Plan for a Production Database Server

Production-quality database server products available on PythonAnywhere:

PythonAnywhere offers support for MySQL and PostgreSQL, both of which are production-quality relational database management systems (RDBMS). These databases are suitable for a wide range of applications and offer more advanced features, scalability, and performance compared to SQLite3.

Recommended database server product for EZU:

For EZU, I would recommend PostgreSQL. It offers several advantages over MySQL, such as better compliance with SQL standards, support for advanced data types, and more advanced indexing options. Additionally, PostgreSQL is known for its extensibility, performance, and reliability, making it an excellent choice for production environments.

Additional product choices with a paying account:

With a paying account, you will still have access to MySQL and PostgreSQL on PythonAnywhere. However, having a paid account allows you to access more resources, such as increased CPU and memory allocations, better support, and custom domain names. These additional resources can help you optimize your database performance and better scale your application.

Further actions to switch from SQLite3 to the recommended database server product:

To switch from SQLite3 to PostgreSQL, you'll need to follow these steps:

a. Create a PostgreSQL database on PythonAnywhere: i. Sign in to your PythonAnywhere account. ii. Go to the "Databases" tab and create a new PostgreSQL database.

b. Install the PostgreSQL adapter for Python (psycopg2) in your virtual environment:

```
phpCopy code
pip install psycopg2-binary
```

c. Update your Django settings to use the PostgreSQL database: In your `settings.py` file, modify the `DATABASES` configuration to use the PostgreSQL database you just created. Replace the existing SQLite3 configuration with the following:

```
bashCopy codeDATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': '<your_db_name>',
        'USER': '<your_pythonanywhere_username>',
        'PASSWORD': '<your_db_password>',
        'HOST': 'localhost',
        'PORT': '',
    }
}
```

d. Run migrations to create tables in the PostgreSQL database:

```
Copy code
python manage.py migrate
```

e. Update your PythonAnywhere WSGI configuration to use the virtual environment with psycopg2 installed.

f. If necessary, transfer any data from your SQLite3 database to the PostgreSQL database using a tool like `django-postgres-copy` or by writing custom data migration scripts.

g. Test your application to ensure everything is working correctly.

With these changes, your application will be using PostgreSQL instead of SQLite3 on the PythonAnywhere server.