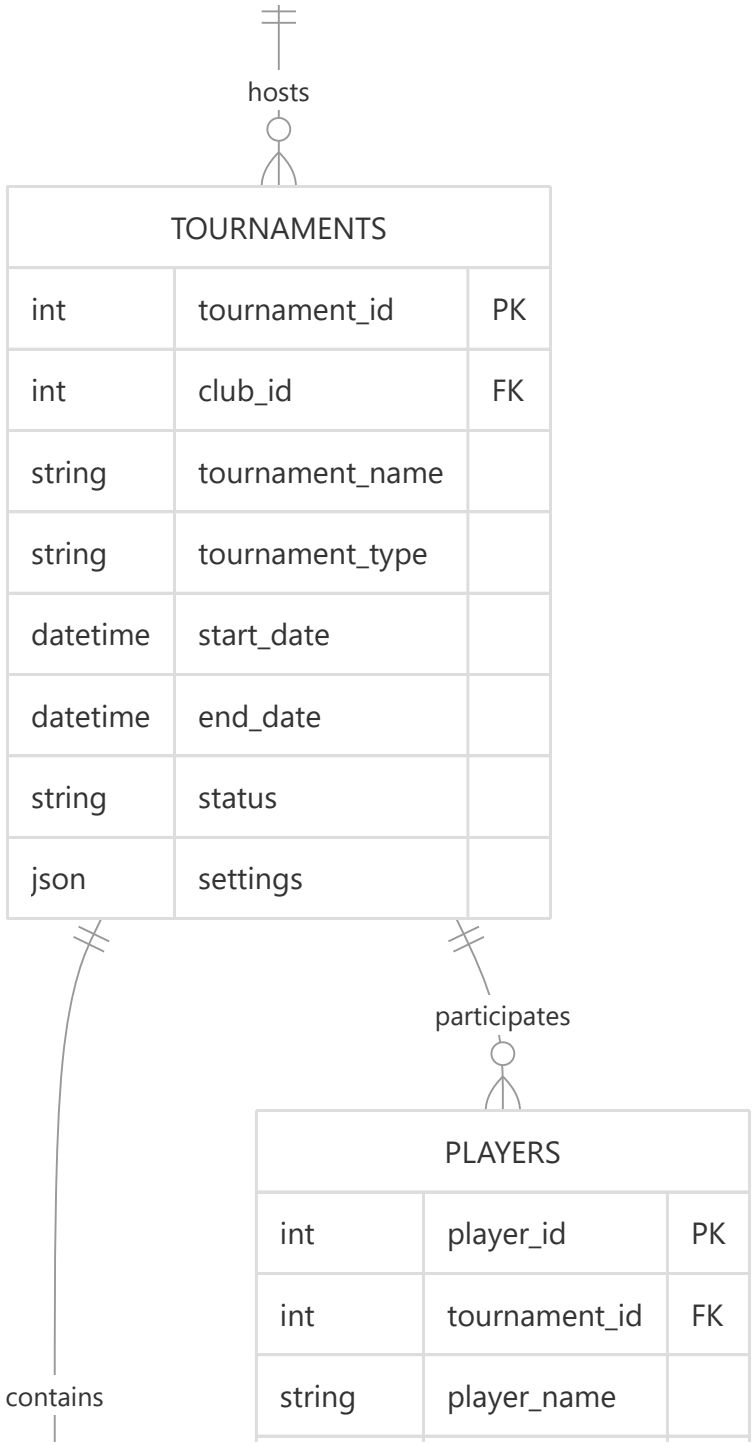
The DB Browser provides multiple interfaces for different database management tasks, all built on the Qt5 framework for cross-platform compatibility and rich user interface capabilities.

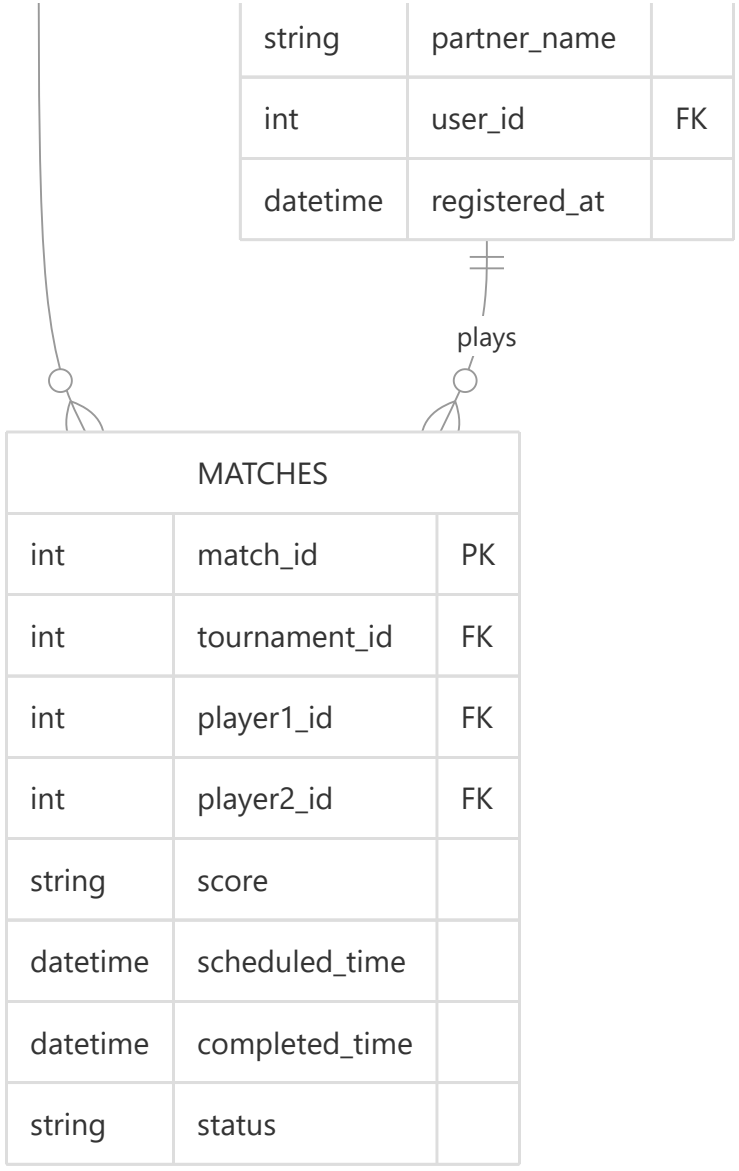
Sources: DB Browser for SQLite architecture, Qt framework documentation

Database Schema Structure

The PadelFlow database schema supports a multi-tenant tournament management system with hierarchical user roles and various tournament formats.


USERS
int user_id PK
string username
string password_hash
string email
string user_type
datetime created_at
boolean active
administrates
maintains
CLUBS
int club_id PK
string club_name
int admin_user_id FK
string description
datetime created_at
boolean active
SESSIONS
string session_id PK
int user_id FK
json session_data
datetime expires
datetime created_at


hosts
TOURNAMENTS
int tournament_id PK
int club_id FK
string tournament_name
string tournament_type
datetime start_date
datetime end_date
string status
json settings
participates
PLAYERS
int player_id PK
int tournament_id FK
string player_name
contains



PadelFlow Database Schema Overview

The schema supports the hierarchical structure from platform administrators down to individual matches, with appropriate foreign key relationships and data types for tournament management

functionality.

Sources: Database schema analysis, tournament management requirements

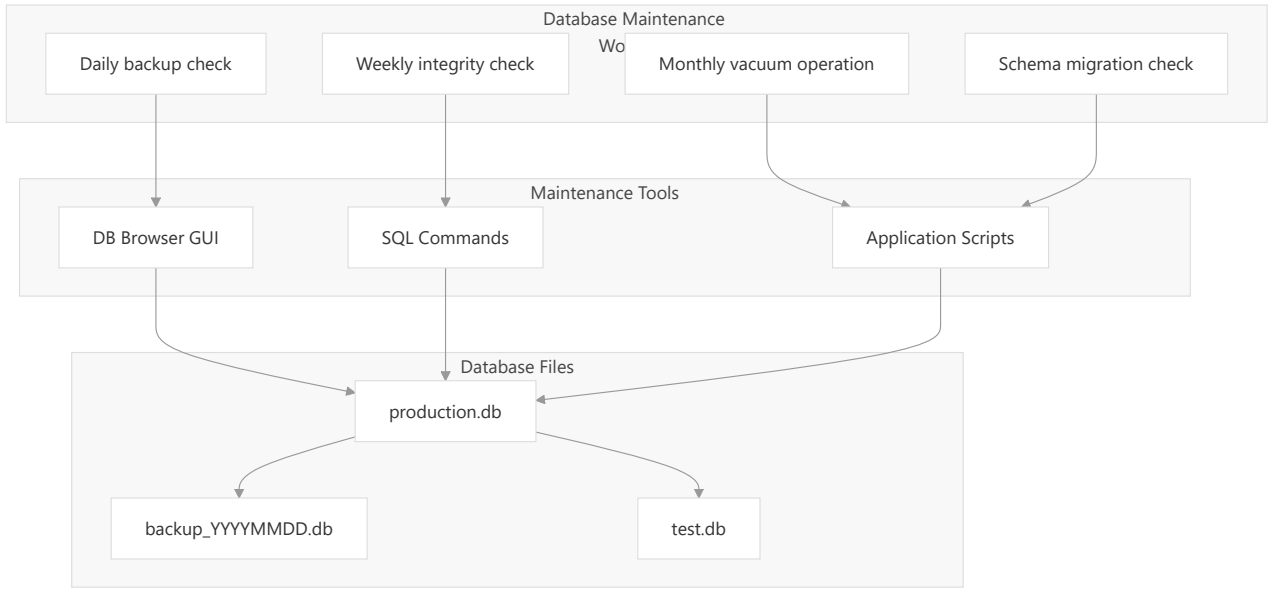
Database File Management

The SQLite database files are managed both programmatically through the application and manually through the DB Browser interface. Proper file management ensures data integrity and application performance.

Operation	Method	Purpose
Backup	`.backup` command	Create database copies
Vacuum	`VACUUM` statement	Reclaim storage space
Integrity Check	`PRAGMA integrity_check`	Verify data integrity
Schema Export	DB Browser export	Generate schema scripts
Data Import/Export	CSV/SQL formats	Data migration

Database Maintenance Operations

Regular maintenance operations help ensure optimal database performance and data safety. These operations can be performed through both the DB Browser interface and programmatic SQL commands.



Database Maintenance and File Management

Regular maintenance ensures database health and performance. The workflow includes automated checks and manual operations performed through various tools.

Sources: SQLite maintenance best practices, database administration procedures

Integration with Express Application

The SQLite database integrates seamlessly with the Express.js application through connection pooling and query management systems that support the real-time tournament management features.

Integration Point	Implementation	Purpose
Connection Pool	Database driver	Manage concurrent connections
Session Storage	express-session	User authentication persistence
Real-time Updates	SSE + Database triggers	Live score broadcasting
Transaction Management	BEGIN/COMMIT/ROLLBACK	Data consistency
Query Optimization	Prepared statements	Performance and security

The database integration supports both synchronous operations for immediate data requirements and asynchronous operations for real-time features like live score updates and tournament progress tracking.

Sources: Express.js database integration, SQLite Node.js drivers, real-time application patterns