# SoftwareEngineering

## Assignment (4)

Created By :
Dheya Al-Hadhrami

Supervisor :
Eng. Malek Al-Mossanif

# Introduction

- إن اختيار قاعدة البيانات قرار معماري استراتيجي يلقي بظلاله على أداء التطبيق ، وقابليته للتوسع، وتكاليف صيانته على المدى الطويل.

- يهدف هذا التقرير إلى تقديم تحليل شامل لأنظمة إدارة قواعد البيانات العلائقية (RDBMS) الأكثر شيوعًا في سياق استخدامها مع إطار عمل Django .

# Report Objectives

- تحليل مقارن لأنظمة إدارة قواعد البيانات العلائقية (RDBMS) .

- تكامل قواعد البيانات مع إطار عمل Django .

- لماذا PostgreSQL .

- تثبيت وربط PostgreSQL مع Django .

# Part I :

# تحليل مقـــــــــارن لأنظمة إدارة قواعد البيانات العلائقية (RDBMS)

## PostgreSQL

نظام كائني-علائقي متقدم، يركز على الامتثال الصارم للمعايير وقابلية التوسع. الخيار الموصى به رسميًا من Django.

## MySQL

"حصان عمل الويب"، يركز على السرعة وسهولة الاستخدام، خاصة في أعباء العمل التي تعتمد على القراءة.

## SQLite

قاعدة بيانات مدمجة عديمة الخادم. مثالية للتطوير والنماذج الأولية والتطبيقات الصغيرة.

## SQL Server

حل بيانات مؤسسي قوي من Microsoft، متكامل بعمق مع النظام البيئي لـ Windows و Azure.

## PostgreSQL

أداء ممتاز في الاستعلامات المعقدة وعمليات القراءة/الكتابة المتزامنة. يستخدم نموذج "عملية لكل اتصال" الذي يعزز العزل والاستقرار.

## MySQL

سريع جدًا في عمليات القراءة البسيطة. يستخدم نموذج "خيط لكل اتصال" الأكثر كفاءة في استخدام الذاكرة للاتصالات الكثيرة.

## SQLite

سريعة جدًا للوصول المحلي. تستخدم قفلًا على مستوى قاعدة البيانات للكتابة، مما يحد من تزامن الكتابة ويجعلها غير مناسبة للإنتاج.

## SQL Server

أداء قوي ومُحسّن لبيئات Windows. يوفر آليات قفل متقدمة و MVCC.

## PostgreSQL

دعم غني لأنواع البيانات المتقدمة مثل JSONB, المصفوفات, HSTORE, والبيانات الجغرافية عبر PostGIS. قابلية توسع ممتازة.

## MySQL

بنية محركات تخزين قابلة للتبديل (InnoDB, MyISAM). نسخ متماثل ناضج وسهل الإعداد لتوسيع نطاق القراءة.

## SQLite

بسيطة للغاية، لا تتطلب أي إدارة. قاعدة البيانات بأكملها في ملف واحد، مما يجعلها محمولة بشكل لا يصدق.

## SQL Server

مجموعة واسعة من الأدوات المدمجة لذكاء الأعمال (BI) والتقارير وتكامل البيانات (SSRS, SSIS, SSAS).

## PostgreSQL

ترخيص PostgreSQL متساهل (شبيه بـ MIT)، مجاني بالكامل للاستخدام التجاري وغير التجاري. يقوده المجتمع.

## MySQL

ترخيص مزدوج: نسخة مجتمع (GPL) ونسخ تجارية مدفوعة من Oracle. قد تفرض رخصة GPL قيودًا على التوزيع التجاري.

## SQLite

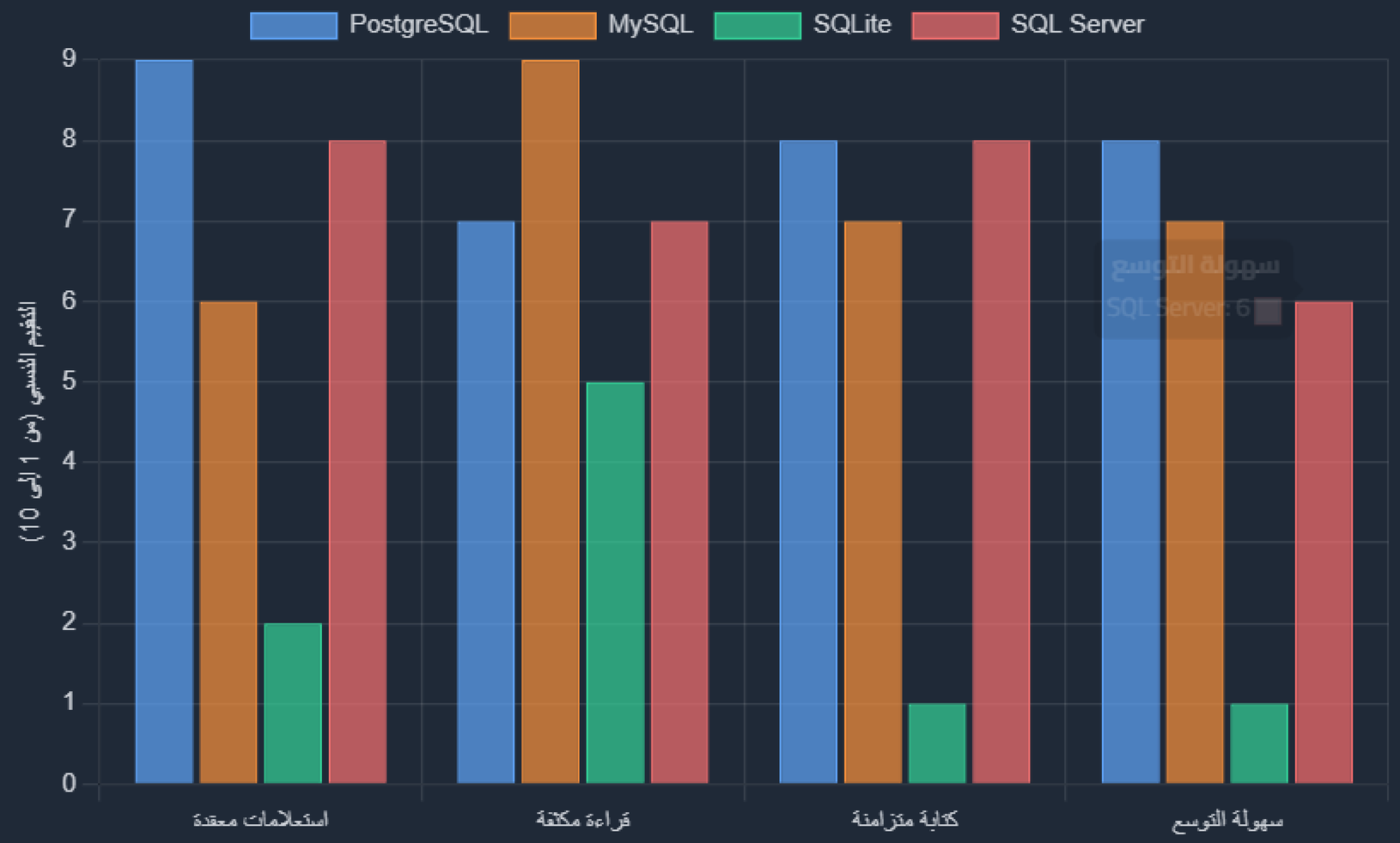
ملكية عامة. لا توجد أي حقوق نشر على الإطلاق، مجانية تمامًا لأي استخدام.

## SQL Server

منتج تجاري يتطلب تراخيص باهظة الثمن. توجد إصدارات مجانية (Express, Developer) بقيود كبيرة.

# تصور الأداء (نموذج توضيحي)

رسم بياني يوضح نقاط القوة النسبية لكل قاعدة بيانات في أعباء عمل مختلفة.



# Part II :

# متى تختار كل قاعدة بيانات ؟!!

## اختر SQLite عندما...

- تبني نموذجًا أوليًا أو إثبات مفهوم.
- تطور تطبيقًا مكتبيًا أو محمولًا.
- أنت في مراحل التعلم والتطوير الأولية.

## اختر MySQL عندما...

- تطبيقك يعتمد بشكل كبير على القراءة.
- تحتاج لإعداد سريع ضمن حزمة ويب قياسية (LAMP).
- فريقك لديه خبرة واسعة في MySQL.

## اختر PostgreSQL عندما...

- تبني تطبيقًا جادًا ومعقدًا يهدف للنمو.
- تكامل البيانات والموثوقية هي الأولوية.
- تحتاج لميزات متقدمة مثل JSONB أو PostGIS.

## اختر SQL Server عندما...

- تعمل في بيئة مؤسسية تعتمد على تقنيات Microsoft.
- تحتاج لتكامل وثيق مع أدوات BI من Microsoft.
- الميزانية تسمح بتكاليف الترخيص التجاري.

# Part III :

# تكامل قواعد البيانات مع إطار عمل ( Django )

# تكامل قواعد البيانات مع إطار عمل Django

- يدعم Django رسميًا أربع قواعد بيانات علائقية: SQLite, MySQL, PostgreSQL, و Oracle.

- يتم تحديد قاعدة البيانات المستخدمة من خلال إعداد ENGINE داخل متغير DATABASES في ملف settings.py.

-هذا الإعداد يخبر Django بالواجهة الخلفية (backend) التي يجب استخدامهـــــا لترجمة أوامر ORM إلى استعلامات SQL مناسبة.

# Part IV :

# كيف يتم اعداد ملف settings.py ؟!!

SQL Server PostgreSQL MySQL **SQLite**

**Required Library:**

```
# (Built-in with Python)
```

**`settings.py` Configuration:**

```
import os
# ...
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

SQL Server PostgreSQL **MySQL** SQLite

**Required Library:**

```
pip install mysqlclient
```

**`settings.py` Configuration:**

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'mydatabase',
        'USER': 'myuser',
        'PASSWORD': 'mypassword',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

SQL Server PostgreSQL MySQL SQLite

**Required Library:**

```
pip install psycopg2-binary
```

**`settings.py` Configuration:**

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'mydatabase',
        'USER': 'myuser',
        'PASSWORD': 'mypassword',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

# SQL Server PostgreSQL MySQL SQLite

**Required Library:**

```
pip install mssql-django
```

**`settings.py` Configuration:**

```
DATABASES = {
    'default': {
        'ENGINE': 'mssql',
        'NAME': 'mydatabase',
        'USER': 'myuser',
        'PASSWORD': 'mypassword',
        'HOST': 'localhost',
        'PORT': '1433',
        'OPTIONS': {
            'driver': 'ODBC Driver 17 for SQL Server',
        },
    },
}
```

# Part V :

# لماذا PostgreSQL ؟

توصي وثائق Django رسميًا باستخدام PostgreSQL للمشاريع الجادة. لهذه الأسباب الرئيسية :

### اكتمال الميزات

يدعم مجموعة واسعة من ميزات SQL المتقدمة التي تتكامل بسلاسة مع Django ORM.

### الموثوقية وتكامل البيانات

يضمن الامتثال الكامل لخصائص ACID، مما يجعله موثوقًا للغاية للبيانات الحساسة.

### أنواع بيانات غنية

دعم أصلي لأنواع متقدمة مثل JSONB والمصفوفات، والتي تتوافق مع حقول Django المخصصة.

### أداء قوي تحت الحمل

يتعامل بكفاءة عالية مع عمليات الكتابة والقراءة المتزامنة بفضل آلية MVCC.

# Thank You

Looking forward to seeing you
in the second assignment ;)