PostgreSQL-Only Migration Summary

Overview

This document summarizes the changes made to enforce PostgreSQL as the only supported database for all environments (local, test, and production). SQLite is no longer supported.

Date: October 25, 2025
Migration Reason: Standardize on PostgreSQL for consistency across all environments and eliminate SQLite-specific code paths.

Changes Made

1. Configuration Files

app/config.py

Changes:

- Removed SQLite fallback logic
- Added strict PostgreSQL validation
- Database URL construction now only supports PostgreSQL
- Raises `ValueError` if DATABASE_URL is not set or not PostgreSQL
- Updated error messages to reference local setup guide

Key Changes:

```
# BEFORE: Would fall back to SQLite in dev environments
DATABASE_URL = os.getenv("DATABASE_URL") or "sqlite:///./app.db"

# AFTER: Requires PostgreSQL for ALL environments
DATABASE_URL = os.getenv("DATABASE_URL") or
_construct_azure_postgres_url() or None
if not DATABASE_URL:
    raise ValueError("DATABASE_URL must be set for ALL environments...")
if not DATABASE_URL.startswith("postgresql"):
    raise ValueError("Only PostgreSQL databases are supported...")
```

app/db/base.py

Changes:

- Removed all SQLite-specific code
- Removed SQLite fallback logic
- Simplified engine creation to PostgreSQL-only
- Enhanced error messages and documentation

- Removed `connect_args={"check_same_thread": False}` (SQLite-specific)

Key Changes:

```
# BEFORE: Had conditional logic for SQLite vs PostgreSQL
if database_url.startswith("sqlite"):
    return create_engine(database_url, connect_args=
{"check_same_thread": False})
else:
    return create_engine(database_url, pool_size=5, ...)

# AFTER: Only PostgreSQL configuration
if not database_url.startswith("postgresql"):
    raise ValueError("Only PostgreSQL databases are supported...")
return create_engine(database_url, pool_size=5, max_overflow=10, ...)
```

2. Environment Files

.env.test

Changes:

- Updated DATABASE_URL from SQLite in-memory to PostgreSQL
- Added clear instructions for test database setup

Before:

```
DATABASE_URL=sqlite:///:memory:
```

After:

```
DATABASE_URL=postgresql://postgres:postgres@localhost:5432/eventplanner_
test
```

.env.example (NEW)

Created comprehensive environment template with:

- PostgreSQL DATABASE_URL examples
- Clear instructions for local setup
- Reference to LOCAL_POSTGRES_SETUP.md
- All required and optional configuration variables

3. Documentation

docs/LOCAL_POSTGRES_SETUP.md (NEW)

Created comprehensive setup guide covering:

1. **Installation Instructions**
    - macOS (Homebrew and Postgres.app)
    - Linux (Ubuntu/Debian)
    - Windows (Installer and Docker)
    - Docker (all platforms)
2. **Database Setup**
    - Accessing PostgreSQL
    - Creating databases
    - User management
    - Connection verification
3. **Application Configuration**
    - Environment file setup
    - Connection string format
    - SSL configuration
4. **Troubleshooting**
    - Common issues and solutions
    - Connection problems
    - Authentication errors
    - Permission issues
5. **Testing Setup**
    - Test database creation
    - Test configuration
    - Running tests
6. **Docker Compose Alternative**
    - Complete docker-compose.yml example
    - Separate dev and test databases
7. **Useful Commands**
    - PostgreSQL CLI commands
    - Backup and restore
    - Database management

Migration Path for Developers

For Local Development

1. **Install PostgreSQL**

```
# macOS
brew install postgresql@15
brew services start postgresql@15

# Or use Docker
docker run --name eventplanner-postgres \
  -e POSTGRES_PASSWORD=postgres \
  -e POSTGRES_DB=eventplanner \
  -p 5432:5432 -d postgres:15
```

2. **Create Databases**

```
createdb eventplanner
createdb eventplanner_test
```

3. **Update .env File**

```
cp .env.example .env
# Edit .env and set:
DATABASE_URL=postgresql://postgres:postgres@localhost:5432/eventpla
nner
```

4. **Run Migrations**

```
alembic upgrade head
```

5. **Start Application**

```
uvicorn app.main:app --reload
```

For Testing

1. **Ensure Test Database Exists**

```
createdb eventplanner_test
```

2. **Update .env.test**

```
DATABASE_URL=postgresql://postgres:postgres@localhost:5432/eventpla
nner_test
```

3. **Run Tests**

```
pytest
```

For Production (Azure)

No changes needed for Azure deployment as it was already using PostgreSQL. Ensure:

- DATABASE_URL is set in Azure App Service configuration
- Connection uses SSL (`sslmode=require`)
- Appropriate connection pool settings are configured

Breaking Changes

What No Longer Works

1. **SQLite Databases**
   - Local SQLite files (e.g., `app.db`, `./test.db`)
   - In-memory SQLite (`:memory:`)
   - Any `sqlite:///` connection strings
2. **Missing DATABASE_URL**
   - Application will fail to start without DATABASE_URL
   - No automatic fallback to SQLite
3. **Non-PostgreSQL Databases**
   - MySQL, MariaDB, or other databases are not supported
   - Only `postgresql://` URLs are accepted

Migration Required

All developers must:

1. Install PostgreSQL locally
2. Create local databases
3. Update their `.env` files
4. Run database migrations

Validation

Application Startup Validation

The application now validates database configuration at startup:

1. **app/config.py** validates:
    - DATABASE_URL is set
    - DATABASE_URL starts with `postgresql`
    - Provides helpful error messages with setup guide reference
2. **app/db/base.py** validates:
    - Database URL is not empty
    - Database URL is PostgreSQL
    - Connection can be established

Error Messages

Clear error messages now guide users to the setup documentation:

```
ValueError: DATABASE_URL must be set for ALL environments.
PostgreSQL is required for local development, testing, and production.
Please set DATABASE_URL in your environment or .env file.
For local development, see docs/LOCAL_POSTGRES_SETUP.md
```

Testing

Test Changes Required

Tests that previously relied on in-memory SQLite now need:

1. PostgreSQL test database
2. Updated test fixtures if database-specific
3. Proper cleanup between tests

Test Database Setup

```
# Create test database
createdb eventplanner_test

# Update .env.test
DATABASE_URL=postgresql://postgres:postgres@localhost:5432/eventplanner_
test

# Run tests
pytest
```

Benefits

1. **Consistency**: Same database in all environments
2. **Production Parity**: Dev/test environments match production
3. **Better Testing**: Tests run against production-like database
4. **Feature Parity**: Use PostgreSQL-specific features everywhere
5. **Simplified Code**: No SQLite-specific code paths
6. **Performance**: Better connection pooling and performance tuning

Rollback Plan

If rollback is needed:

1. Restore previous versions of `app/config.py` and `app/db/base.py`
2. Update `.env.test` to use SQLite
3. Revert documentation changes

However, **rollback is not recommended** as:

- PostgreSQL provides better production parity
- SQLite limitations can cause issues in development
- Migration is straightforward with provided documentation

Support

Resources

- **Setup Guide**: `docs/LOCAL_POSTGRES_SETUP.md`
- **Environment Template**: `.env.example`
- **Test Configuration**: `.env.test`
- **PostgreSQL Docs**: https://www.postgresql.org/docs/

Common Issues

See "Common Issues and Solutions" section in `docs/LOCAL_POSTGRES_SETUP.md`

Getting Help

1. Review error messages (they reference the setup guide)
2. Check `docs/LOCAL_POSTGRES_SETUP.md`
3. Verify PostgreSQL is running: `psql -l`
4. Test connection: `python scripts/test_postgres_connection.py`

Files Modified

Core Application Files

- `app/config.py` - Database configuration and validation
- `app/db/base.py` - Database engine creation

Environment Configuration

- `.env.test` - Test environment configuration
- `.env.example` - Environment template (new file)

Documentation

- `docs/LOCAL_POSTGRES_SETUP.md` - PostgreSQL setup guide (new file)
- `POSTGRESQL_ONLY_MIGRATION_SUMMARY.md` - This file (new)

No Changes Needed

- Database models (`app/db/models*.py`)
- Migration files (`migrations/versions/*.py`)
- Application logic (works with PostgreSQL without changes)
- Azure deployment configuration (already using PostgreSQL)

Verification Checklist

Before considering migration complete, verify:

- [ ] PostgreSQL installed locally
- [ ] Local database created (`eventplanner`)
- [ ] Test database created (`eventplanner_test`)
- [ ] `.env` file updated with PostgreSQL URL
- [ ] `.env.test` file updated with PostgreSQL URL
- [ ] Migrations run successfully (`alembic upgrade head`)
- [ ] Application starts without errors
- [ ] Tests pass with PostgreSQL
- [ ] Development workflow documented
- [ ] Team notified of changes

Timeline

- **Planning**: Review current SQLite usage
- **Implementation**: Update configuration and validation
- **Documentation**: Create setup guides and examples
- **Testing**: Verify all environments work
- **Deployment**: No deployment changes needed (already PostgreSQL)
- **Developer Migration**: Developers update local environments

Conclusion

The migration to PostgreSQL-only is complete and provides:

- Consistent development environment
- Better production parity
- Simplified codebase
- Clear documentation for setup
- Helpful error messages guiding users

All developers should follow the setup guide in `docs/LOCAL_POSTGRES_SETUP.md` to update their local environments.

Last Updated: October 25, 2025
Status: Complete
Impact: Breaking change for local development (requires PostgreSQL setup)